Workgroup: QUIC
Internet-Draft:
draft-seemann-quic-reliable-stream-reset-00
Published: 9 September 2022
Intended Status: Informational
Expires: 13 March 2023
Authors: M. Seemann
Protocol Labs

Reliable QUIC Stream Resets

Abstract

QUIC ([RFC9000]) defines a RESET_STREAM frame to reset a stream. When a sender resets a stream, it stops retransmitting STREAM frames for this stream. On the receiver side, there is no guarantee that any of the data sent on that stream is delivered to the application. This document defines a new QUIC frame, the RELIABLE_RESET_STREAM frame, that resets a stream, while guaranteeing reliable delivery of stream data up to a certain byte offset.

Discussion Venues

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

Discussion of this document takes place on the QUIC Working Group mailing list (quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/.

Source for this draft and an issue tracker can be found at https://github.com/marten-seemann/draft-seemann-quic-reliable-stream-reset.

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 13 March 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

QUIC v1 ([RFC9000]) allows streams to be reset. When a stream is reset, the sender doesn't retransmit stream data for the respective stream. On the receiver side, the QUIC stack is free to surface the stream reset to the application immediately, even if it has already received stream data for that stream.

Application running on top of QUIC might need to send an identifier at the beginning of the stream in order to associate that stream with a specific subpart of the application. For example, WebTransport ([WEBTRANSPORT]) uses a QUIC varint to encode the ID of the WebTransport session.

It is desirable that the receiver is able to associate incoming streams with their respective subpart of the application, even if the QUIC stream is reset before the identifier at the beginning of the stream was read.

This document describes a QUIC extension defining a new frame type, the RELIABLE_RESET_STREAM frame. This frame allows an endpoint to

mark a portion at the beginning of the stream which will then be guaranteed to be delivered to receiver's application, even if the stream was reset.

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. Negotiating Extension Use

Endpoints advertise their support of the extension described in this document by sending the reliable_reset_stream (0x727273) transport parameter (Section 7.2 of [RFC9000]) with an empty value. An implementation that understands this transport parameter **MUST** treat the receipt of a non-empty value as a connection error of type TRANSPORT_PARAMETER_ERROR.

In order to allow reliable stream resets in 0-RTT packets, the client **MUST** remember the value of this transport parameter. If 0-RTT data is accepted by the server, the server **MUST** not disable this extension on the resumed connection.

4. RELIABLE_RESET_STREAM Frame

Conceptually, the RELIABLE_RESET_STREAM frame is a RESET_STREAM frame with an added Reliable Size field.

```
RELIABLE_RESET_STREAM Frame {
  Type (i) = 0x72,
  Stream ID (i),
  Application Protocol Error Code (i),
  Final Size (i),
  Reliable Size (i)
}
```

RELIABLE_RESET_STREAM frames contain the following fields:

Stream ID: A variable-length integer encoding of the stream ID of the stream being terminated.

Application Protocol Error Code: A variable-length integer containing the application protocol error code (see Section 20.2) that indicates why the stream is being closed.

Final Size: A variable-length integer indicating the final size of the stream by the RESET_STREAM sender, in units of bytes; see (Section 4.5 of [RFC9000]).

Reliable Size: A variable-length integer indicating the amount of data that needs to be delivered to the application before the error code can be surfaced, in units of bytes.

If the Reliable Size is larger than the Final Size, the receiver **MUST** close the connection with a connection error of type FRAME_ENCODING_ERROR.

Semantically, a RESET_STREAM frame is equivalent to a RELIABLE_RESET_STREAM frame with the Reliable Size set to 0.

RELIABLE_RESET_STREAM frames are ack-eliciting. When lost, they **MUST** be retransmitted, unless a RESET_STREAM frame or another RELIABLE_RESET_STREAM frame was sent for the same stream (see Section 5.1).

5. Resetting Streams

When resetting a stream, the node has the choice between using a RESET_STREAM frame and a RELIABLE_RESET_STREAM frame. When using a RESET_STREAM frame, the behavior is unchanged from the behavior described in ([RFC9000]).

The initiator **MUST** guarantee reliable delivery of stream data of at least Reliable Size bytes. If STREAM frames containing data up to that byte offset are lost, the initiator **MUST** retransmit this data, as described in (Section 13.3 of [RFC9000]). Data sent beyond that byte offset **SHOULD NOT** be retransmitted.

A receiver that delivers stream data to the application as an ordered byte stream **MUST** deliver all bytes up to the Reliable Size before surfacing the stream reset error. As described in (Section 3.2 of [RFC9000]), it **MAY** deliver data beyond that offset to the application.

5.1. Multiple RELIABLE_RESET_STREAM / RESET_STREAM frames

The initiator **MAY** send multiple RELIABLE_RESET_STREAM frames for the same stream in order to reduce the Reliable Size. It **MAY** also send a RESET_STREAM frame, which is equivalent to sending a RELIABLE_RESET_STREAM frame with a Reliable Size of 0.

When sending multiple frames for the same stream, the iniator **MUST NOT** increase the Reliable Size. When receiving a RELIABLE_RESET_STREAM frame with a lower Reliable Size, the receiver only needs to deliver data up the lower Reliable Size to the

application before surfacing the stream reset error. It **MUST NOT** expect the delivery of any data beyond that byte offset.

Reordering of packets might lead to a RELIABLE_RESET_STREAM frame with a higher Reliable Size to be received after a RELIABLE_RESET_STREAM frame with a lower Reliable Size. The receiver **MUST** ignore any RELIABLE_RESET_STREAM frame that increases the Reliable Size.

When sending another RELIABLE_RESET_STREAM or RESET_STREAM frame for the same stream, the initiator **MUST NOT** change the Application Error Code or the Final Size. If the receiver detects a change in those fields, it **MUST** close the connection with a connection error of type STREAM_STATE_ERROR.

6. Security Considerations

TODO Security

7. IANA Considerations

This document has no IANA actions.

8. 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>.

[WEBTRANSPORT] Frindell, A., Kinnear, E., and V. Vasiliev, "WebTransport over HTTP/3", Work in Progress, Internet-Draft, draft-ietf-webtrans-http3-03, 6 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-http3-03>.

Acknowledgments

TODO acknowledge.

Author's Address

Marten Seemann
Protocol Labs

Email: martenseemann@gmail.com