QUIC
Internet-Draft
Intended status: Standards Track
Expires: 18 November 2022

Y. Ma
Y. Liu
Alibaba Inc.
C. Huitema
Private Octopus Inc.
X. Yu
Alibaba Inc.
17 May 2022

An Advanced Scheduling Option for Multipath QUIC
draft-ma-quic-mpqoe-00

Abstract

This document specifies an advanced scheduling option for multipath QUIC protocol. The goal is to enable the use of multipath QUIC for applications that have tight latency constraints. For general purpose multipath packet scheduling, please refer to [I-D.bonaventure-iccrg-schedulers].

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/yfmascgy/draft-ma-quic-advance-scheduling.

Status of This Memo

Table of Contents

1. Introduction

Multi-path QUIC transport, which allows the simultaneous usage of multiple paths for a single QUIC connection, has recently gained attention [QUIC-MULTIPATH]. In practice, however, it turns out to be not straightforward to apply multipath QUIC to applications that have tight latency constraints (e.g., video streaming and gaming) with only basic scheduling options [I-D.bonaventure-iccrg-schedulers]. In this draft, we introduce an advanced scheduling option for the usage of multi-path QUIC in latency-constraint applications.

The proposed scheduling option in this draft includes two major components: (1) parallel transmission of duplicate data on several paths, and (2) a feedback mechanism to make the scheduling strategy adaptive based on the state of the application or the state of the network.

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.

We assume that the reader is familiar with the terminology used in [QUIC-TRANSPORT].

3. Packet scheduling

3.1. General-purpose Packet Scheduling

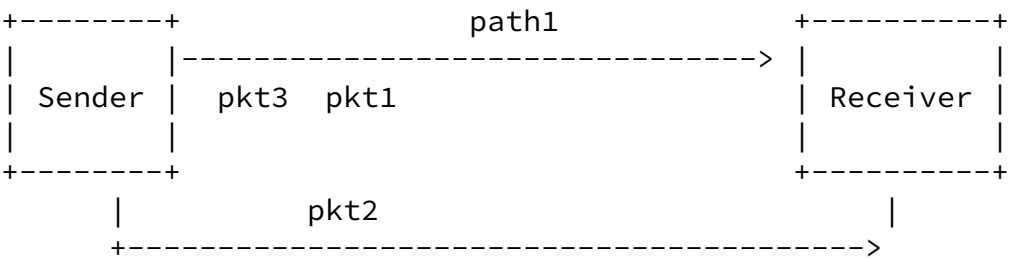
Experience with multipath transport protocols shows that the packet scheduler can have a huge impact on the transport performance. In general-purpose multipath scheduling strategies [I-D.bonaventure-iccrg-schedulers], whether it is round-robin, strict

priority, or lowest round-trip-time, we often face a dilemma: On one hand, in order to aggregate bandwidth, we need to split traffic across multiple paths, but in doing so, the overall latency is hurt by the slower path as faster paths have to wait for packets scheduled on it to be received. On the other hand, if we choose to pick only one path to use at a time, then we lose not only the benefit of bandwidth aggregation, but also the reliability of using multipath as the path condition of a wireless link can vary quickly. A fundamental problem with multipath scheduling is the head-of-line blocking when paths have large delay difference. Deployment experience shows that multi-path HoL blocking has negative impact on the quality of experience of applications that have tight latency

constraints, such as video streaming [XLINK].

3.2. Head-of-line blocking issues in multi-path scheduling

The head-of-line blocking happens when a scheduler splits an application's traffic across multiple paths, on one of which, the transmitted packets take significantly longer time to deliver due to either large path delay or high packet loss rate on that path. As shown in Figure 1, at t=t1, a sender transmits a media chunk that consists of three packets (pkt1, pkt2, and pkt3) with two paths (path1 and path2), where path2 has much longer delay than path1. Due to the limit of path1's congestion window, after sending pkt1 on path1, the sender has to switch to path2 for transmitting pkt2. When the congestion window of path1 becomes available later, the sender transmits pkt3 on it. At t=t2, pk1 and pkt3 on path1 are received by the receiver, but pkt2 on path2 is still in flight even though it is sent before pkt3, causing out-of-order delivery. The out-of-order packet, pkt3, is not eligible to be submitted to the application and the client on the receiver-side has to wait for pkt2 to process the entire media chunk. In other words, the data transmission from the application's point of view is blocked by pkt2.



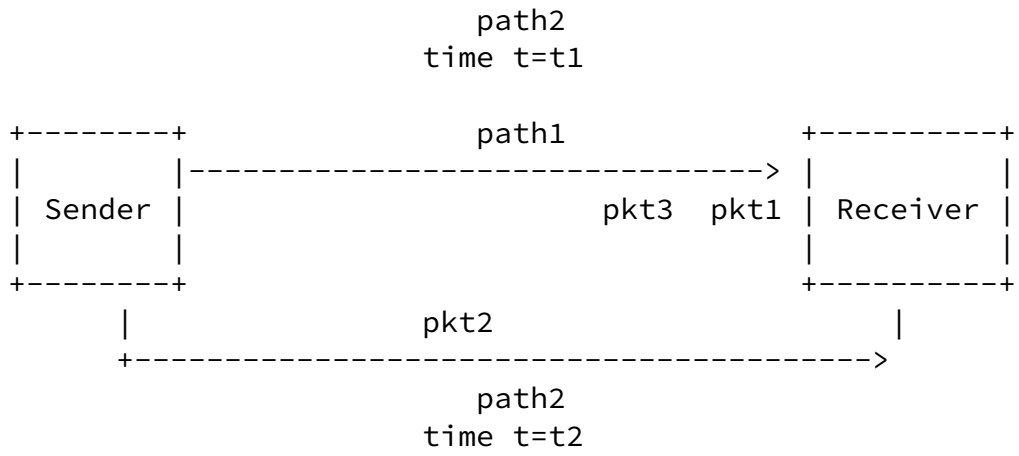


Figure 1: Head-of-line blocking in multipath

4. Proposed advanced scheduling option

To cope with HoL blocking, we propose an advanced scheduling option, which includes two major components: (1) parallel transmission of duplicate data on several paths, and (2) a feedback mechanism to make the scheduling strategy adaptive based on the state of the application or the state of the network.

4.1. Parallel transmission of duplicate data on several paths

The main solution to overcome HoL blocking is to allow concurrent transmission of packets that contain duplicate copies of data on multiple paths. In doing so, we can avoid the excessive delay when a packet becomes stagnant on a path that has large delay or high loss rate becuase a copy of it on another path could arrive instead. Such a transmission mode can be put into two categories: (1) full-redundancy mode and (2) re-injection mode.

4.1.1. Full-redundancy transmission

In full-redundancy mode, a scheduler sends duplicate copies of data on every path that has available congestion window. As shown in

Figure 2, pkt1, pkt2, and pkt3 contain original copies of a media chunk, while pkt1', pkt2', and pkt3' contain the corresponding duplicate copies. In multipath QUIC [QUIC-MULTIPATH], pktN and pktN' are either in different packet number spaces or in the same packet number space but assigned with different packet numbers. The media data carried in those packets should be in the same QUIC stream so that only a single copy of data is delivered to the receiver. The full redundancy mode takes the advantage of path diversity to obtain lowest possible latency. However, it does not aggregate bandwidth of multiple paths. When the required data rate is larger than the minimum bandwidth of paths, such a full-redundancy mode is not recommended.

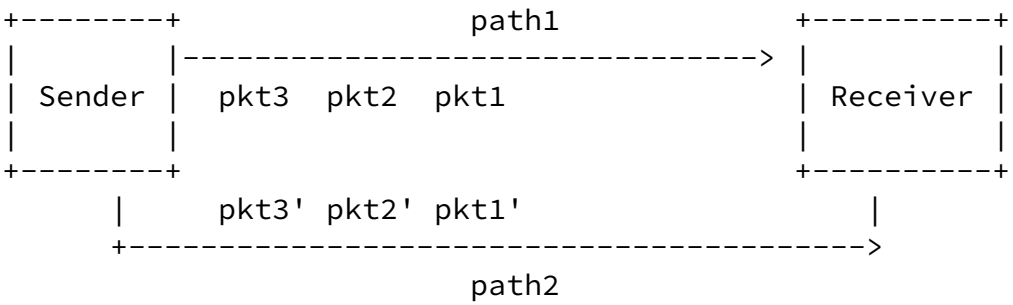


Figure 2: Full-redundancy transmission mode

4.1.2. Re-injection mode

In re-injection mode, a scheduler sends duplicated content of unacknowledged packets from one path into another one without waiting for the loss recovery on the original path. For example, a scheduler decides to re-send the content of a packet on another path if the ACK of it is not received after a certain time threshold. In another example, when a sender finishes sending packets carrying the content of a video chunk, it immediately starts re-sending those previously sent copies on a different path before moving on to send the next video chunk. Re-injection mode is illustrated in Figure 3, where path2 has a much larger delay than path1. After a certain time threshold, the sender detects that the ACK of pkt2 is not received, to avoid excessive waiting by the receiver, it re-injects pkt2' on the faster path even before the loss recovery kicks in on the slower

path. Similar to what is discussed above, in multipath QUIC [QUIC-MULTIPATH], pktN and the re-injected pktN' are either in different packet number spaces or in the same packet number space but assigned with different packet numbers. The media data carried in those packets should be in the same QUIC stream so that only a single copy of data is eventually delivered to the receiver. Re-injection mode strikes a balance between transmission latency and aggregated bandwidth. It is also flexible to use as one can tune the parameters such as the time threshold to optimize for various applications.

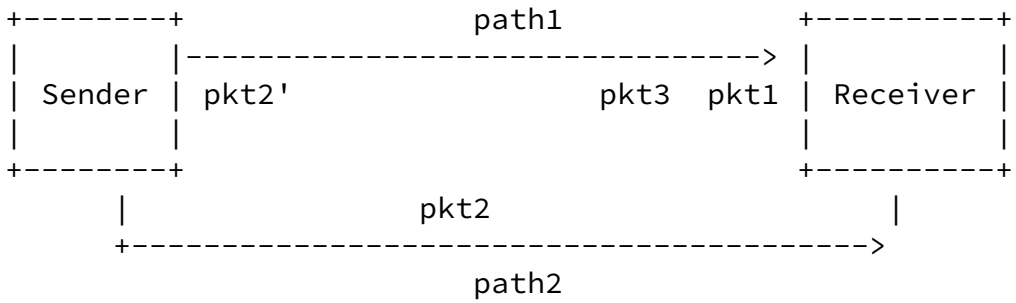


Figure 3: Re-injection transmission mode

4.2. QoE Feedback

The second major component is a QoE feedback mechanism that enables a sender to adapt its scheduling strategy. On one hand, applications may have different QoE requirements---the interactive applications are delay sensitive, while the video streaming applications are more throughput sensitive. There is thus a trend of cross-layer design that takes applications' demands into account when managing paths or scheduling packets. On the other hand, the network conditions (e.g., bandwidth, loss rate, and latency), as well as the application states (e.g., video bitrate, video buffer level) are constantly changing, so it is desired to have a scheduling strategy that is adaptive to those

conditions.

The QoE feedback is used to fully support adaptive multipath scheduling and is carried in the QOE_CONTROL_SIGNALS frames Figure 4. A sender can adjust its scheduling strategy based on the received QoE feedback. The QOE_CONTROL_SIGNALS frames can include two types of information that is needed by the scheduler: (1) application-level information and (2) the network-level information. The frequency of

such feedback should be controlled to limit the amount of extra packets. The QoE control signal allows a synchronization of viewpoints between two endhosts. The network-level information can include interface types and interface priorities. For example, a client on the cellphone can inform a server at this moment if a wifi interface is more preferred than a 5G interface. The application-level information can include video bitrate, video framerate, video buffer level, etc., which can inform the server how likely a future rebuffering event might happen. It is up to the application to determine the interpretation of QoE control signals.

5. Combination of the two components.

The two components can be used independently, but are recommended to work hand in hand. Parrallel transmission of duplicate data enables quicker recovery from out-of-order delivery. However, the downside of such a strategy is the additional traffic cost when it is aggressively used. One example is to control traffic redundancy when packet re-injection is implemented to improve multi-path transport performance [XLINK]. As discussed above, the problem with packet re-injection is that it MAY introduce a lot of redundant packets, increasing traffic cost. Indeed, redundant packets are not always needed as the video player MAY cache video chunks. Therefore, if the number of cached frames is large in the video player, the play-time left until the next possible re-buffering is long, and hence, the urgency of using re-injection is low. On the contrary, if the number of cached frames is small in the video player, the time left until the next possible re-buffering is short and, hence, the urgency of using re-injection is high. Knowing that the client video player's buffer occupancy level is an indicator of the user-perceived QoE, one can capture the related information (such as number of cached frames and framerate) in client, encapsulate the information in QoE_CONTROL_SIGNAL and send it back to the server to decide when to turn on or turn off its re-injection usage.

6. New frames

All the new frames MUST be sent in 1-RTT packet, and MUST NOT use other encryption levels.

If an endpoint receives MP frames from packets of other encryption

levels, it MUST return MP_PROTOCOL_VIOLATION as a connection error and close the connection.

6.1. QOE_CONTROL_SIGNALS frame

QOE_CONTROL_SIGNALS frame is used to carry quality of experience (QoE) information. A typical use of such information is to provide feedback to help application-aware scheduling. Note that different applications may have very different needs, the interpretation of the QoE control signal can be up to the users. QOE_CONTROL_SIGNALS frames are formatted as shown in Figure 4.

```
QOE_CONTROL_SIGNALS Frame {
  Type (i) = TBD-02 (experiments use 0xbaba02),
  Path Identifier (..),
  QoE Control Signals Length(8),
  QoE Control Signals (..)
}
```

Figure 4: QOE_CONTROL_SIGNALS Frame Format

Path Identifier: An identifier of the path, which is defined in [QUIC-MULTIPATH].

QOE_CONTROL_SIGNALS frames may be received out of order, peers SHOULD pass them to the application as they arrive. Although QOE_CONTROL_SIGNALS frames are not retransmitted upon loss detection, they are ack-eliciting [QUIC-RECOVERY].

7. Implementation Considerations

TBD.

8. Security Considerations

TBD.

9. IANA Considerations

TBD.

10. Contributor

This document is a collaboration of authors that combines works from several proposals. Further contributors that were also involved are: * Dapeng Liu * Juan Deng

11. Changelog

12. Appendix.A Difference from past proposals

TBD.

13. References

13.1. Normative References

[QUIC-MULTIPATH]
Liu, Y., Ma, Y., Coninck, Q., Bonaventure, Q., Huitema, C., and M. Kuehlewind, "Multipath Extension for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-multipath, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath>.

[QUIC-RECOVERY]
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/rfc/rfc9002>.

[QUIC-TLS] 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/rfc/rfc9001>.

[QUIC-TRANSPORT]
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>.

[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>.

13.2. Informative References

[I-D.bonaventure-iccrg-schedulers]
Bonaventure, O., Piraux, M., Coninck, Q. D., Baerts, M., Paasch, C., and M. Amend, "Multipath schedulers", Work in Progress, Internet-Draft, draft-bonaventure-iccrg-schedulers-02, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-bonaventure-iccrg-schedulers-02>.

[XLINK] Zheng, Z., Ma, Y., Liu, Y., Yang, F., Li, Z., Zhang, Y., Shi, W., Chen, W., Li, D., An, Q., Hong, H., Liu, H., and M. Zhang, "XLINK: QoE-driven multi-path QUIC transport in large-scale video services", August 2021, <https://dl.acm.org/doi/10.1145/3452296.3472893>.

Authors' Addresses

Yunfei Ma
Alibaba Inc.
Email: yunfei.ma@alibaba-inc.com

Yanmei Liu
Alibaba Inc.
Email: miaoji.lym@alibaba-inc.com

Christian Huitema
Private Octopus Inc.
Email: huitema@huitema.net

Xiaobo Yu
Alibaba Inc.
Email: shibo.yxb@alibaba-inc.com