QUIC Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2018

P. Tiesel
M. Palmer
B. Chandrasekaran
A. Feldmann
TU Berlin
J. Ott
TU Munich

October 30, 2017

Considerations for Unreliable Streams in QUIC
draft-tiesel-quic-unreliable-streams-01

Abstract

This memo outlines how to support unreliable streams as well as partially-reliable streams within QUIC. The intention of this document is to collect requirements and considerations, to frame the design space, and to give an example how unreliable stream support could be realized.

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 May 3, 2018.

Table of Contents

1. Conventions and Definitions

The words "MUST", "MUST NOT", "SHALL", "SHALL NOT", "SHOULD", and "MAY" are used in this document. It's not shouting; when these words are capitalized, they have a special meaning as defined in [RFC2119].

2. Introduction

This memo describes how QUIC can provide reliable, partially-reliable, and unreliable transmissions within the same connection. There are many use cases for unreliable delivery of stream data, e.g., to meet deadlines for data delivery in the presence of short-time congestion by avoiding head-of-line blocking. For partial reliable streams, the sender can decide which frames to retransmit. This model allows applications to request the required kind of reliability on a per-stream level and to mix of reliable and

unreliable transmissions within the same stream. Still, some control data or metadata often needs to be transmitted reliably.

This draft is based on [I-D.draft-ietf-quic-transport-07] with the addition of uni-directional streams (https://github.com/quicwg/base-drafts/pull/885) and variable-length integer encoding (https://github.com/quicwg/base-drafts/pull/877). It gives a comprehensive overview about considerations for adding support for unreliable streams to QUIC in a way that each stream is either fully reliable or unreliable (including partially-reliable). Our implementation proposal in Appendix A needs minimal changes to the wire format - the third least significant bit of the Stream ID is repurposed to signal whether a stream is reliable or partial reliable / unreliable.

3. Modes of Unreliable Transmission

- Reliable transmission suggests that all application data is re-transmitted if lost.
- Unreliable transmission suggests that no application data is re-transmitted if lost.
- Partial reliable transmission suggests that some application data is re-transmitted if lost.

In principle, the way [I-D.draft-ietf-quic-recovery-06] realizes reliable transmission allows to realize all three levels of reliability mentioned above without changing the wire format. The combination of packet-based acknowledgments with stream-based retransmits allows deciding on a per stream-frame or byte range base whether or how a lost stream frame is retransmitted. For this memo, we define unreliable streams as streams, for which some or all lost stream frames are not re-transmitted. Thus, our definition covers fully unreliable as well as partially-reliable transmission.

4. Protocol Considerations

For clear application semantic, the receiver must be able to know whether to rely on the sender to retransmit lost stream data. Therefore, an endpoint opening an unreliable stream MUST indicate that lost stream data might not be retransmitted. The indication must either be carried on each frame, that could be received first by an endpoint, or the indication must be transmitted reliably and acknowledged before the first frame of an unreliable stream is sent.

We anticipate two options to indicate whether a stream is unreliable:

- Indicate unreliable streams within the Stream ID.

- Leave the signaling of unreliable streams completely to the application layer.

The authors advocate for the former option. This approach does not introduce complex interwinding between QUIC and the application.

4.1. Unreliable Stream Support Negotiation

Support of unreliable streams should be optional to reduce the complexity of a minimal QUIC implementation. An endpoint signals its willingness to receiving unreliable stream frames during the TLS handshake using the transport parameter allow_unreliable_streams in analogy to the omit_connection_id flag specified in [I-D.draft-ietf-quic-transport-07]. An application that makes no use of partial delivery of stream data, should not signal willingness to allow unreliable streams.

4.2. Stream as a Message

Unreliable streams are particularly useful if QUIC streams are used as a message abstraction. If the messages can be sent using a single stream frame on a unidirectional stream, the state committed at the sender-side and receiver-side can be minimized, and signaling of stream close can be omitted. The loss of such a frame does not introduce state at the perceived receiver, nor does it require the sender to keep state for retransmission.

4.3. Stream ID 0x0

Data of stream 0x0 MUST be transmitted reliably as TLS expects reliable transmission.

4.4. Congestion Control on Unreliable Streams

Unreliable streams are subject to regular congestion control.

It should be clarified that ACK and RST_STREAM Frames are not subject to congestion control.

4.5. Flow Control on Unreliable Streams

Unreliable streams are subject to regular flow control on connection and stream level.

Note: It is up to further discussion under which assumptions this can be relaxed.

4.6. Stream Open

A receiver may want to treat reliable and unreliable streams differently, e.g., in the way state is represented or how the stream data is presented to the application. Therefore, the receiver needs to know upon the reception of the first stream frame whether the respective stream is reliable or unreliable.

As the first frame sent may be lost or re-ordered, it is not sufficient to mark the first stream frame. Instead, it is necessary to either reliably transmit the fact whether a stream is reliable or unreliable or to have all frames of the stream annotated.

4.7. Retransmission of Partially-Reliable Stream Data

Retransmissions of partially-reliable stream data SHOULD always contain the same data, as was sent in the original transmission.

Note: It is up to further discussion under which assumptions this can be relaxed.

4.8. Stream Close

As frames of unreliable streams may not be retransmitted, the loss of a frame indicating the end of a stream may result in a "zombie" stream state. A QUIC version with unreliable stream support MUST ensure that either such a zombie state does not occur or include a mechanism to clean up streams in such a zombie state, e.g., by using a window of active unreliable stream ids.

The authors advocate for solving this issue by reliably transmitting the final offset of all streams, that consist of more than a single stream frame. The retransmit of the stream end can be achieved by sending an empty stream frame with the final offset and the FIN bit set or by using an RST_STREAM frame. For unidirectional streams that only consist of a single stream frame, retransmission of the final offset is not necessary - See Section 4.2 for details.

5. Application Interface Considerations

5.1. Retransmissions within Unreliable Streams

While unreliable streams suggest just disabling retransmissions for these streams, applications may choose to apply arbitrary retransmission strategies for unreliable streams, e.g., retransmit stream data as long it will likely be delivered on-time with respect to an application provided deadline, or only retransmit certain byte ranges.

A QUIC implementation that implements retransmissions on a per-packet basis, therefore, may retransmit unreliable stream data even if not requested by the application.

5.2. Presentation of Unreliable Streams

The presentation of unreliable streams is application specific. The anticipated use cases include the following.

- Data being delivered annotated with its offset as it is received.
- Data being delivered after a deadline with an annotated list of holes.
- Data being delivered as concatenation of the stream fragments received. This can be useful if the application's framing is aligned with the QUIC stream frames.

5.3. Prioritization of Unreliable Streams

Prioritization of unreliable streams uses the same priority mechanism as reliable streams.

Applications that want UDP-like behavior, and thus want to avoid data sent over unreliable streams queued in send buffers, must ensure that:

- The flow control windows are large enough to send at any given point in time
- The data rate sent in all frames stays below the one permitted by the congestion window.
- The priority of unreliable streams is high enough to transmit data, even if there are retransmissions outstanding on other streams.

To enable writing portable applications, guidelines how prioritization should be handled in a QUIC implementation and how it is exposed to the application are required.

6. Security Considerations

Unreliable Streams open a few additional risks of information disclosure.

- An active, on path attacker can drop selected frames and see whether the transmission timings change to see whether unreliable streams are used.

- Using streams as a message as described in Section 4.2 will most likely result in packets which sizes resemble the size of the individual message contained. If not mitigated, this can disclose the individual message length.

7. IANA Considerations

TBD

8. Acknowledgements

This work has been supported in part by Leibniz Prize project funds of DFG - German Research Foundation: Gottfried Wilhelm Leibniz-Preis 2011 (FKZ FE 570/4-1) and received funding from the European Union's Horizon 2020 research and innovation programme 2014-2018 under grant agreement No. 644866, Scalable and Secure Infrastructures for Cloud Operations (SSICLOPS).

9. Informative References

[I-D.draft-ietf-quic-applicability-01]
Kuehlewind, M. and B. Trammell, "Applicability of the QUIC Transport Protocol", draft-ietf-quic-applicability-01 (work in progress), October 2017.

[I-D.draft-ietf-quic-recovery-06]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-06 (work in progress), September 2017.

[I-D.draft-ietf-quic-transport-07]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-07 (work in progress), October 2017.

[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>.

Appendix A. Implemenation Proposal

This proposal realizes unreliable streams by modifying [I-D.draft-ietf-quic-transport-07] with the addition of uni-directional streams (https://github.com/quicwg/base-drafts/pull/872) and variable-length integer encoding (https://github.com/quicwg/base-drafts/pull/877) to support unreliable streams. It modifies the Stream ID to signal whether a stream is reliable in analogy to the way unidirectional streams are signaled. In addition, it addresses the Stream Close concerns raised in Section 4.8.

A.1. Stream Identifiers

Streams are identified by a variable-length integer, referred to as the Stream ID. The least significant three bits of the Stream ID are used to identify the type of stream (unidirectional or bidirectional), (unreliable or reliable) and the initiator of the stream.

The second least significant bit (0x2) of the Stream ID differentiates between unidirectional streams and bidirectional streams. Unidirectional streams always have this bit set to 1 and bidirectional streams have this bit set to 0.

The third least significant bit (0x4) of the Stream ID differentiates between unreliable and reliable streams. Unreliable streams always have this bit set to 1 and reliable streams have this bit set to 0.

The three type bits from a Stream ID, therefore, identify streams as summarized in Appendix A.2.

Low Bits	Stream Type
0x0	Client-Initiated, Bidirectional , Reliable
0x1	Server-Initiated, Bidirectional , Reliable
0x2	Client-Initiated, Unidirectional, Reliable
0x3	Server-Initiated, Unidirectional, Reliable
0x4	Client-Initiated, Bidirectional , Unreliable
0x5	Server-Initiated, Bidirectional , Unreliable
0x6	Client-Initiated, Unidirectional, Unreliable
0x7	Server-Initiated, Unidirectional, Unreliable

A.2. Stream Close

Streams MUST be explicitly closed. This can either be done by setting the FIN bit on the frame carrying the final offset of the stream or by using an RST_STREAM frame indicating the final offset.

For unreliable streams transmitted using more than one stream frame, the stream close has to be transmitted reliably to prevent zombie stream state. If the packet containing the FIN flagged stream frame is lost, and the data in this stream is not to be retransmitted, the sender can (re-)transmit the stream close by sending an empty FIN flagged stream frame carrying the final offset.

Authors' Addresses

Philipp S. Tiesel
TU Berlin
Marchstr. 23
Berlin
Germany

Email: philipp@inet.tu-berlin.de

Mirko Palmer
TU Berlin
Marchstr. 23
Berlin
Germany

Email: mirko@inet.tu-berlin.de

Balakrishnan Chandrasekaran
TU Berlin
Marchstr. 23
Berlin
Germany

Email: balac@inet.tu-berlin.de

Anja Feldmann
TU Berlin
Marchstr. 23
Berlin
Germany

Email: anja@inet.tu-berlin.de

Joerg Ott
TU Munich
Boltzmannstrasse 3
Garching bei Muenchen
Germany

Email: ott@in.tum.de