Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2021

V. Vasiliev
Google

June 30, 2020

WebTransport over QUIC
draft-vvv-webtransport-quic-02

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 QuicTransport, a transport protocol that uses a dedicated QUIC [QUIC] connection and provides support for unidirectional streams, bidirectional streams and datagrams.

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/vasilvv/webtransport/issues>. The web API draft corresponding to this document can be found at <https://wicg.github.io/web-transport/>.

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 https://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 1, 2021.

Table of Contents

1. Introduction

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 QuicTransport, a transport protocol that uses a dedicated QUIC [QUIC]

connection and provides support for unidirectional streams, bidirectional streams and datagrams.

QUIC [QUIC] is a UDP-based multiplexed secure transport. It is the underlying protocol for HTTP/3 [I-D.ietf-quic-http], and as such is reasonably expected to be available in web browsers and server-side web frameworks. This makes it a compelling transport to base a WebTransport protocol on.

This document defines QuicTransport, a protocol conforming to the WebTransport protocol framework. QuicTransport is an application protocol running directly over QUIC. The protocol is designed to have low implementation overhead on the server side, meaning that server software that already has a working QUIC implementation available would not require large amounts of code to implement QuicTransport. Where possible, WebTransport concepts are mapped directly to the corresponding QUIC concepts.

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]. The diagrams describe encoding following the conventions described in Section 1.3 of [QUIC].

2. Protocol Overview

Each instance of QuicTransport uses a single dedicated QUIC connection. This allows the peers to exercise a greater level of control over the way their data is being transmitted. However, this also means that multiple instances of QuicTransport cannot be pooled, and thus do not benefit from sharing a congestion controller with other connections.

QuicTransport is designed to be a minimal extension of QUIC, and as such does not provide much higher-level functionality, such as pooling, exchanging metadata at session establishment, redirects, and other similar capabilties not provided by QUIC itself. Http3Transport [I-D.vvv-webtransport-http3] can be used in situations where these features are desired.

When a client requests a QuicTransport session to be created, the user agent establishes a QUIC connection to the specified address.

It verifies that the the server is a QuicTransport endpoint using ALPN, and additionally sends a client indication containing the requested path and the origin of the initiating website to the server. At that point, the connection is ready from the client's perspective. The server MUST wait until the client indication is received before processing any application data.

WebTransport streams are provided by creating an individual unidirectional or bidirectional QUIC stream. WebTransport datagrams are provided through the QUIC datagram extension [QUIC-DATAGRAM].

3. Connection Establishment

In order to establish a QuicTransport session, a QUIC connection must be established. From the client perspective, the session becomes established when the client both have received a TLS Finished message from the server and has sent a client indication. From the server perspective, the session is established after the client indication has been successfully processed.

3.1. Identifying as QuicTransport

In order to identify itself as a WebTransport application, QuicTransport relies on TLS Application-Layer Protocol Negotiation [RFC7301]. The user agent MUST request the ALPN value of "wq-vvv-01" and it MUST close the connection unless the server confirms that ALPN value.

3.2. Client Indication

In order to verify that the client's origin is allowed to connect to the server in question, the user agent has to communicate the origin to the server. This is accomplished by sending a special message, called client indication, on stream 2, which is the first client-initiated unidirectional stream.

The client indication is a sequence of key-value pairs that are formatted in the following way:

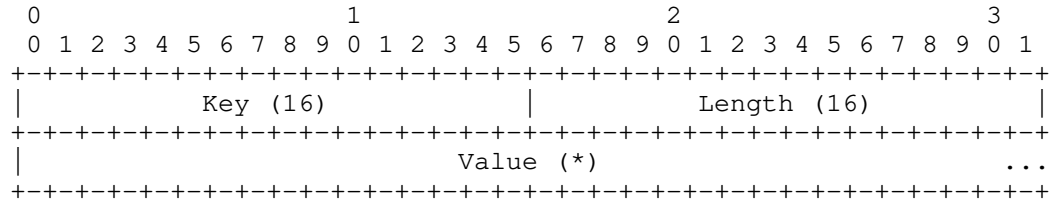

Figure 1: Client indication format

The pair includes the following fields:

Key: Indicates the field that is being expressed.

Length: Indicates the length of the value (the length of the key and the length itself are not included).

Value: The value of the field, the semantics of which are determined by the key.

A FIN on the stream 2 SHALL indicate that the message is complete. The client MUST send the entirety of the client indication and a FIN immediately after opening the connection. The server MUST NOT process any application data before receiving the entirety of the client indication. The total length of the client indication MUST NOT exceed 65,535 bytes.

In order to ensure that the user agent can send the client indication immediately, the server MUST set "initial_max_streams_uni" transport parameter to at least "1". The user agent MUST close the connection if the server sets "initial_max_streams_uni" to "0".

The server MUST ignore any field it does not recognize. All of the fields MUST be unique; the server MAY close the connection if any of the keys is used more than once.

3.2.1. Origin Field

In order to allow the server to enforce its origin policy, the user agent has to communicate the origin in the client indication. This can be accomplished using the "Origin" field:

Name: Origin

Key: 0x0000

Description: The origin [RFC6454] of the client initiating the connection.

The user agent MUST send the "Origin" field. The "Origin" field MUST be set to the origin of the client initiating the connection, serialized as described in the "serializing a request origin" section of [FETCH].

3.2.2. Path Field

In order to allow multiplexing multiple application on the same host-port tuple, QuicTransport allows specifying extra routing information in the path component of the URI. That component is communicated using the "Path" field in the client indication:

Name: Path

Key: 0x0001

Description: The path component of the QuicTransport URI.

The user agent MUST send a non-empty "Path" field. When the connection is initiated through a URI Section 6, that value SHALL be the "path-abempty" part, followed by a concatenation of the "?" literal and the "query" componenet if such is present. In case when "path-abempty" is empty, the value sent SHALL be "/".

Unlike HTTP, the "authority" portion of the URL is not communicated in the client indication. As QuicTransport has its own connection dedicated to it, the host name portion can be retrieved from the "server_name" TLS extension [RFC6066].

The server MAY use the value of the "Path" field in any way defined by the target application.

3.3. 0-RTT

QuicTransport provides applications with the ability to use the 0-RTT feature described in [RFC8446] and [QUIC]. 0-RTT allows a client to send data before the TLS session is fully established. It provides lower latency, but has the drawback of being vulnerable to replay attacks. Since only the application can make an informed decision as to whether some data is safe to send in that context, 0-RTT requires the client API to only send data over 0-RTT when specifically requested by the client.

0-RTT support in QuicTransport is OPTIONAL, as it is in QUIC and TLS 1.3.

4. Streams

QuicTransport unidirectional and bidirectional streams are created by creating a QUIC stream of the corresponding type. All other operations (read, write, close) are also mapped directly to the operations defined in [QUIC]. The QUIC stream IDs are the stream IDs that are exposed to the application.

5. Datagrams

QuicTransport uses the QUIC DATAGRAM frame [QUIC-DATAGRAM] to provide WebTransport datagrams. A QuicTransport endpoint MUST negotiate and support the DATAGRAM frame. The datagrams provided by the application are sent as-is.

6. QuicTransport URI Scheme

NOTE: the URI scheme definition in this section is provisional and subject to change, especially the name of the scheme.

QuicTransport uses the "quic-transport" URI scheme for identifying QuicTransport servers.

The syntax definition below uses Augmented Backus-Naur Form (ABNF) [RFC5234]. The definitions of "host", "port", "path-abempty", "query" and "fragment" are adopted from [RFC3986]. The syntax of a QuicTransport URI SHALL be:

```
quic-transport-URI = "quic-transport:" "//"
                              host [ ":" port ]
                              path-abempty
                              [ "?" query ]
                              [ "#" fragment ]
```

The "path-abempty" and the "query" portions of the URI are communicated to the server in the client indication as described in Section 3.2.2. The "quic-transport" URI scheme supports the "/.well-known/" path prefix defined in [RFC8615].

This document does not assign any semantics to the "fragment" portion of the URI. Any QuicTransport implementation MUST ignore those until a subsequent specification assigns semantics to those.

The "host" component MUST NOT be empty. If the "port" component is missing, the port SHALL be assumed to be 0.

In order to connect to a QuicTransport server identified by a given URI, the user agent SHALL establish a QUIC connection to the specified "host" and "port" as described in Section 3. It MUST immediately signal an error to the client if the port value is 0.

NOTE: this effectively requires the port number to be specified. This specification may include an actually usable default port number in the future.

7. Transport Properties

QuicTransport supports most WebTransport features as described in Table 1.

Property	Support
Stream independence	Always supported
Partial reliability	Always supported
Pooling support	Not supported
Connection mobility	Implementation-dependent

Table 1: Transport properties of QuicTransport

8. Security Considerations

QuicTransport 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 the client is potentially untrusted.

QuicTransport uses QUIC with TLS, and as such, provides the full range of security properties provided by TLS, including confidentiality, integrity and authentication of the server.

QUIC is a client-server protocol where a client cannot send data until either the handshake is complete or a previously established session is resumed. This ensures that clients cannot send data to a network endpoint that has not accepted an incoming connection. Furthermore, the QuicTransport session can be immediately aborted by the server through a connection close or a stateless reset, causing the user agent to stop the traffic from the client. This provides a defense against potential denial-of-service attacks on the network by untrusted clients.

QUIC provides a congestion control mechanism [I-D.ietf-quic-recovery] that limits the rate at which the traffic is sent. This prevents potentially malicious clients from overloading the network.

WebTransport requires user agents to continually verify that the server is still interested in talking to them. QuicTransport accomplishes that by virtue of QUIC being an acknowledgement-based protocol; if the client is attempting to send data, and the server

does not send any ACK frames in response, the client side of the QUIC connection will time out.

QuicTransport prevents WebTransport clients from connecting to arbitrary non-Web servers through the use of ALPN. Unlike TLS over TCP, successful ALPN negotiation is mandatory in QUIC. Thus, unless the server explicitly picks the QuicTransport ALPN value, the TLS handshake will fail.

QuicTransport uses a unidirectional QUIC stream to provide the server with the origin of the client.

In order to avoid the use of QuicTransport to scan internal networks, the user agents MUST NOT allow the clients to distinguish different connection errors before the correct ALPN is received from the server.

Since each instance of QuicTransport opens a new connection, a malicious client can cause resource exhaustion, both on the local system (through depleting file descriptor space or other per-connection resources) and on a given remote server. Because of that, user agents SHOULD limit the amount of simultaneous connections opened. The server MAY limit the amount of open connections from a given client.

9. IANA Considerations

9.1. ALPN Value Registration

The following entry is added to the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established by [RFC7301]:

The "wq-vvv-01" label identifies QUIC used as a protocol for WebTransport:

Protocol: QuicTransport

Identification Sequence: 0x77 0x71 0x2d 0x76 0x76 0x76 0x2d 0x30 0x31 ("wq-vvv-01")

Specification: This document

9.2. Client Indication Fields Registry

IANA SHALL add a registry for "QuicTransport Client Indication Fields" registry. Every entry in the registry SHALL include the following fields:

Name: The name of the field.

Key: The 16-bit unique identifier that is used on the wire.

Description: A brief description of what the parameter does.

Reference: The document that describes the parameter.

The IANA policy, as described in [RFC8126], SHALL be Standards Action for values between 0x0000 and 0x03ff; Specification Required for values between 0x0400 and 0xefff; and Private Use for values between 0xf000 and 0xffff.

9.3. URI Scheme Registration

This document contains the request for the registration of the URI scheme "quic-transport". The registration request is in accordance with [RFC7595].

Scheme name: quic-transport

Status: Permanent

Applications/protocols that use this scheme name: QuicTransport

Contact: IETF Chair chair@ietf.org [1]

Change controller: IESG iesg@ietf.org [2]

Reference: Section 6 of this document.

Well-Known URI Support: Section 6 of this document.

10. References

10.1. Normative References

[FETCH] WHATWG, "Fetch Standard", June 2020, <https://fetch.spec.whatwg.org/>.

[OVERVIEW] Vasiliev, V., "The WebTransport Protocol Framework", draft-ietf-webtrans-overview-latest (work in progress).

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-latest (work in progress).

[QUIC-DATAGRAM]
Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", draft-pauly-quic-datagram-latest (work in progress).

[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/info/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/info/rfc3986>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

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

[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/info/rfc7301>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/info/rfc7595>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

10.2. Informative References

[I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-29 (work in progress), June 2020.

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-29 (work in progress), June 2020.

[I-D.vvv-webtransport-http3]
Vasiliev, V., "WebTransport over HTTP/3", draft-vvv-webtransport-http3-01 (work in progress), November 2019.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

10.3. URIs

[1] mailto:chair@ietf.org

[2] mailto:iesg@ietf.org

Author's Address

Victor Vasiliev
Google

Email: vasilvv@google.com