ï»¿

An Unreliable Datagram Extension to QUIC

Abstract

   This document defines an extension to the QUIC transport protocol to
   add support for sending and receiving unreliable datagrams over a
   QUIC connection.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   The QUIC transport protocol [RFC9000] provides a secure, multiplexed
   connection for transmitting reliable streams of application data.
   QUIC uses various frame types to transmit data within packets, and
   each frame type defines whether the data it contains will be
   retransmitted. Streams of reliable application data are sent using
   STREAM frames.

   Some applications, particularly those that need to transmit real-time
   data, prefer to transmit data unreliably. In the past, these
   applications have built directly upon UDP [RFC0768] as a transport
   and have often added security with DTLS [RFC6347]. Extending QUIC to
   support transmitting unreliable application data provides another
   option for secure datagrams with the added benefit of sharing the
   cryptographic and authentication context used for reliable streams.

   This document defines two new DATAGRAM QUIC frame types that carry
   application data without requiring retransmissions.

1.1. Specification of Requirements

   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.

2. Motivation

   Transmitting unreliable data over QUIC provides benefits over
   existing solutions:

   * Applications that want to use both a reliable stream and an
      unreliable flow to the same peer can benefit by sharing a single
      handshake and authentication context between a reliable QUIC
      stream and a flow of unreliable QUIC datagrams. This can reduce
      the latency required for handshakes compared to opening both a TLS
      connection and a DTLS connection.

   * QUIC uses a more nuanced loss recovery mechanism than the DTLS
      handshake. This can allow loss recovery to occur more quickly for
      QUIC data.

   * QUIC datagrams are subject to QUIC congestion control. Providing
      a single congestion control for both reliable and unreliable data
      can be more effective and efficient.

   These features can be useful for optimizing audio/video streaming
   applications, gaming applications, and other real-time network
   applications.

   Unreliable QUIC datagrams can also be used to implement an IP packet
   tunnel over QUIC, such as for a Virtual Private Network (VPN).
   Internet-layer tunneling protocols generally require a reliable and
   authenticated handshake followed by unreliable secure transmission of
   IP packets. This can, for example, require a TLS connection for the
   control data and DTLS for tunneling IP packets. A single QUIC
   connection could support both parts with the use of unreliable
   datagrams in addition to reliable streams.

3. Transport Parameter

   Support for receiving the DATAGRAM frame types is advertised by means
   of a QUIC transport parameter (name=max_datagram_frame_size,
   value=0x20). The max_datagram_frame_size transport parameter is an
   integer value (represented as a variable-length integer) that
   represents the maximum size of a DATAGRAM frame (including the frame
   type, length, and payload) the endpoint is willing to receive, in
   bytes.

The default for this parameter is 0, which indicates that the
endpoint does not support DATAGRAM frames. A value greater than 0
indicates that the endpoint supports the DATAGRAM frame types and is
willing to receive such frames on this connection.

An endpoint MUST NOT send DATAGRAM frames until it has received the
max_datagram_frame_size transport parameter with a non-zero value
during the handshake (or during a previous handshake if 0-RTT is
used). An endpoint MUST NOT send DATAGRAM frames that are larger
than the max_datagram_frame_size value it has received from its peer.
An endpoint that receives a DATAGRAM frame when it has not indicated
support via the transport parameter MUST terminate the connection
with an error of type PROTOCOL_VIOLATION. Similarly, an endpoint
that receives a DATAGRAM frame that is larger than the value it sent
in its max_datagram_frame_size transport parameter MUST terminate the
connection with an error of type PROTOCOL_VIOLATION.

For most uses of DATAGRAM frames, it is RECOMMENDED to send a value
of 65535 in the max_datagram_frame_size transport parameter to
indicate that this endpoint will accept any DATAGRAM frame that fits
inside a QUIC packet.

The max_datagram_frame_size transport parameter is a unidirectional
limit and indication of support of DATAGRAM frames. Application
protocols that use DATAGRAM frames MAY choose to only negotiate and
use them in a single direction.

When clients use 0-RTT, they MAY store the value of the server's
max_datagram_frame_size transport parameter. Doing so allows the
client to send DATAGRAM frames in 0-RTT packets. When servers decide
to accept 0-RTT data, they MUST send a max_datagram_frame_size
transport parameter greater than or equal to the value they sent to
the client in the connection where they sent them the
NewSessionTicket message. If a client stores the value of the
max_datagram_frame_size transport parameter with their 0-RTT state,
they MUST validate that the new value of the max_datagram_frame_size
transport parameter sent by the server in the handshake is greater
than or equal to the stored value; if not, the client MUST terminate
the connection with error PROTOCOL_VIOLATION.

Application protocols that use datagrams MUST define how they react
to the absence of the max_datagram_frame_size transport parameter.
If datagram support is integral to the application, the application
protocol can fail the handshake if the max_datagram_frame_size
transport parameter is not present.

4. Datagram Frame Types

DATAGRAM frames are used to transmit application data in an
unreliable manner. The Type field in the DATAGRAM frame takes the
form 0b0011000X (or the values 0x30 and 0x31). The least significant
bit of the Type field in the DATAGRAM frame is the LEN bit (0x01),
which indicates whether there is a Length field present: if this bit
is set to 0, the Length field is absent and the Datagram Data field
extends to the end of the packet; if this bit is set to 1, the Length
field is present.

DATAGRAM frames are structured as follows:

DATAGRAM Frame {
  Type (i) = 0x30..0x31,
  [Length (i)],
  Datagram Data (..),
}

                   Figure 1: DATAGRAM Frame Format

DATAGRAM frames contain the following fields:

Length: A variable-length integer specifying the length of the
   Datagram Data field in bytes. This field is present only when the

LEN bit is set to 1. When the LEN bit is set to 0, the Datagram
Data field extends to the end of the QUIC packet. Note that empty
(i.e., zero-length) datagrams are allowed.

Datagram Data: The bytes of the datagram to be delivered.

5. Behavior and Usage

When an application sends a datagram over a QUIC connection, QUIC
will generate a new DATAGRAM frame and send it in the first available
packet. This frame SHOULD be sent as soon as possible (as determined
by factors like congestion control, described below) and MAY be
coalesced with other frames.

When a QUIC endpoint receives a valid DATAGRAM frame, it SHOULD
deliver the data to the application immediately, as long as it is
able to process the frame and can store the contents in memory.

Like STREAM frames, DATAGRAM frames contain application data and MUST
be protected with either 0-RTT or 1-RTT keys.

Note that while the max_datagram_frame_size transport parameter
places a limit on the maximum size of DATAGRAM frames, that limit can
be further reduced by the max_udp_payload_size transport parameter
and the Maximum Transmission Unit (MTU) of the path between
endpoints. DATAGRAM frames cannot be fragmented; therefore,
application protocols need to handle cases where the maximum datagram
size is limited by other factors.

5.1. Multiplexing Datagrams

DATAGRAM frames belong to a QUIC connection as a whole and are not
associated with any stream ID at the QUIC layer. However, it is
expected that applications will want to differentiate between
specific DATAGRAM frames by using identifiers, such as for logical
flows of datagrams or to distinguish between different kinds of
datagrams.

Defining the identifiers used to multiplex different kinds of
datagrams or flows of datagrams is the responsibility of the
application protocol running over QUIC. The application defines the
semantics of the Datagram Data field and how it is parsed.

If the application needs to support the coexistence of multiple flows
of datagrams, one recommended pattern is to use a variable-length
integer at the beginning of the Datagram Data field. This is a
simple approach that allows a large number of flows to be encoded
using minimal space.

QUIC implementations SHOULD present an API to applications to assign
relative priorities to DATAGRAM frames with respect to each other and
to QUIC streams.

5.2. Acknowledgement Handling

Although DATAGRAM frames are not retransmitted upon loss detection,
they are ack-eliciting ([RFC9002]). Receivers SHOULD support
delaying ACK frames (within the limits specified by max_ack_delay) in
response to receiving packets that only contain DATAGRAM frames,
since the sender takes no action if these packets are temporarily
unacknowledged. Receivers will continue to send ACK frames when
conditions indicate a packet might be lost, since the packet's
payload is unknown to the receiver, and when dictated by
max_ack_delay or other protocol components.

As with any ack-eliciting frame, when a sender suspects that a packet
containing only DATAGRAM frames has been lost, it sends probe packets
to elicit a faster acknowledgement as described in Section 6.2.4 of
[RFC9002].

If a sender detects that a packet containing a specific DATAGRAM

frame might have been lost, the implementation MAY notify the application that it believes the datagram was lost.

Similarly, if a packet containing a DATAGRAM frame is acknowledged, the implementation MAY notify the sender application that the datagram was successfully transmitted and received. Due to reordering, this can include a DATAGRAM frame that was thought to be lost but, at a later point, was received and acknowledged. It is important to note that acknowledgement of a DATAGRAM frame only indicates that the transport-layer handling on the receiver processed the frame and does not guarantee that the application on the receiver successfully processed the data. Thus, this signal cannot replace application-layer signals that indicate successful processing.

5.3. Flow Control

DATAGRAM frames do not provide any explicit flow control signaling and do not contribute to any per-flow or connection-wide data limit.

The risk associated with not providing flow control for DATAGRAM frames is that a receiver might not be able to commit the necessary resources to process the frames. For example, it might not be able to store the frame contents in memory. However, since DATAGRAM frames are inherently unreliable, they MAY be dropped by the receiver if the receiver cannot process them.

5.4. Congestion Control

DATAGRAM frames employ the QUIC connection's congestion controller. As a result, a connection might be unable to send a DATAGRAM frame generated by the application until the congestion controller allows it [RFC9002]. The sender MUST either delay sending the frame until the controller allows it or drop the frame without sending it (at which point it MAY notify the application). Implementations that use packet pacing (Section 7.7 of [RFC9002]) can also delay the sending of DATAGRAM frames to maintain consistent packet pacing.

Implementations can optionally support allowing the application to specify a sending expiration time beyond which a congestion-controlled DATAGRAM frame ought to be dropped without transmission.

6. Security Considerations

The DATAGRAM frame shares the same security properties as the rest of the data transmitted within a QUIC connection, and the security considerations of [RFC9000] apply accordingly. All application data transmitted with the DATAGRAM frame, like the STREAM frame, MUST be protected either by 0-RTT or 1-RTT keys.

Application protocols that allow DATAGRAM frames to be sent in 0-RTT require a profile that defines acceptable use of 0-RTT; see Section 5.6 of [RFC9001].

The use of DATAGRAM frames might be detectable by an adversary on path that is capable of dropping packets. Since DATAGRAM frames do not use transport-level retransmission, connections that use DATAGRAM frames might be distinguished from other connections due to their different response to packet loss.

7. IANA Considerations

7.1. QUIC Transport Parameter

This document registers a new value in the "QUIC Transport Parameters" registry maintained at <https://www.iana.org/assignments/quic>.

Value: 0x20
Parameter Name: max_datagram_frame_size
Status: permanent
Specification: RFC 9221

7.2. QUIC Frame Types

   This document registers two new values in the "QUIC Frame Types"
   registry maintained at <https://www.iana.org/assignments/quic>.

   Value: 0x30-0x31
   Frame Name: DATAGRAM
   Status: permanent
   Specification: RFC 9221

8. References

8.1. 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/info/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/info/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/info/rfc9000>.

   [RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure
              QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021,
              <https://www.rfc-editor.org/info/rfc9001>.

   [RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection
              and Congestion Control", RFC 9002, DOI 10.17487/RFC9002,
              May 2021, <https://www.rfc-editor.org/info/rfc9002>.

8.2. Informative References

   [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
              DOI 10.17487/RFC0768, August 1980,
              <https://www.rfc-editor.org/info/rfc768>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <https://www.rfc-editor.org/info/rfc6347>.

Acknowledgments

Authors' Addresses

   Tommy Pauly
   Apple Inc.
   One Apple Park Way
   Cupertino, CA 95014
   United States of America
   Email: tpauly@apple.com

   Eric Kinnear
   Apple Inc.
   One Apple Park Way
   Cupertino, CA 95014
   United States of America

Email: ekinnear@apple.com

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America
Email: dschinazi.ietf@gmail.com