TOO: Accelerating Loss Recovery by Taming On-Off Traffic Patterns

Xu Yan, Tong Li*
Renmin University of China

Bo Wu, Cheng Luo, Fuyu Wang
Tencent

Haiyang Wang
University of Minnesota Duluth

Ke Xu
Tsinghua University

ABSTRACT

As the ubiquitous phenomenon occurs in applications such as live streaming and video conferencing, the on-off traffic pattern is regarded as a disadvantage for congestion control. However, we argue that it can be transformed as an advantage for accelerating loss recovery. In this paper, we report the design of TOO, a loss recovery acceleration mechanism that tames on-off patterns for loss duplicate reinjection without incurring non-trivial traffic overhead.

CCS CONCEPTS

• **Networks** → **Transport protocols**.

KEYWORDS

QUIC, loss recovery, application limitation

ACM Reference Format:
Xu Yan, Tong Li, Bo Wu, Cheng Luo, Fuyu Wang, Haiyang Wang, and Ke Xu. 2023. TOO: Accelerating Loss Recovery by Taming On-Off Traffic Patterns. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3603269.3610841

1 INTRODUCTION

The ubiquitous packet loss is an essential factor affecting client-side quality-of-experience (QoE) in live-streaming services, in which the introduced head-of-line (HOL) blocking might result in long-time video freezing. Most loss recovery schemes [1–6] fall into the category of dual-side solutions that require modification or upgrade on both the server side and the client side. Unfortunately, they all suffer from deployment issues, especially under Multi-Supplier Strategy [7] that is applied by application providers (e.g., Tiktok Live) to select better-performed CDN vendors.

In fact, most modern CDN vendors usually employ the automatic-repeat-request (ARQ) paradigm [8, 9] to control loss tolerance as the commercial solution, which retransmits data once any packet is detected lost. However, from the deployment experience of real product networks, we find that legacy ARQ-based loss recovery is far from satisfactory. Our large-scale measurements show that 21.7% and 8.5% of live streams suffer from at least 2 and 5 retransmissions for recovering some lost packet(s), respectively. Besides, 6.8% of resent losses become more deteriorated (e.g., detected lost again) during recoveries, introducing additional 123.2ms (up to 279.3ms) recovery delay. Meanwhile, the performed measurements also show that on-off switching ubiquitously occurs in current live streams. In particular, over 290- and 3100-time on-off switching has actually been experienced in 50% and 10% of live streams, which results in 12s and 80s off-mode duration, respectively. Note that it is well-studied that the on-off traffic pattern is not conducive to transmission control [10]. For example, as shown in Figure 1, during the off-mode, the sender stops pushing data into the network, and the delivery rate might be underestimated. However, we argue that the "wasted" off-mode can be regarded as an essential opportunity for accelerating loss recoveries.

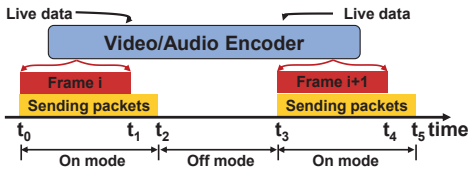


Figure 1: The sketch of on-off mode in live streams.

In this paper, we present TOO, an enhanced loss recovery mechanism that only requires single-side modification (i.e., CDN server). TOO can transform the disadvantages of on-off switching into an advantage of loss tolerance controls under large-scale live streaming. Basically, by applying TOO, traffic senders will reinject loss duplicates once entering off-streaming mode. To better balance the tradeoff between the introduced recovery benefits and the incurred traffic overhead, TOO models the client-side waiting time for any packet loss and constructs its activator, in which the proposed loss reinjection will be activated only under deteriorated network conditions (e.g., with higher deliver delays or larger loss rates). We implement TOO upon QUIC and evaluate it via real-world deployments of commercial live-streaming services, whose results demonstrate the practicability and profitability of TOO (see §3).

2 DESIGN

2.1 The TOO Framework

As a sender-side extension, TOO enables traffic senders to reinject loss duplicates once entering the off-streaming mode, as Figure 2 shows. In particular, each TOO sender establishes and maintains a reinjection queue (Q_{rein}) for each live stream, which only records retransmitted packets that has already been resent but unacknowledged yet by its receiver. To further mitigate the additional traffic

*Tong Li is the corresponding author (tong.li@ruc.edu.cn).

ACM SIGCOMM '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0236-5/23/09.
https://doi.org/10.1145/3603269.3610841

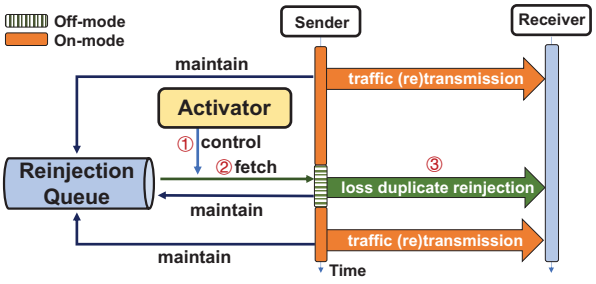


Figure 2: The TOO Framework.

overhead, TOO enables the proposed duplicate reinjection on demand, which is based on the designed activator that depicts client-side waiting time for loss recovery. As a result, only the packet losses under worse-performed connections will be reinjected.

2.2 Reinjection Queue

The TOO sender establishes and maintains a reinjection queue Q_{rein} for each live stream, in which all retransmitted packets are sorted according to the timestamps of the latest retransmission/ reinjection. In TOO, Q_{rein} will be updated if any retransmitted packet has been resent or acknowledged. Concretely, retransmitted packet will be inserted to the end of Q_{rein} if it has been retransmitted while being removed if the retransmitted packet is successfully received by its receiver. Besides, the TOO sender will move retransmitted packet from the head to the end of Q_{rein} when the proposed off-mode reinjection is performed.

2.3 Activator

To mitigate the significant reinjection overhead, TOO can identify the poor-performed live streaming, and then activate the proposed recovery optimizations for these selected streams. Actually, a live stream that suffers from a higher loss rate and larger smooth RTT (SRTT) can easily introduce unsatisfied QoE, whose senders should "turn on" the function of TOO-enabled loss reinjection in time. To better discover the activation opportunity, we model the expected value of the receiver-side waiting time ($E_{waiting}$) for any loss recovery as follows:

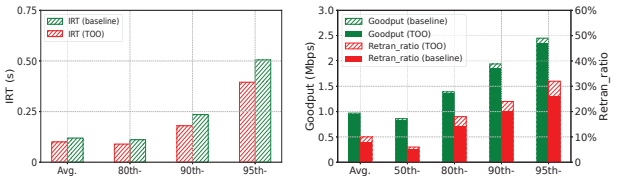
$$E_{waiting} = (1\text{-}r) \cdot SRTT \cdot \sum_{k=0}^{n-1} (k+1) \cdot r^k \quad (1)$$

where n denotes the maximum lost times of all packets, and r denotes the loss rate. In particular, $E_{waiting}$ will be updated periodically (e.g., 2×SRTT). When entering a new cycle, the sender recomputes $E_{waiting}$ based on the monitored transmission metrics (i.e., SRTT and r) of the last cycle.

In this paper, when $E_{waiting}$ exceeds its threshold Θ_{thres} (i.e., $E_{waiting} > \Theta_{thres}$), the loss reinjection of TOO will be activated. Note that Θ_{thres} actually tries to reflect the time length of available data cached at the video player. By setting a larger Θ_{thres}, TOO can achieve targeted optimizations for those live streams, especially under "imperfect" network status or conditions, e.g., with higher transmission latency and more frequent packet losses.

3 EVALUATION

We integrate all the components described in Section 2 into Tencent's CDN servers (only sender-side upgrades) and implement the TOO prototype based on the user-space QUIC protocol (with



(a) Performance gain (b) Overhead

Figure 3: The performance of TOO.

the version of LSQUIC Q043) [11]. The experimental evaluation is performed on commercial live-streaming services, where the CDN proxy server that has deployed TOO prototype can pull and transmit the requested live streaming from our live CDN to real-network users. The CDN proxy servers all employ BBR (v1) [12] scheme as the congestion controller. Note that Θ_{thres} and n in §2.3 are set as 50ms and 4, respectively, according to the production requirements.

To better depict the quality of loss recovery, we introduce a metric called **invalid-response-time (IRT)**, which is defined as the duration from when any data is detected lost to when resending a recovery packet that will be successfully received. IRT reflects the additional recovery time for data loss that ideally consumes only one SRTT. Figure 3(a) records each stream's average IRT value of the lost data whose recovery requires two or more retransmissions. We can learn the average IRT can be lowered by the ratio of 16.0%, whose values are reduced from 119.3ms to 100.2ms, respectively. In particular, the high-percentile (i.e., 95th-) IRT is even reduced by 110ms. These results demonstrate that TOO is worthwhile for optimizing the timeliness of loss recovery. As for the incurred cost, Figure 3(b) shows TOO makes the average goodput to deteriorate only with a ratio of 2.4% (from 974.3Kbps to 951.1Kbps). Besides, only 1.9% of redundancy (retran_ratio) is additionally introduced, on average, while only 10% of streams require an extra 6.5% of retran_ratio for their loss recoveries.

4 CONCLUSION

This paper proposes an enhanced recovery framework named TOO that enables senders to optimize the timeliness of loss recovery by reinjecting loss duplicates once entering control-unfriendly off-streaming mode. The real-world experiments demonstrate that TOO can effectively accelerate loss recovery of live streaming without incurring unbearable overhead.

However, our real-world deployment experience further reveals that the pre-configured and fixed transmission (e.g., recovery) policy may not well adapt to network dynamics in some cases. As future work, we are trying to design an online learning-based scheduler to adaptively determine the thresholds of both reinjection times for each data in Q_{rein} and the expected waiting time $E_{waiting}$ for the client side, which can also better balance the tradeoff between the loss recovery benefits and the incurred reinjection overhead.

ACKNOWLEDGMENTS

This work is supported by the fund from Tencent, the fund for building world-class universities (disciplines) of Renmin University of China, the NSFC Projects (No. 62202473 and No. 61932016), the China National Funds for Distinguished Young Scientists (No. 61825204), and the Beijing Outstanding Young Scientist Program (No. BJJWZYJH01201910003011).

REFERENCES

[1] Mirko Palmer, Malte Appel, Kevin Spiteri, Balakrishnan Chandrasekaran, Anja Feldmann, and Ramesh K Sitaraman. VOXEL: Cross-layer optimization for video streaming with imperfect transmission. In *ACM CoNext*, pages 359–374, 2021.

[2] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *ACM SIGCOMM*, 2021.

[3] Chao Zhou, Wenjun Wu, Dan Yang, Tianchi Huang, Liang Guo, and Bing Yu. Deadline and priority-aware congestion control for delay-sensitive multimedia streaming. In *ACM MM*, pages 4740–4744, 2021.

[4] Michael Rudow, Francis Y. Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and K.V. Rashmi. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *USENIX NSDI*, pages 953–971, 2023.

[5] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. Tack: Improving wireless transport performance by taming acknowledgments. In *ACM SIGCOMM*, pages 15–30, 2020.

[6] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. Revisiting acknowledgment mechanism for transport control: Modeling, analysis, and implementation. *IEEE/ACM TON*, 29(6):2678–2692, 2021.

[7] Yasemin Arda and Jean-Claude Hennet. Inventory control in a multi-supplier system. *International Journal of Production Economics*, 104(2):249–259, 2006.

[8] D BertsekasandR. Gallager, data networks. *Prentice-Hall*, 1(99):2, 1992.

[9] Hui Xie and Li Tong. Revisiting loss recovery for high-speed transmission. In *IEEE WCNC*, pages 1–6, 2022.

[10] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1):92–99, 2010.

[11] LiteSpeed Tech. LiteSpeed QUIC and HTTP/3 Library. https://github.com/litespeedtech/lsquic.

[12] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. BBR: congestion-based congestion control. *Communications of the ACM*, 60(2):58–66, 2017.