Media Over QUIC
Internet-Draft
Intended status: Experimental
Expires: 12 May 2026

B. Wu
Tencent
T. Li
Renmin University of China
C. Luo
J. Lou
Tencent
K. Xu
Tsinghua University

8 November 2025

Enhanced QUIC Recovery for Video Streaming
draft-wu-moq-recovery-for-video-streaming-00

Abstract

The current loss recovery in QUIC provides reliable delivery but is not optimized for latency, potentially hindering time-sensitive applications. The document analyzes the issues caused by the current QUIC recovery scheme during live video streaming and then proposes an enhanced QUIC recovery mechanism that leverages additional loss retransmissions for optimizing client-side QoE. At a high level, the QUIC sender performs additional recovery logic when transport runs into application limited state, which is for the lost data that has been retransmitted before but has not been acknowledged yet. This mechanism can act as a supplement to the existing recovery scheme that performs immediate loss retransmissions once a packet is detected lost.

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 12 May 2026.

Table of Contents

1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Problem Statement

QUIC-powered traffic transmission introduces an obvious optimization for the performance of live video streaming. The existing design logic of QUIC recovery enables to perform data retransmission once a packet is detected lost while ignoring the quality of loss recovery, which can not well adapt to the timeliness requirement of live video services. Specially, if a piece of data is detected lost more than once, e.g., due to network congestions, the QUIC client has to keep waiting until the retransmitted lost data is successfully received. During this time, the follow-up data of this video stream can not be uploaded to the player buffer even though the related packets have already arrived at the QUIC client. In this case, the client-side QoE can be further deteriorated (e.g., video freezing occurs) when the data in application queue is consumed. The root cause of the above issue is the existing loss recovery scheme actually provide few optimizations for retransmission quality so that the lost data might not be received by clients in time.

3. Overview of Standards on QUIC Loss Recovery

QUIC is an UDP-based reliable transport protocol, which can provide secure and quick traffic transmissions for general-purpose Internet services [RFC9000]. Similar to common TCP designs, QUIC senders should also perform loss detections and recovery from loss of data while taking appropriate congestion control action. Once a packet is detected lost, a QUIC sender requires to immediately recover from that lost in another packet with a new and increased packet number, which can be used to accurately distinguish whether the potential lost data is caused by transmitted or retransmitted packets.

The exist loss recovery scheme of QUIC protocol is detailed in [RFC9002], in which acknowledgments can be used to detect lost packets and a PTO (probe timeout) will be employed to ensure acknowledgments are received. Besides, the spirits of TCP’s Fast Retransmit [RFC5681], Early Retransmit [RFC5827], Forward Acknowledgement [FACK], SACK loss recovery [RFC6675] and RACK-TLP [RFC8985] are also implemented during QUIC recovery for the lost data.

Even though the immediate loss recovery can reduce the waiting time of receivers for the lost data, the retransmitted packets might fail to reach the clients in time due to changeable network status. For example, if the resent data is lost again during its retransmission, the QUIC receiver has to wait for at least another 1 RTT and might suffer from the deteriorated QoE such as video freezing. This is because the follow-up data of this stream can not be delivered to the application layer while the remaining data previously cached in

player buffer is actually continuously consumed. Once the remaining data is exhausted after a long waiting time, deteriorated QoE will occur on the QUIC receiver side. The current QUIC recovery scheme lacks further optimizations for the quality of loss recovery, which might introduce a long waiting time for the lost data, especially when the retransmitted packet is lost again. This document mainly focuses on designing an enhanced loss recovery mechanism to reduce the receiver-side waiting time for the lost data, which can be leveraged to optimize the timeliness of live video streaming.

4. Optimized Loss Recovery for Live Streaming

4.1. Requirements

4.1.1. Endpoint and Network Changes

This mechanism only requires to make some changes on QUIC sender side that should performs additional loss recovery when application limited. There is no need to make some modifications for QUIC receivers that still obeys the logic declared in common QUIC designs.

The existing network equipments (e.g., routers and switches) and the format of QUIC packets do not require any changes to match the enhanced loss recovery proposed in this document.

4.1.2. Compatibility and Suitability

This mechanism is compatible with the existing loss detection and recovery scheme, in which supplemental loss recoveries are employed to optimize the receiver-side waiting time for the lost data.

This mechanism is suitable for optimizing the receiver-side QoE of live video streaming that has the following characteristics:

1. Timeliness requirement: the live video services have more strict requirements for the timeliness of traffic data, in which the lost data should be successfully transmitted to its receiver within a period of time. Otherwise, the deteriorated QoE such as video freezing will occur on the receiver side.

2. Application limitation: the live video streaming often makes senders become application-limited, i.e., there is no data to send. This is because the data of live video is encoded with a configured frame rate, which actually introduces a certain time interval between adjacent frames. When a frame has been sent and next frame has not yet been generated, the sender will become application-limited.

This mechanism supports the existing loss recovery scheme that relies on retransmitting lost data, sending an updated frame or discarding the frame, see Section 13.3 of [RFC9000].

4.2. Application Limit Identification

During live video streaming, QUIC senders will frequently become application-limited, in which no data can be obtained temporarily for continuous transmissions. This is because the time interval (T_interval) between the generations of two adjacent frames is ubiquitous, which is introduced by the configured frame rate (F_rate) for frame encoding:

T_interval = 1 / F_rate

From the perspective of senders, T_interval reflects the sending time interval between two adjacent frames.

The cost time (T_cost) that a QUIC sender takes to send the current frame and recovers from previous loss can be calculated using the valide transport speed (S_trans) and the size of each encoded frame (S_frame):

T_cost = S_frame / S_trans

The QUIC sender will become application-limited if the following condition is satisfied:

T_cost < T_interval

Note that S_trans is bounded by the current maximum available bandwidth (max_bw) and S_frame can be obtained from video bitrate (B_video) and F_rate, as follows:

S_frame = B_video / F_rate

Therefore, the following condition can also cause QUIC senders to be application-limited:

B_video < S_trans

For example, 60-fps F_rate will introduce a new video frame every 16.7 ms (i.e., T_interval = 16.7 ms) on average, where the value of B_video is assumed as 2 Mbps. If S_trans is larger than 2 Mbps, QUIC senders will frequently step to the application-limited status, in which they only keep waiting for the next newly encoded frame.

In this document, QUIC sender can identify whether it has become application-limited, which has already been supported in the existing QUIC solutions, such as [RFC9002] and [RATE_ESTIMATION].

4.3. Unacked Lost Packet Queue

In this mechanism, a QUIC sender requires to record the lost data for each connection, which has already been retransmitted but not been acknowledged by its receiver. To achieve the above function, an additional data queue (called Queue_unack_loss) should be established and maintained by QUIC senders. The data in Queue_unack_loss will be leveraged to enhance the performance of QUIC recovery and optimize receiver-side QoE of timeliness-sensitive live streaming.

4.3.1. Queue_unack_loss Establishment and Release

The Queue_unack_loss for each QUIC connection will be established and released by QUIC senders if it meets one of the following conditions:

1. A QUIC sender will establish Queue_unack_loss when a QUIC connection is created, and release this queue after closing this connection.

2. A QUIC sender will establish Queue_unack_loss when any packet loss is detected and no Queue_unack_loss for this QUIC connection has already been established at the current moment. If there is no lost data that has not been acknowledged by QUIC receiver (i.e., Queue_unack_loss is empty), the Queue_unack_loss created previously will be released by the QUIC sender.

4.3.2. Queue_unack_loss Maintenance

The Queue_unack_loss for a QUIC connection will be maintained on the QUIC-sender side if any lost data is newly detected or any loss retransmission is acknowledged. Overall, the data in Queue_unack_loss should be updated when it meets one of the following conditions:

1. If a packet is detected lost and the packet data has not existed in Queue_unack_loss, the resent data will be added to the end of Queue_unack_loss after its retransmission.

2. If a packet is detected lost and the packet data has already existed in Queue_unack_loss, the resent data will be added to the end of Queue_unack_loss after its retransmission. At the same time, this lost data of this packet will be deleted from Queue_unack_loss.

3. If the lost data recorded in Queue_unack_loss is retransmitted by QUIC sender, it will be moved (from the middle or the head) to the end of Queue_unack_loss. Note that the data location changes in Queue_unack_loss can be achieved by deleting and adding operations.

4. If the lost data is successfully recovered (i.e., acknowledged by its QUIC receiver), it will be deleted from Queue_unack_loss that is operated by its QUIC sender.

As Figure 1 shows, Queue_unack_loss records four lost data (i.e., Loss 0 ~ 3) with the location of 0 ~ 3. When Loss 0 is retransmitted again, the QUIC sender will remove it from location 1 to 4. At the same time, if a retransmitted packet is acknowledged by receiver, the data in this packet will be deleted from Queue_unack_loss.

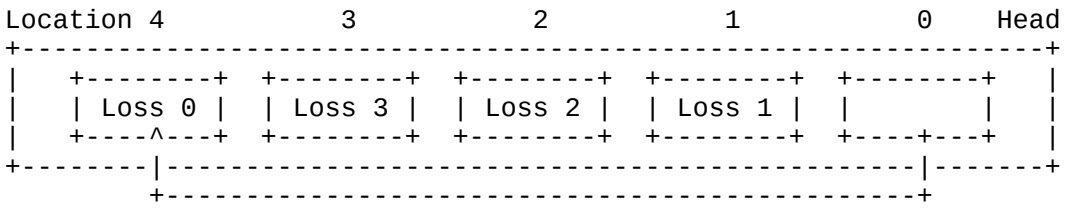

Figure 1: Queue_unack_loss update

In this mechanism, any two packets (e.g., packet A and B) are strongly correlated if packet B acts as an recovery packet for the loss of packet A. In QUIC protocol, the data of packet B might differ from the data of packet A, as their sender could send an updated frame packeted in packet B for recovering the loss of packet A.

The key rules for Queue_unack_loss maintenance can be described as follows:

1. Queue_unack_loss only records the resent but unacknowledged data.

2. The data in Queue_unack_loss is different from each other, whose corresponding packets can not be strongly correlated.

3. The data in Queue_unack_loss is sorted based on the time of their most recent retransmissions, in which the latest resent data is at the end of this queue.

4.4. Sender-side Operations

4.4.1. Additional Loss Recovery

In this mechanism, the existing QUIC-based traffic transmission [RFC9000] and loss recovery scheme [RFC9002] will be performed as usual, in which the lost data will be retransmitted immediately once it is detected lost. Meanwhile, QUIC senders continuously maintain the status of Queue_unack_loss and keep identifying whether it becomes application-limited during live video streaming in each connection. If application-limited, a QUIC sender will actively retransmit the lost data in the order of data locations in Queue_unack_loss. Specially, the lost data at the head of Queue_unack_loss will be fetched to perform an additional loss recovery and then be moved to the end of Queue_unack_loss. When not application limited (app_unlimited), the QUIC sender will employ the existing QUIC protocol to send (or resend) the next traffic (lost) data of live video services.

The workflow of this mechanism is shown in Figure 2.

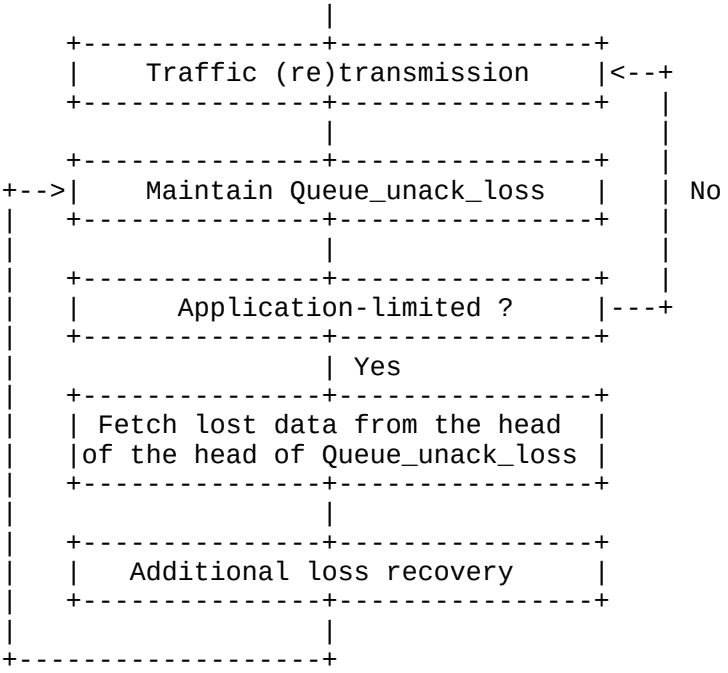

Figure 2: The Mechanism Workflow

4.4.2. No Lost Data

When application limited, QUIC senders firstly check whether Queue_unack_loss is established or whether any lost and unacknowledged data exists in this queue. If there is no data recorded in this queue or this queue has not been established, QUIC senders will not perfrom additional recoveries for data loss.

4.4.3. Additional Recovery Times

In this mechanism, the QUIC sender performs additional loss recovery at most once for each lost data every time this sender is application limited. In other words, the lost data recorded in Queue_unack_loss will be retransmitted only once or not be retransmitted between two adjacent timestamps when a QUIC sender starts and ends to become application-limited (app_limited). As Figure 3 shows, the QUIC sender keeps application-limited from t1 to t2. In this duration, the data can be taken from the head of Queue_unack_loss and then be used to perfrom further recovery from the loss. If all data has been retransmitted once during this time, QUIC sender will wait for another newly detected loss or being not application-limited. If there is remaining data in Queue_unack_loss that has not been retransmitted when not application-limited, these data will be leveraged to loss recovery when the sender becomes application-limited again.

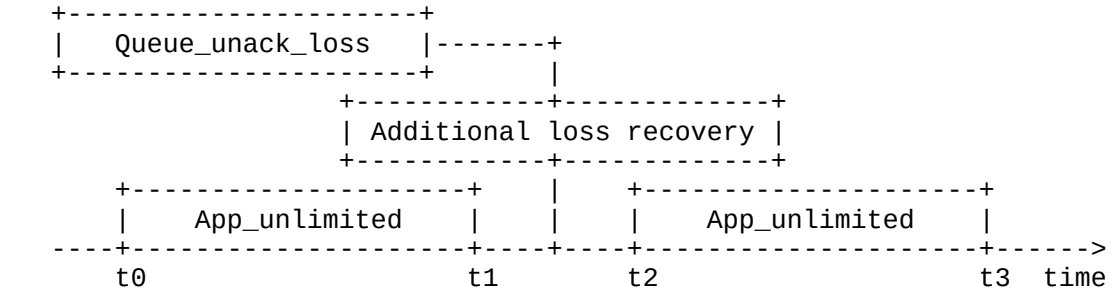

Figure 3: The Timing for Additional Loss Recovery

4.4.4. Flow Control and Congestion Control

The additional loss recovery complies with the flow control scheme in the existing QUIC protocol.

The additional loss recovery scheme employs the data in Queue_unack_loss for optimizing the timeliness of live video streaming. This could introduce worse network congestions on traffic forwarding path, which will lower the transmission performance of

next video streaming. In order to avoid the above issue, this mechanism enables additional loss recovery to obey the congestion control of the current QUIC connection. For example, the congestion window (cwnd) and pacing rate should not exceed the corresponding values when the QUIC sender begins to be application-limited.

5. Security Considerations

This proposal does not introduce any changes to the security of the existing QUIC solutions or congestion control scheme.

6. IANA Considerations

No request of IANA is made in this documents.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Acknowledgments

We would like to thank Fuyu Wang, Changkui Ouyang, and Xu Yan for their feedback and suggestions.

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>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC5827] Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J., and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)", RFC 5827, DOI 10.17487/RFC5827, May 2010, <https://www.rfc-editor.org/info/rfc5827>.

[RFC6675] Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., and Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP", RFC 6675, DOI 10.17487/RFC6675, August 2012, <https://www.rfc-editor.org/info/rfc6675>.

[RFC8985] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "The RACK-TLP Loss Detection Algorithm for TCP", RFC 8985, DOI 10.17487/RFC8985, February 2021, <https://www.rfc-editor.org/info/rfc8985>.

[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>.

[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

[FACK] Matthew, M., Ed. and J. Mahdavi, Ed., "Forward Acknowledgement: Refining TCP Congestion Control", ACM SIGCOMM Computer Communication Review, DOI 10.1145/248157.248181 https://doi.org/10.1145/248157.248181, August 1996.

[RATE_ESTIMATION] Cheng, Y., Ed., Cardwell, N., Ed., Hassas Yeganeh, S., Ed., and V. Jacobson, Ed., "Delivery Rate Estimation", Work in Progress, Internet-Draft, draft-cheng-iccrg-delivery-rate-estimation-02, 8 September 2022, <https://datatracker.ietf.org/doc/html/draft-cheng-iccrg-delivery-rate-estimation-02>.

Authors' Addresses

Bo Wu
Tencent
Tencent Headquarters Building
Haidian District
Beijing
China
Email: brynwu@tencent.com

Tong Li
Renmin University of China
No. 59, Zhongguancun Street
Haidian District
Beijing
China
Email: tong.li@ruc.edu.cn

Cheng Luo
Tencent
Jindiwei Center, Yuehai Street
Nanshan District
Shenzhen
China
Email: lancelotluo@tencent.com

Jibing Lou
Tencent
Jindiwei Center, Yuehai Street
Nanshan District
Shenzhen
China
Email: lollylou@tencent.com

Ke Xu
Tsinghua University
30 Shuangqing Road
Haidian District
Beijing
China
Email: xuke@tsinghua.edu.cn