

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 26, 2021

N. Kuhn
CNES
E. Stephan
Orange
G. Fairhurst
T. Jones
University of Aberdeen
February 22, 2021

Transport parameters for 0-RTT connections
draft-kuhn-quic-0rtt-bdp-08

Abstract

0-RTT mechanisms reduce the time it takes for the first bytes of application data to be processed in a transport connection and can greatly reduce connection latency during setup. The 0-RTT transport features described by quic-transport help clients establish secure connections with a minimal number of round-trips.

This document describes a generic method to exchange path parameters relating to transport. The additional transport parameters can help a connection that continues after an interruption or restarts by sharing connection properties. They can be used to increase the performance for a path with large RTT.

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 August 26, 2021.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Motivation	3
2.1. Optimizing client's requests	4
2.2. Safe jump start	4
2.3. Sharing transport information accross multiple connections	4
3. BDP metadata parameters	5
4. Extension activation	5
5. Discussion	6
5.1. Relevance of the transport parameter	6
5.2. The client point-of-view	6
5.3. BDP extension protected as Much as initial_max_data . . .	6
5.4. Congestion control safety	6
6. Acknowledgments	7
7. IANA Considerations	7
8. Security Considerations	7
9. Informative References	7
Appendix A. Example of server solution	8
Authors' Addresses	9

1. Introduction

Each transport connection typically starts without knowledge of the path between the endpoints. Transport protocols use implicit signals from the network to discover the properties of the path. This information is used to adapt the transport mechanisms to the network path. For example, an Internet transport endpoint is unable to determine a safe rate at which to start or continue their transmission, and uses slow-start to determine a safe rate. This applies to the 1-RTT mode of QUIC.

QUIC supports the sending of data in two different modes, after the transport handshake has completed, 1-RTT mode, and sending data along with handshake packets, 0-RTT mode. Using 0-RTT data an application is able to send transport parameters with the handshake packets, making it possible to reduce the latency of the connection setup.

In 0-RTT mode, a QUIC server must store a copy of a number of flow control related transport parameters, or receives an integrity-protected copy of these values in the ticket the client includes in the first message of the handshake, to enable the use of 0-RTT data. The setting or omission of one of these parameters can result in QUIC creating a connection, but flow control can still prevent any data being sent by the client.

For 0-RTT data to be sent, the QUIC server must record the values of:

- o initial_max_data
- o initial_max_stream_data_bidi_local
- o initial_max_stream_data_bidi_remote
- o initial_max_stream_data_uni
- o initial_max_streams_bidi
- o initial_max_streams_uni

These values set the flow control limits within which a connection must operate. The server has to store these parameters for a client to send data when resuming during 0-RTT. The stored values are used for any data that is transmitted before the handshake has completed and 1-RTT data is able to be sent on the connection. Once the handshake has completed, these values are discarded and the values established during the handshake are used.

This document proposes an extension to the transport parameters that are shared during the 0-RTT phase to allow resumption using additional transport and connection properties that were discovered in previous connections.

2. Motivation

Reducing the number of round-trips required to start a connection is an important way to reduce setup time and lower overall connection latency. 0-RTT mechanisms that allow a client to feed requests to a server in the first RTT do not alone improve the total time-to-service. The BDP extension described in this document aims to

improve traffic delivery by allowing the connection to short-cut slow RTT-based processes that grow connection parameters.

Currently each side has a proprietary solution to measure and to store path characteristics. Recalling the parameters of a previous would allow the use-cases presented in this section.

2.1. Optimizing client's requests

In cases with Dynamic Adaptive Streaming over HTTPS (DASH), clients may encounter issues in knowing the available bandwidth or DASH can encounter issues in reaching the best available video playback quality. The client's requests could be adapted and specific traffic could use information from the paths characteristics (such as trying to impress the client with high quality videos, to fill the buffers and avoid video blocking or to send high quality adds).

In other cases, applications may provide additionnal services if clients can know the server's estimation of the path characteristics.

2.2. Safe jump start

The server could use information from the paths characteristics to adapt to non-default path characteristics. Moreover, some transport parameters ma not need to be re-estimated (i.e. minRTT, MTU, bottleneck capacity, etc.).

CDNs currently exploit a very high Initial Window [TMA18]. Using the knowledge of previous path characteristics, CDN could:

- o adapt these values to save resource at the server and increase safely the initial congestion window such as proposed in [I-D.irtf-iccrs-sallantin-initial-spreading][CONEXT15];
- o consider client's limitation if the client decided to reject the extension.

2.3. Sharing transport information accross multiple connections

There is an interest in sharing transport information across multiple connections. [I-D.ietf-tcpm-2140bis] considers the sharing of transport parameters between connections originating from the same host. The proposal in this document have the advantage of storing the information at the client and not requiring the server to retain additional state for each client.

3. BDP metadata parameters

Section 7.3.1 of [I-D.ietf-quic-transport] describes the parameters that must be remembered if a client wishes to send 0-RTT data. Both endpoints store the value of the server transport parameters from a previous connection and apply them to any 0-RTT packets that are sent in subsequent connections to the same peer. Of the six mandatory parameters, only `initial_max_data` improves the time-to-service of the 0-RTT connection. The BDP metadata extension augments the list of server transport parameters that are shared with the client to improve the time-to-service and save resources such as CPU, memory and power.

The BDP extension proposes two new parameters

- o `recon_bytes_in_flight` (0x000X): The bytes in flight measured on the previous connection by the server. Integer number of bytes. Using the `bytes_in_flight` defined in [I-D.ietf-quic-recovery], `recon_bytes_in_flight` can be set to `bytes_in_flight`.
- o `recon_min_rtt` (0x000X): The minimum RTT measured on the previous connection by the server. Integer number of milliseconds. Using the `min_rtt` defined in [I-D.ietf-quic-recovery], `recon_min_rtt` can be set to `min_rtt`. The `min_rtt` parameter may not track a decreasing RTT: the `min_rtt` that is reported here may not be the actual minimum RTT measured during the 1-RTT connection, but still reflects the characteristics of the latency on the network.

4. Extension activation

The BDP extension is protected by the same mechanism that protects the exchange of the 0-RTT transport parameters. A client that activates 0-RTT data sends back the transport parameters received from the server during the previous connection (see Section 7.3.1 of [I-D.ietf-quic-transport]).

The client reads the parameters in the BDP metadata extension, but can not change them.

Accept: A client MAY use the extension parameters. Then, it activates ingress optimization and sends back the transport parameters of the BDP metadata extension that it received from the server during the previous connection.

Refuse: A client could choose not to use there parameters. Then, it does not support ingress optimisation and drops the extension signal. A client that disagrees with the extension parameters received from the server refuses the optimization.

5. Discussion

5.1. Relevance of the transport parameter

The `recon_bytes_in_flight` parameter is higher than the number of bytes in the actual BDP since it may include bytes in buffers along the path. That being said, the `recon_bytes_in_flight` may be lower than the actual value at the end of a connection since there may be a low amount of data to send to terminate the transmission.

5.2. The client point-of-view

The client can read the values of the extension. The client may want to reject the extension, to accept and adapt the resource and flow control parameters or adapt requests. The client cannot change the values of the extension.

5.3. BDP extension protected as Much as `initial_max_data`

The BDP metadata parameters are measured by the server during a previous connection. The BDP extension is protected by the mechanism that protects the exchange of the 0-RTT transport parameters. For the version 1 of QUIC, the BDP extension is protected using the mechanism that already protects the `"initial_max_data"` parameter. This is defined in sections 4.5 to 4.7 of [I-D.ietf-quic-tls]. It provides the server with a way to check the parameters proposed by the client are those that the server sent to the client during the previous connection.

5.4. Congestion control safety

The maximum number of initial data packets that can be sent without acknowledgment needs to be chosen to avoid congestion collapse. A mechanism is needed to ensure that the information sent by the server is relevant and that network conditions have not changed. The client cannot change the values of the extension.

The initial window is considered a safe starting point for an unknown path to avoid adding congestion to a congested network. If the reception of IW is confirmed for the first RTT of data, and also the path is determined to be similar to that of a recent previous session (e.g., similar RTT), the method permits the sender to use the previous path information as an input to help determine a new safe rate. Another safety net could be the pacing of packets as a function of the estimated RTT. This follows the ideas of [I-D.cardwell-iccr-g-bbr-congestion-control] and [RFC4782].

6. Acknowledgments

The authors would like to thank Gabriel Montenegro, Patrick McManus, Ian Swett, Igor Lubashev, Christian Huitema and Tom Jones for their fruitful comments on earlier versions of this document.

7. IANA Considerations

TBD: Text is required to register the extension BDP_metadata field. Parameters are registered using the procedure defined in [I-D.ietf-quic-transport].

8. Security Considerations

The BDP metadata parameters are measured by the server during a previous connection.

The BDP extension is protected by the mechanism that protects the exchange of the 0-RTT transport parameters. For the version 1 of QUIC, the BDP extension is protected using the mechanism that already protects the "initial_max_data" parameter. This is defined in sections 4.5 to 4.7 of [I-D.ietf-quic-tls]. It provides the server with a way to check the parameters proposed by the client are those that the server sent to the client during the previous connexion.

9. Informative References

[CONEXT15]

Li, Q., Dong, M., and P. Godfrey, "Halfback: Running Short Flows Quickly and Safely", ACM CoNEXT , 2015.

[I-D.cardwell-iccr-g-bbr-congestion-control]

Cardwell, N., Cheng, Y., Yeganeh, S., and V. Jacobson, "BBR Congestion Control", draft-cardwell-iccr-g-bbr-congestion-control-00 (work in progress), July 2017.

[I-D.ietf-quic-recovery]

Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-34 (work in progress), January 2021.

[I-D.ietf-quic-tls]

Thomson, M. and S. Turner, "Using TLS to Secure QUIC", draft-ietf-quic-tls-34 (work in progress), January 2021.

[I-D.ietf-quic-transport]

Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-34 (work in progress), January 2021.

[I-D.ietf-tcpm-2140bis]

Touch, J., Welzl, M., and S. Islam, "TCP Control Block Interdependence", draft-ietf-tcpm-2140bis-07 (work in progress), December 2020.

[I-D.irtf-iccr-g-sallantin-initial-spreading]

Sallantin, R., Baudoin, C., Arnal, F., Dubois, E., Chaput, E., and A. Beylot, "Safe increase of the TCP's Initial Window Using Initial Spreading", draft-irtf-iccr-g-sallantin-initial-spreading-00 (work in progress), January 2014.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.

[RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, DOI 10.17487/RFC4782, January 2007, <<https://www.rfc-editor.org/info/rfc4782>>.

[TMA18] Ruth, J. and O. Hohlfeld, "Demystifying TCP Initial Window Configurations of Content Distribution Networks", 2018 Network Traffic Measurement and Analysis Conference (TMA) , 2018.

Appendix A. Example of server solution

This section details a solution at the server to safely increase the maximum amount of packets that the server sends when receiving a 0-RTT packet from a client.

The initial window is considered a safe starting point for an unknown path to avoid adding congestion to a congested network. The general assumption is that a path does not currently suffer persistent congestion, and therefore the initial window is applicable until feedback about the path is received. The resulting initial sending rate is only tentative until the capacity is confirmed to be available. If there is loss within this initial transmission, then this could be evidence that the path is congested, and the sender needs to adjust to this congestion.

Significant loss could be an indication of congestion collapse - i.e. persistent loss, requiring back-off of the sending rate.

If however, the reception of IW is confirmed for the first RTT of data, and also the path is determined to be similar to that of a recent previous session (e.g., similar RTT), the method permits the sender to use the previous path information as an input to help determine a new safe rate. One possibility is to immediately jump to a new sending rate that is derived from the previously sustained rate. This follows the ideas of [I-D.cardwell-iccrg-bbr-congestion-control] and [RFC4782].

The QoS mechanisms that are deployed in the networks can help in prevent the congestion collapse from occurring. However, the sender must provide a significant reduction if there is evidence of potential congestion collapse [RFC2914] from his point of view. Precautions can be taken to guarantee that it is reasonably safe to jump to a high sending rate : measuring that network conditions did not change, allowing some space for other flows that have started and pace transmission of packets. The sender might need to rapidly reduce its rate, if the higher sending rate does not prove to be supported. If it is supported, the sender can resume standard congestion control.

Authors' Addresses

Nicolas Kuhn
CNES

Email: nicolas.kuhn@cnes.fr

Emile Stephan
Orange

Email: emile.stephan@orange.com

Gorry Fairhurst
University of Aberdeen

Email: gorry@erg.abdn.ac.uk

Tom Jones
University of Aberdeen

Email: tom@erg.abdn.ac.uk