ART: Adaptive Retransmission for Wide-Area Loss Recovery in the Wild

Tong Li†, Wei Liu†, Xinyu Ma†, Shuaipeng Zhu†, Jingkun Cao†, Senzhen Liu‡, Taotao Zhang‡, Yinfeng Zhu‡, Bo Wu§, and Ke Xu§
Renmin University of China†, ByteDance‡, Tsinghua University§

***Abstract*—Packet losses significantly impact the user experience of wide-area applications such as content distribution and remote procedure call (RPC) based services. However, our production network measurement studies show that the legacy loss recovery is far from satisfactory due to the wide-area loss characteristics (i.e., dynamics and burstiness) in the wild. In this paper, we propose a sender-side Adaptive ReTransmission scheme, ART, which minimizes the recovery time of lost packets with minimal redundancy cost. Distinguishing itself from forward-error-correction (FEC), which preemptively sends redundant data packets to prevent loss, ART functions as an automatic-repeat-request (ARQ) scheme. It applies redundancy specifically to lost packets instead of unlost packets, thereby addressing the characteristic patterns of wide-area losses in real-world scenarios. We implement ART upon QUIC protocol and evaluate it via both trace-driven emulation and real-world deployment. The results show that ART reduces up to 34% of flow completion time (FCT) for delay-sensitive transmissions, improves up to 28% of goodput for throughput-intensive transmissions, and saves up to 90% of redundancy cost.**

***Index Terms*—loss pattern, loss recovery, redundancy**

I. Introduction

The ubiquitous wide-area packet loss is a critical factor affecting the performance of geo-distributed applications including both delay-sensitive applications (e.g., live video streaming, interactive online gaming, and remote procedure call (RPC) based services [1]) and throughput-intensive applications (e.g., disaster recovery, cloud migration, and data backup-and-archiving [2]). Take a famous content delivery network (CDN) platform as an example [3], the flows incur an average 5.2% packet loss rate in Turkey and 3.8% in Brazil, respectively. The flow completion time (FCT) in these regions is enlarged because the high loss rate introduces head-of-line (HOL) blocking and even incurs service failure when loss recovery is excessively delayed.

There exist two fundamental ways of loss recovery in transmission control, i.e., forward-error-correction (FEC) [4]–[6] and automatic-repeat-request (ARQ) [7]–[11]. However, it is well-studied that FEC is far from satisfactory due to the wide-area loss characteristics such as *burstiness* in the wild [6].

This work is supported by the fund from ByteDance, the NSFC Projects (No. 62202473 and No. 61932016), the fund for building world-class universities (disciplines) of Renmin University of China, the China National Funds for Distinguished Young Scientists (No. 61825204), and the Beijing Outstanding Young Scientist Program (No. BJJWZYJH012019100030011).

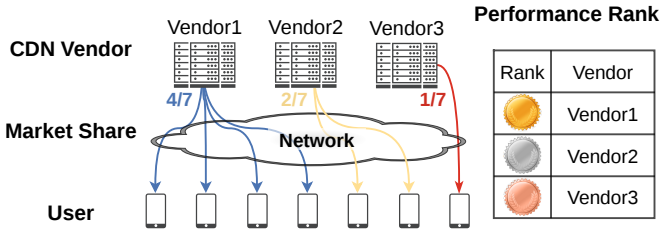
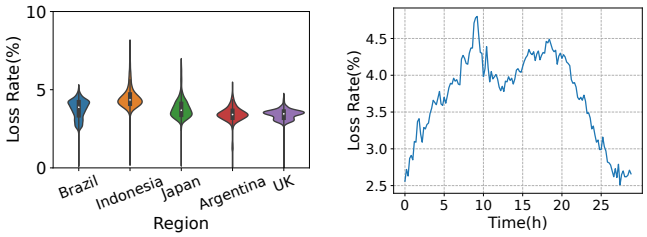


Fig. 1: An example of the multi-supplier market for wide-area applications.

Although some works are proposed to protect against bursts of losses [5], [6], they require dual-side (i.e., CDN server and client-side application) modifications. These FEC-based advancements might suffer from deployment issues in the multi-supplier CDN market. As shown in Figure 1, application operators (e.g., TikTok Live) usually apply the Multi-Supplier Strategy [12] in the CDN market. As a result, it is the CDN vendor's duty that optimize the transmission performance (e.g., loss recovery), according to which the application operators will choose the better-performed CDN vendors to carry more traffic (i.e., larger market share). In this case, only server-side sending policies can be adjusted by the selected CDN vendors, which lack the proper authority to synchronize client-side control rules.

In fact, most modern applications only apply the ARQ paradigm to control loss tolerance as the commercial solution, which retransmits data once any packet is detected lost. Unfortunately, from our production network measurement studies, we find that wide-area loss shows characteristics of dynamics and burstiness in the wild (see §II). We further find that the legacy ARQ-based loss recovery induces unexpected latency due to poor adaption to these loss characteristics. Specifically, it still suffers from both *data reassembling starvation* and *receiving buffer starvation* in delay-sensitive transmissions and throughput-intensive transmissions, respectively (see §III).

In this paper, we propose the adoption of FEC to improve loss recovery in ARQ-based protocols. Unlike the traditional FEC-based approach, which involves sending redundant unlost data packets, our method focuses on sending redundant lost packets without the need for coding, thereby avoiding dual-side modifications and expediting the loss recovery process. Furthermore, our approach has the potential to eliminate the costly traffic overhead typically associated with traditional



(a) Loss distribution in different regions (b) Loss dynamics

Fig. 2: Examples of loss dynamics in the wild.

FEC. This is due to the fact that the number of lost packets is usually much smaller than the number of unlost packets. Introducing redundancy to retransmission may seem straightforward, but in practice, the dynamics and burstiness of packet loss present several challenges that must be addressed: (i) The trade-off between performance and cost is decided by the redundancy level, which refers to the number of replicas of a lost packet. The optimal redundancy level for each loss varies with the dynamics of the loss, and striking the right balance is crucial. (ii) A loss event usually contains multiple consecutive lost packets (i.e., burstiness), the redundant replicas might be lost together again when burst losses occur.

To tackle these issues, we present ART[1] (Adaptive ReTransmission), a sender-side approach without requiring any modifications on the receiver side. Simple but useful, ART overcomes the above-mentioned challenges through two tightly coupled systems: Redundancy Adaption and Replica Scheduling. To adapt to loss dynamics, Redundancy Adaption alters the redundancy level step by step using the method of test and then verification. To deal with the loss burstiness, Replica Scheduling schedules the replicas in a random number of sending cycles (one cycle equals the interval of sending one packet at a specific pacing rate). With Redundancy Adaption and Replica Scheduling, ART can achieve consistently rapid loss recovery with the minimized cost of redundant traffic.

The contributions are summarized as follows.

- We conduct large-scale measurement studies on the wide-area loss characteristics in the wild, and disclose that dynamics and burstiness are two key features of loss in practice (see §II). And then we identify the critical challenges when facing the wide-area loss characteristics (see §III).
- We propose ART that can dynamically adjust the redundancy level according to the loss dynamics, and can randomly schedule the sending of each replica so as to protect against bursts of losses (see §IV and §V).
- We implement ART prototype in the user-space QUIC protocol and deploy it on both the testbed and production network (see §VI). The experimental results demonstrate the practicability and profitability of ART, in which ART can accelerate loss recovery with the minimum extra traffic overhead by automatically adjusting the sender's loss recovery capability. For example, ART reduces the FCT by up to 34% for delay-sensitive transmissions when suffering data reassembling starvation. ART also improves up to 28% of goodput for throughput-intensive transmissions when suffering receiving buffer starvation. Furthermore, ART achieves up to 90% of lower redundancy cost with considerable recovery performance (see §VII).

[1]The open-source implementation is maintained at https://github.com/litonglab/quic-art

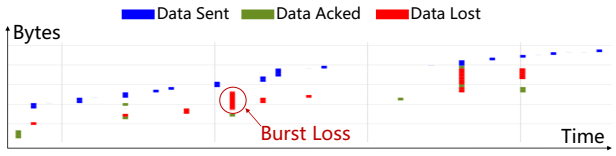
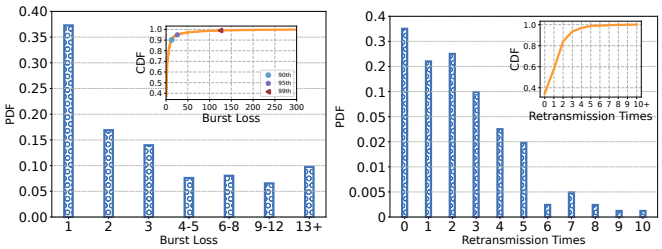


Fig. 3: An example of loss burstiness in the wild.



(a) Burst size of loss (b) Maximum retransmission times

Fig. 4: (a) Burst loss distribution in the wild. The x-axis is the number of continuously lost packets. (b) The distribution of maximum retransmission times in the wild.

II. Loss in the Wild: A Measurement Study

Logs from our production network (i.e., the ByteDance public cloud) are collected from a random sample over two weeks. Each log corresponds to a QUIC connection, which contains the size of a connection, sequence number, and packet sending and loss information of multiple QUIC streams. We analyze these logs and record the specific information of packet loss, such as the times of loss events and the burst size of the loss. We measure over 200,000 connections from different application scenarios in different regions all around the world.

A. Loss Shows Dynamics

We first explore the distribution of the loss rate (every 5 minutes) of each connection in different worldwide regions. Figure 2(a) shows the results. Although there are regions with similar average packet loss rates, they have different packet loss rate deviations (i.e., the violin shapes are different). For example, Brazil and Japan both have an average packet loss rate of 3.78%, but Japan has the highest packet loss rate of 7.1% while Argentina has only 5.7%. Figure 2(b) further gives an example of how the packet loss rate evolves over time. Specifically, the loss rate is never constant and varies from 2% to 5% during 24 hours. This confirms that loss shows dynamics in the wild.

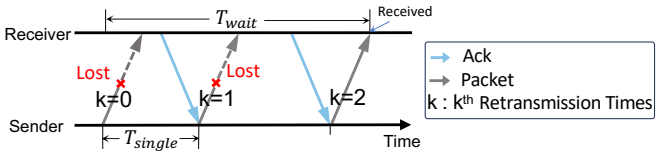


Fig. 5: An example of the retransmission loss.

B. Loss Shows Burstiness

Figure 3 gives an example of loss burstiness (denoted by red blocks) in a randomly selected connection. We further explore the burst loss distribution in the production network. Figure 4(a) shows the results. Surprisingly, the probability of only one packet being lost is not as high as imagined (i.e., 37.2%), instead, the probability of losing multiple packages (≥ 2) together accounts for a larger proportion (i.e., 62.8%). Specifically, as illustrated in the sub-figure of Figure 4(a), the 90^{th}, 95^{th}, and 99^{th} percentile burst size of loss (i.e., the number of continuously lost packets) is 13, 27, and 125, respectively, showing extremely bursty packet loss. Numerous factors contribute to the emergence of the phenomenon of loss burstiness. Here we provide readers with a duo of potential explanations worthy of contemplation. First, it might be due to the queue management strategy [6] or traffic handling policy [13] employed by the Internet Service Provider (ISP). Moreover, burst packet loss can also be caused by wireless interference in unstable access networks of end users [14].

C. Retransmission Loss is Ubiquitous

A packet might be retransmitted more than once before it is correctly received by the receiver. By counting the number of occurrences of each frame according to the QUIC *stream offset*, we can further analyze how many times each packet was retransmitted before successfully received [15]. Given a connection, its maximum retransmission times can be computed as the maximum value of retransmission times among all packets in a connection. Figure 4(b) shows the results of the distribution of the maximum retransmission times in the production network. We find that the proportion of connections with maximum retransmission times of two or more exceeds 43%. Among them, a considerable portion of the connections have certain packets that are retransmitted even more than 10 times. This retransmission loss is harmful to both delay-sensitive applications and throughput-intensive applications, as we will discuss next.

III. Wide-Area Loss Recovery: Issues and Challenges

Ideally, successful delivery of a lost packet should only require a single retransmission to the receiver. However, our previous measurement studies have shown that retransmission loss is ubiquitous. Next, we will investigate the impacts on transmission performance when a single retransmission attempt is insufficient to successfully deliver a lost packet.

A. Delay-Sensitive Transmission Suffers From Data Reassembling Starvation

Consider a distributed system that relies on RPC for inter-process communication. If a critical RPC call is lost and experiences a high recovery latency, the client's request will be delayed, and subsequent dependent operations may be blocked. This can result in a significant increase in flow completion time, negatively affecting the responsiveness and overall performance of the service. Real-world scenarios, such as financial trading platforms or online multiplayer games that heavily rely on quick response times, can suffer from degraded user experience and potential financial losses due to increased recovery latency.

Under these circumstances, the prioritizing loss recovery attempts to mitigate receiver-side waiting time (T_{wait}) for the lost data and enables delivery of the follow-up data (that has already been received) to the application layer. As illustrated in Figure 5, we define T_{wait} as the duration between when a packet is sent and when the packet is first received, and T_{single} as the duration between when a packet is sent and when the packet is detected lost. Then we have

$$T_{wait} = K \cdot T_{single} + \frac{rtt}{2} \tag{1}$$

where rtt refers to the round-trip time (RTT) and K refers to the retransmission times of a specific packet before successfully being received at the receiver. If a packet is never lost and retransmitted, then $K = 0$. Given a certain loss detection algorithm, we have

$$T_{single} \propto rtt \tag{2}$$

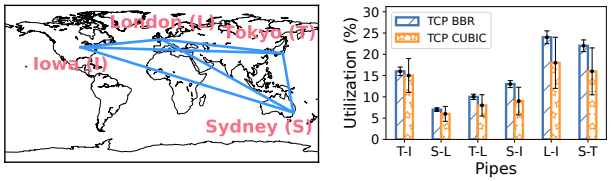
Furthermore, given the RTT, we have

$$T_{wait} \propto K \tag{3}$$

Based on the above analysis, we infer that delay-sensitive transmission suffers from data reassembling starvation in the case of the large K. However, the current loss recovery paradigms mainly rely on rapid loss detection (i.e., optimizing T_{single}) while ignoring retransmission times (i.e., optimizing K). For example, RACK [16] is regarded as the state-of-the-art loss detection advancement that assures a relatively deterministic T_{single}, i.e., $T_{single} \approx 1.25rtt$. In this case, Equation (1) is instantiated as $T_{wait} \approx (1.25K + 0.5)rtt$.

B. Throughput-Intensive Transmission Suffers From receiving buffer Starvation

Transport protocols like TCP and QUIC [15] provide reliable and ordered byte-stream transmission. As a result, before being handed off to upper applications, the subsequent packets (stored temporarily in the receiver's queue) of the lost packet will be stalled in the receiving buffer until the "hole" (the lost data sequence space) is filled via retransmissions. However, retransmissions might be lost again. Since the receiving buffer required by a connection is closely related to the maximum times of retransmissions, we focus on the metric of K_{max}, the maximum value of K among all packets in a connection, where K denotes the retransmission times of a specific packet.



(a) Topology [17] (b) Bandwidth utilization

Fig. 6: Pipes are verified not full but TCP only achieves bandwidth utilization of less than 25%. Where bandwidth (bw) is 1 Gbps, $rtt \in [100, 300]$ ms, and average loss is 0.5%. The maximum receiving buffer of TCP is 16 MB in the cloud end-hosts [17], [18].

Based on the above analysis, we infer that throughput-intensive transmissions might suffer from receiving buffer starvation in the case of a large K_{max}. This issue is unremarkable for transmissions when both throughput and loss are low. However, when running under large-BDP and lossy network conditions, the buffer starvation issues may significantly impact the performance of throughput-intensive transmissions. Figure 6 shows an example of how high-throughput transmission bottlenecks the receiving buffer in the Pantheon [18]. Based on the fact that the receiving buffer is usually small (*e.g.*, 16 MB in the *c5.xlarge* instances of Amazon EC2) in modern wide-area Linux servers. It is observed that buffer limitation makes TCP only achieve bandwidth utilization of less than 25% even when there is sufficient available bandwidth.

It's worth noting that network operators often increase the end-host buffer through protocol tuning to address the capability mismatch between loss recovery and high-throughput requirements. However, this approach is ineffective when operators only have control over one side of the network, such as in public cloud services where CDN vendors cannot modify the end devices.

In summary, it may cause both data reassembling starvation and receiving buffer starvation when a single retransmission attempt is insufficient to successfully deliver a lost packet (i.e., when $K > 1$). Hence it becomes crucial to carefully manage loss recovery at the sender side, minimizing K for each loss when data reassembling is excessively delayed, and minimizing K_{max} for all the losses in the connection when the end-host buffer is insufficient. These efforts would provide a valuable contribution from the perspective of CDN vendors, which encourages the development of ART, as we will elaborate next.

IV. The ART Overview

The main objective of ART is to ensure the successful reception of each lost packet by the receiver, minimizing the receiver-side waiting time of each packet. However, in order to effectively implement ART in production networks where cloud services (sender side) charge based on traffic volume, it is vital to not only prioritize the quick recovery of lost packets but also minimize redundancy cost and prevent any detrimental impact on regular packet transmission. Striking a balance between rapid recovery and efficient resource utilization is crucial for the practicality of ART in production networks.

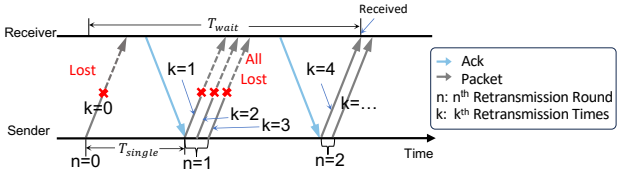


Fig. 7: An example of the retransmission loss when applying redundant retransmission. For this lost packet, the retransmission times $K = max(k) = 4$ and the retransmission round $N = max(n) = 2$.

Conceptual definition. The fundamental concept behind ART is to introduce redundancy during retransmissions, i.e., *redundant retransmission*, merging the features of FEC and ARQ. Instead of the legacy FEC that applies redundancy to the original (unlost) packets, ART applies redundancy specifically to the retransmitted (lost) packets themselves. When considering redundant retransmissions, it is necessary to re-evaluate the variability in waiting time on the receiver side (represented by T_{wait}). To address this, we propose a novel concept called "retransmission round" (designated as N) to replace the retransmission times (K) in Equation (1) when computing T_{wait}:

$$T_{wait} = N \cdot T_{single} + \frac{rtt}{2} \tag{4}$$

where N refers to the retransmission round of a specific packet before successfully being received at the receiver. Figure 7 illustrates an example of the difference between N and K, where n represents the n^{th} retransmission round and k represents k^{th} retransmission times. For each packet, we have $N = max(n)$ and $K = max(k)$. Similar to K_{max}, we define the maximum retransmission round (N_{max}) for a connection that transmits multiple packets as

$$N_{max} = max(N) \tag{5}$$

In the absence of redundancy, these two terms (N and K) are essentially interchangeable. However, in the presence of redundancy, the retransmission times will accumulate within the same retransmission round. Specifically in the example of Figure 7, the sender retransmits 3 replicas of the lost packet (i.e., $k = 1, 2, 3$) during the first retransmission round (i.e., $n = 1$). Generally, whenever a packet is retransmitted, the count of retransmission times for that packet increases by 1. However, only when all retransmitted packets for a specific packet in a complete retransmission round are identified as lost, will the retransmission round for that packet be increased by 1.

Redundancy adaptation. The redundancy level, denoted as R, is defined as the number of replicas that should be resent for a specific lost packet. R_n represents the redundancy level within the n^{th} retransmission round. For instance, in the case of Figure 7, we have $R_1 = 3$. We define the *redundancy cost* as

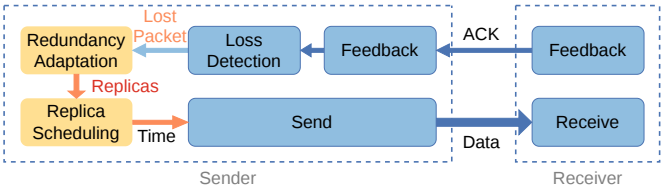


Fig. 8: Key modules in ART.

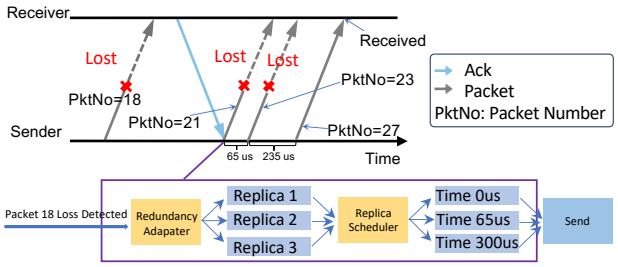
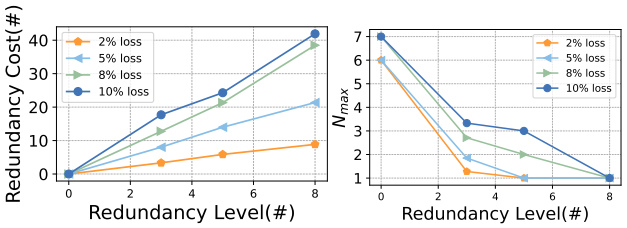


Fig. 9: An example of the ART workflow.

the total number of replicas that are sent during transmission. Initially, a naive approach for ART would be to use a fixed redundancy level for all rounds, i.e., $R_n = R$ $(n = 1, 2, ..., N)$. However, our deployment experiences have shown that this fixed approach to redundant retransmission is suboptimal (high redundancy cost or long recovery latency) due to the dynamic nature of packet loss in real-world scenarios (see §V-A). Consequently, we incorporate redundancy adaptation to allow the redundancy level to vary dynamically. To achieve this, we introduce the *Redundancy Adapter*, which gradually learns the feature of loss dynamics and carefully selects the most appropriate redundancy level for each retransmission round of lost packets. This ensures that ART can adapt to the dynamics of packet loss while minimizing the redundancy cost.

Replica scheduling. For each round of retransmission, more than one replica might be injected into the network. Initially, a naive approach for ART would be to send all the replicas (with the same retransmission round) at once if the send window is sufficient. However, our deployment experiences have shown that this burst send pattern may fail to accelerate loss recovery due to the burst nature of packet loss in real-world scenarios (see §V-B), that is, a loss event usually contains multiple consecutive losses, the back-to-back replicas might be all lost again under a burst of losses. In this case, the redundant replicas should be carefully scheduled. Consequently, we introduce the *Replica Scheduler*, which adopts randomization to enable the redundant replicas to be sent out in a random number of sending cycles. This ensures that ART can adapt to the burstiness of packet loss.

The architecture of ART. ART is a sender-side modification to the protocol stack whose key modules are illustrated in Figure 8. Particularly, once a loss is detected, ART adopts redundancy adaption to compute the number of replicas of the lost packet that should be retransmitted next. Given the number of replicas, ART then adopts replica scheduling to determine the specific time interval for sending out each replica from



(a) Redundancy cost (b) N_{max}

Fig. 10: Cost and Performance of ART under different redundancy levels, where $bw = 100$ Mbps and $rtt = 100$ ms.

the sender. To explain this more clearly, we further give an example of the ART workflow in Figure 9. Assuming a packet (packet number = 18) is detected lost. In this case, ART first runs the redundancy adapter to compute the redundancy level for packet 18, we assume redundancy level = 3. Then ART runs the replica scheduler to determine when to send each replica. For example, the 3 replicas should be sent in 0 us, 65 us, and 300 us, respectively. As a result, the replicas are sent out with packet numbers 21, 23, and 27, respectively. Since the 3 replicas are sent at various intervals, it can greatly enhance the likelihood of a successful retransmission.

V. Detailed Design

In this section, we give the detailed design of ART for its practical deployment.

A. Redundancy Adapter

We first elaborate on the design of the Redundancy Adapter by answering the following questions below.

Why dynamic redundancy level matters? To answer this question, we conducted an investigation into the performance of ART under different loss rates ($p = 2\%, 5\%, 8\%, 10\%$) and different redundancy levels ($R = 0, 3, 5, 8$). As shown in Figure 10(a), when the redundancy level increases, there is a corresponding rise in the redundancy cost. On the other hand, the redundancy levels also significantly impact the maximum retransmission round (N_{max}). As shown in Figure 10(b), the higher the redundancy level, the lower the N_{max}. Intuitively, the optimal redundancy level should be set at the *inflection point*. However, the experimental results demonstrate that the inflection points vary with the loss rates. We then infer that the redundancy level should be set dynamically according to network dynamics. This greatly motivates the design of the Redundancy Adapter, as we will elaborate below.

How to set redundancy level dynamically? Primarily, we establish the replica loss rate (p_r) by dividing the number of lost replicas by the total number of sent replicas. While it is acknowledged that redundancy levels should be adjusted according to the loss rates, it is imperative to consider p_r instead of the general packet loss rate (p) for a more accurate design of the Redundancy Adapter. In this paper, we propose a sliding-window-based way to predict p_r according to the historical packet delivery information. We set up a bitmap

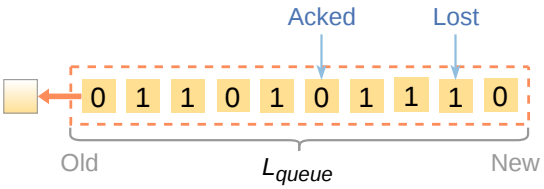


Fig. 11: An example of the sliding-window-based bitmap queue. If a replica is acknowledged, a "0" is appended to the queue. Otherwise, a "1" is appended to the queue.

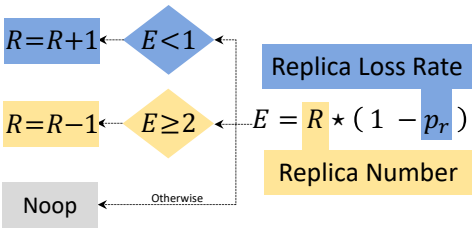


Fig. 12: The design rationale of the Redundancy Adapter.

queue at the sender. As illustrated in Figure 11. When the sender receives an acknowledgment for a redundant packet, a "0" is appended to the queue. Otherwise, a "1" is appended to the queue, indicating that the redundant packet is lost again. In this case, p_r is computed as the proportion of "1" in the queue, i.e., $p_r = \frac{c}{L_{queue}}$, where L_{queue} is the queue length measured in the number of packets (e.g., $L_{queue} = 10$), and c is the number of "1" in the queue. Then the expected number of successfully delivered replicas (denoted as E) can be computed as follows:

$$E = R \cdot (1 - p_r) \tag{6}$$

where R is the redundancy level within a retransmission round. In general, to ensure the effectiveness of redundant retransmission where we only send the number of replicas that is exactly required by the loss recovery, we should keep $E \approx 1$. This is because the unnecessary traffic cost arises when $E > 1$, and recovery latency arises when $E < 1$. To accomplish this, a step-by-step online algorithm is applied. As illustrated in Figure 12, when $E < 1$, then $R = R + 1$; When $E \geq 2$, then $R = R - 1$; Otherwise, R remains unchanged.

B. Replica Scheduler

The redundant replicas might be lost together again when burst losses occur. To withstand burst loss, we introduce the Replica Scheduler to ensure prompt delivery of replicas to the receiver. Instead of sending out all the replicas at once, the Replica Scheduler disperses the multiple replicas into a certain number of sending cycles (one cycle equals the interval of sending one packet at a specific pacing rate). That is, replicas are interspersed with normal packets.

Specifically, given multiple replicas of a specific lost packet, we define the *escape space* (denoted by $escape\_space$) of this lost packet as the total number of sent packets from starting sending its first replica to finishing sending its last replica. For example, as shown in Figure 13, for the traditional QUIC,

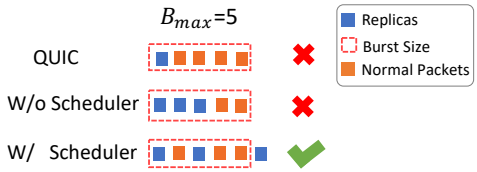


Fig. 13: The design rationale of the Replica Scheduler. Where schemes QUIC, W/o scheduler, and W/ scheduler represent the traditional QUIC, the ART without Replica Scheduler, and the ART with Replica Scheduler, respectively.

we have $escape\_space = 1$. For ART without applying the Replica Scheduler, the escape space of each packet equals the redundancy level, i.e., $escape\_space = R$.

We use B to denote the burst size of loss during transmission. Intuitively, when $B \leq R$, the Replica Scheduler is not a mandatory measure that should be taken. However, when $B > R$, ART depends on the Replica Scheduler to expand the escape space, i.e., $escape\_space \geq B$. This assures that at least one replica "escapes" the burst loss and is successfully delivered. On the other hand, all replicas should be sent in one RTT to ensure an orderly control loop. Based on these observations, we finally give the guideline of how to determine the $escape\_space$ as follows:

$$escape\_space = \min(bdp, \alpha \cdot B_{max}) \tag{7}$$

where bdp refers to the bandwidth and delay product, α refers to a scaling coefficient ($\alpha \geq 1$), and B_{max} refers to the maximum burst size of the loss in the past several RTTs (e.g., $5 \sim 10$ RTTs). Figure 13 shows an example of the ART applying the Replica Scheduler, where $B_{max} = 5$ and $R = 3$. We can see that the distinctive responses of three scenarios (i.e., QUIC, ART without Replica Scheduler, and ART with Replica Scheduler) become evident when confronted with burst loss. Notably, in this specific scenario, only ART featuring the Replica Scheduler demonstrates resilience against the ramifications of burst loss.

VI. Implementation

We implement ART in the user-space QUIC protocol based on LSQUIC commit *850b0a3* [19], consisting of 600+ lines of code [20]. ART only requires modifying the retransmission logic at the sender without interfering with other components such as congestion controllers.

For the Redundancy Adapter implementation, we reuse the function of send_ctl_handle_regular_lost_packet() to generate replicas with newly assigned packet numbers. These replicas are incorporated into the packet chain called po_loss_chain. In order to manage retransmission rounds effectively, we add two additional variables to the packet properties. Specifically, the variable po_retrans_times keeps track of the retransmission rounds, while po_retrans_packet_number records the packet number in each round. We also maintain a queue specifically designated to record the transmission states of repli-

cas. When an ACK for a replica is received (detected by lsquic_send_ctl_got_ack()), or when the packet is identified as lost (detected by send_ctl_detect_losses()), the corresponding states in the queue are updated accordingly.

For the Replica Scheduler implementation, we reuse the alarm function of lsquic_alarmset_set() and add a new alarm AL_ART_SCHE in the alarm_set. We also include an attribute called po_expected_sent to record the expected time at which the next replica should be sent. When the AL_ART_SCHE alarm expires, the Replica Scheduler sends out the replicas according to their po_expected_sent. To determine the value of po_expected_sent, we randomly select a time interval less than escape_space, which is updated each retransmission round according to the max_burst_size.

VII. Evaluation

In this section, we conducted experiments to investigate the performance of ART in both the testbed and production networks. We focused primarily on the following questions: (1) What are the advantages of ART when compared with existing technologies? (2) How does ART accelerate packet loss recovery for delay-sensitive transmissions? (3) How does ART accelerate packet loss recovery for throughput-intensive transmissions? (4) What is the cost of using ART? And (4) how does ART work in practice?

A. Methodology

Experiment setup. In our evaluation, we utilized two distinct networks to comprehensively assess the performance of our approach, namely: (1) Trace-driven testbed network using the mahimahi [21] simulation software. This versatile software allowed us to accurately replicate real-world network conditions by replaying publicly available network traces. The credibility and widespread adoption in numerous research papers [22], [23] further solidify its suitability for our simulations. (2) Production network deployment involved the implementation of ART on a QUIC server, which was generously provided by ByteDance CDN service. This deployment encompassed a diverse array of network links and user profiles, providing a comprehensive and practical assessment of our approach's effectiveness in real-world scenarios.

Schemes. Throughout our subsequent analyses, we compare ART with the following baselines:

- **QUIC**: The traditional QUIC (LSQUIC commit *850b0a3* [19]) without ART or FEC. By default, QUIC adopts the traditional ARQ where it retransmits only one replica at a time when a packet is detected lost.
- **OR3**: The QUIC with OR3 [17], the pioneering approach on redundant data transmission. Note that OR3 adopts a variable redundancy level ($R = 2n-1$), where n denotes the n^{th} retransmission round.
- **FEC**: The QUIC with FEC, which adopts the block codes [24] as the error correcting codes. The block codes approach divides the packets into different blocks. The redundancy ratio of FEC is quantified as the proportion between the count of regular data packets and the count of redundant packets present in each block. For example, a redundancy ratio of 9:1 means 9 data packets are used to create 1 redundant packet. When one of these 10 packets in the block is lost, the 9 data packets can still be recovered.
- We also introduce some variants of QUIC. For example, QUIC ($R = 1$) refers to the QUIC that adopts a fixed redundancy level ($R = 1$) and QUIC ($R = 2n+1$) refers to the QUIC that adopts a variable redundancy level ($R = 2n + 1$), where n denotes the n^{th} retransmission round.

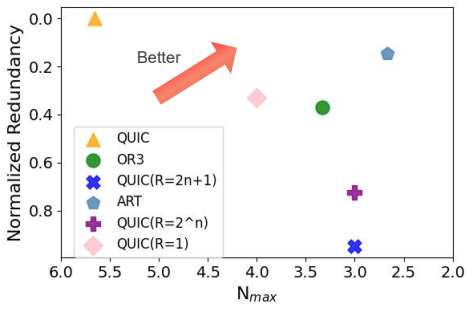


Fig. 14: Overall performance comparison between ART and other schemes. The network condition is set as $bw = 100$ Mbps and $p = 4\%$.

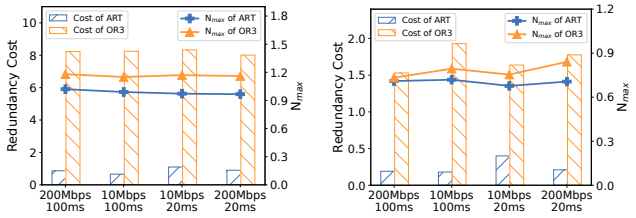
Traffic patterns. In our evaluation, we measure various types of objects, encompassing both file transfers and RPC requests. File transfers have been subjected to evaluation in both testbed networks and production networks, whereas RPC requests have exclusively undergone production-network measurement. The size of RPC requests adheres to the actual body sizes observed on a real-world platform, ensuring the authenticity of our assessment. To ensure a comprehensive evaluation, we have included a diverse range of file sizes, spanning from tens of kilobytes to hundreds of megabytes, thus providing a thorough analysis of the system's performance across different data loads.

Metrics. In delay-sensitive scenarios, our attention is directed toward crucial metrics such as packet recovery time and FCT. Conversely, in throughput-extensive scenarios, our primary consideration centers around achieving optimal goodput. Furthermore, in both of these contexts, meticulous scrutiny is given to factors like the (maximum) retransmission rounds and redundancy cost. The former serves as a reflection of performance, while the latter provides insights into the associated overhead.

B. Overall Performance

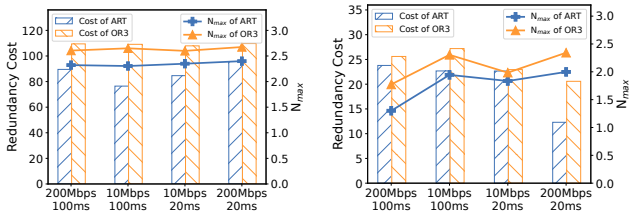
Before diving into the benefit details of applying ART, we first explore the overall performance and cost of different schemes.

Comparison with QUIC and OR3. The performance is represented by the maximum retransmission round (N_{max}) and the cost is represented by the redundancy cost. We



(a) File size=1MB, p=1% (b) File size=200KB, p=1%

Fig. 15: Performance and cost with a packet loss rate of 1%.



(a) File size=1MB, p=10% (b) File size=200KB, p=10%

Fig. 16: Performance and cost with a packet loss rate of 10%.

compare the performance and cost between ART and multiple QUIC variants by transferring 100MB files in the testbed with $bw = 100$ Mbps, $rtt = 30$ ms, and $p = 4\%$. Figure 14 shows the results. The findings from our investigation affirm the superiority of ART in multiple aspects. Notably, it boasts the smallest N_{max}, outperforming all the other QUIC variants. Additionally, ART exhibits a remarkable advantage by having less redundancy cost when juxtaposed with its counterparts. This demonstrates that ART is able to reduce the recovery time of lost packets without imposing significant redundancy costs.

Note that both OR3 and ART accelerate loss recovery, however, ART achieves better performance-cost efficiency. This is due to the adoption of the Redundancy Adaption in ART. To explain this more clearly, we further explore the differences between ART and OR3 under more types of network conditions. As shown in Figures 15 and 16, we run tests in the testbed with file sizes ranging from 200 KB to 1 MB, packet loss rate ranging from 1% to 10%. We observe that ART substantially reduces the redundancy costs, especially in scenarios with lower packet loss. For example in Figure 16, when $p = 10\%$, ART reduces the costs by $12\% \sim 40\%$. While in Figures 15, when $p = 1\%$, the reduction of costs of ART increases to $75\% \sim 90\%$. This is because ART is designed based on network feedback, adapting to the network conditions, and requiring fewer replicas when the network performs well. In contrast, OR3 relies solely on the packets themselves and does not react to external network factors, leading to nearly constant redundancy levels across various packet loss rates.

Comparison with FEC. Since ART is proposed as an ARQ-enhanced scheme that selectively incorporates FEC for lost

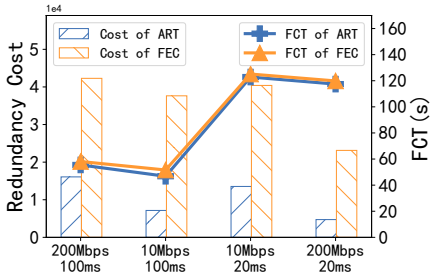
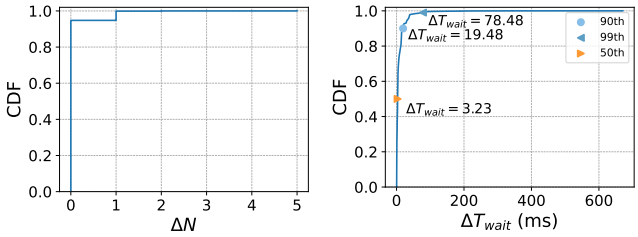


Fig. 17: Overall performance comparison between ART and FEC.



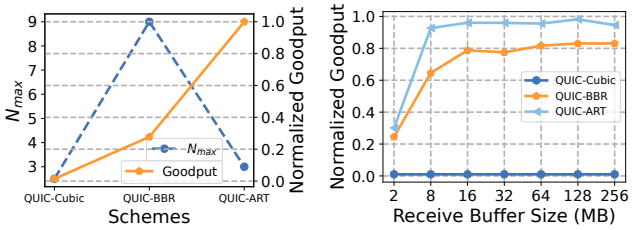
(a) ΔN (b) ΔT_{wait}

Fig. 18: (a) The distribution of the reduction of retransmission rounds. $\Delta N = N_{quic} - N_{art}$, where N_{quic} and N_{art} denote the retransmission round of a packet in QUIC and in ART, respectively. (b) The distribution of the reduction of recovery latency. $\Delta T_{wait} = T^{quic}_{wait} - T^{art}_{wait}$, where T^{quic}_{wait} and T^{art}_{wait} denote the recovery latency of a packet in QUIC and in ART, respectively.

packets, it is necessary to compare the performance and cost between ART and FEC. In this experiment, the redundancy ratio of FEC is set to 9:1 to assure a relatively low redundancy cost, the network condition is set as $bw = 100$ Mbps, $rtt = 20$ ms, and $p = 4\%$. We investigate both the FCT as the performance and the redundancy cost as the cost. Figure 17 shows the results. It reveals that, on the whole, ART does not yield a noteworthy enhancement in FCT when juxtaposed with FEC. However, it significantly reduces the redundancy cost, minimizing the redundant packet occurrences in comparison to FEC. This demonstrates that ART is able to achieve a good trade-off between performance and cost.

C. Mitigating Data Reassembling Starvation of Delay-Sensitive Transmissions

According to Equation (4), the delay-sensitive transmission suffers from data reassembling starvation in the case of a large N, the retransmission round of lost packets. To explore how ART tackles this issue, we first investigate how ART reduces N, and then investigate how ART reduces the loss recovery latency T_{wait}. We have conducted a comprehensive series of measurements using the testbed to analyze the distributions of N and T_{wait} during the transmission of 100 MB files. Specifically, the packet loss rate is varied within the range of 1% to 10%. The bandwidth is varied within the range of 10 Mbps to 200 Mbps, while the RTT is set to span from 20 ms to 100 ms.



(a) Goodput and N_{max} (b) Goodput

Fig. 19: (a) An example of the relationship between N_{max} and goodput, where $bw = 100$ Mbps, $rtt = 300$ ms, $p = 5\%$, and receiving buffer size = 8 MB. (b) Performance in the case of receiving buffer starvation, where $bw = 100$ Mbps, $rtt = 300$ ms, $p = 5\%$.

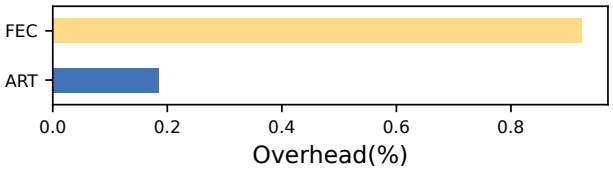


Fig. 20: Comparison of extra CPU overhead between ART and FEC.

Reducing retransmission rounds of lost packets. We compute the reduction of the retransmission round (ΔN) as $\Delta N = N_{quic} - N_{art}$, where N_{quic} denotes the retransmission round of a packet in traditional QUIC without applying ART and N_{art} denotes the retransmission round of the packet in QUIC applying ART. As shown in Figure 18(a), we explore the distribution of ΔN by running tests with the traditional QUIC and ART. It is noteworthy that we have tried our best to make the measurement conditions similar for ART and QUIC in each test. It is observed that in most cases ΔN keeps the value of 0, this is because most losses can be recovered via a single retransmission round. However, it also shows that ART may significantly reduce the retransmission rounds of lost packets, especially in the worst cases. For example, ΔN is up to 5 when the packets are excessively lost.

Reducing loss recovery latency. We compute the reduction of the loss recovery latency (ΔT_{wait}) as $\Delta T_{wait} = T^{quic}_{wait} - T^{art}_{wait}$, where T^{quic}_{wait} denotes the recovery latency of a packet in QUIC without applying ART and T^{art}_{wait} denotes the recovery latency of the packet in QUIC applying ART. As shown in Figure 18(b), we explore the distribution of ΔT_{wait} by testing the traditional QUIC and ART. It is observed that ART significantly reduces the recovery latency of lost packets. Specifically, the 50^{th}, 95^{th}, and 99^{th} percentile ΔT_{wait} are 3.23 ms, 32.83 ms, and 78.48 ms, respectively. Compared with the magnitude relative to RTT (i.e., $[20, 100]$ ms), this shows remarkable recovery latency reduction.

In summary, our analysis demonstrates a strong alignment between the distributions of T_{wait} and N, which provides strong validation for Equation (4). This alignment strongly suggests that an effective reduction in N directly translates to a corresponding decrease in T_{wait}.

D. Mitigating Receiving Buffer Starvation of Throughput-Intensive Transmissions

The throughput-intensive transmission suffers from receiving buffer starvation in the case of a large N_{max}, the maximum retransmission round among all lost packets. To explore how ART tackles this issue, we first investigate how ART reduces N_{max}, and then investigate how ART improves the goodput.

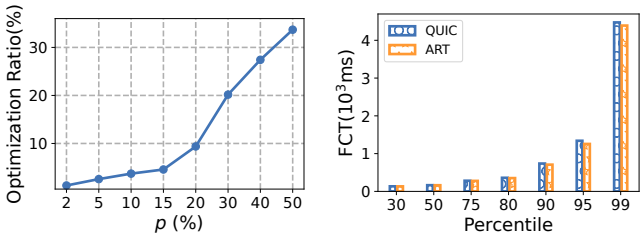
Reducing maximum retransmission round. Figures 15 and 16 illustrate the average value of the maximum retransmission rounds under different network conditions. In Figure 15, when $p = 1\%$, ART's reduction of N_{max} varies across different network environments, ranging from 3% to 16% for the transmission of 200KB-sized files, and from 13% to 16% for the transmission of 1MB-sized files. Similarly, in Figure 16, when $p = 10\%$, ART's reduction of N_{max} varies across different network environments, ranging from 7% to 26% for 200KB-sized files and from 9% to 13% for 1MB-sized files. We can conclude that OR3 is already quite excellent in terms of the maximum retransmission rounds metric. However, our comparison experiments show that there is still room to reduce this metric.

Improving goodput. As shown in Figure 19(a), we give an example of the relationship between N_{max} and goodput, where $bw = 100$ Mbps, $rtt = 300$ ms, $p = 5\%$, and receiving buffer size = 8 MB. The average N_{max} is obtained from 100 runs of each scheme. It is demonstrated that ART greatly increases the goodput (orange line) by decreasing the N_{max} (blue dashed line). As mentioned in §III, a smaller N_{max} alleviates receiving buffer starvation. Figure 19(b) further shows the case of how ART performs under different receiving buffer sizes. It is observed that ART still fills up the pipe when the receiving buffer size is insufficient for QUIC. In particular, ART improves up to 28% of goodput when receiving buffer size = 8 MB. We believe that this can be attributed to ART's efficient loss recovery.

In summary, our analysis demonstrates a strong alignment between goodput and N_{max}, which strongly suggests that an effective reduction in N_{max} directly translates to a corresponding improvement in goodput.

E. The CPU overhead of Using ART

ART is a lightweight packet loss recovery solution. It does not impose a large computational overhead on the CPU. To demonstrate this, we use two machines to act as the sender and the receiver. We set QUIC to run on a single core at the sender side. The CPU utilization of running QUIC, ART, and FEC on the sender side was counted separately with the packet loss rate set to a constant value of 5% without limiting the bandwidth and delay. As shown in Figure 20, the use of ART brings about an additional CPU consumption of 0.18% and FEC brings about an additional consumption of 0.92% compared to the traditional QUIC (without applying any additional means of



(a) Optimization ratio (b) FCT ranges

Fig. 21: Performance in real-world deployment. The optimization ratio is computed as $\frac{FCT_{quic}-FCT_{art}}{FCT_{quic}}$, where FCT_{quic} and FCT_{art} denote the FCT of QUIC and ART, respectively.

packet loss recovery). This reveals that the CPU overhead of using ART is negligible in most cases.

F. Real-world Deployment

To verify that ART indeed accelerates packet recovery speed, we used the actual FCT in the production network as our measurement metric. Particularly, we compute the optimization ratio as $\frac{FCT_{quic}-FCT_{art}}{FCT_{quic}}$, where FCT_{quic} denotes the FCT of QUIC without applying ART and FCT_{art} denotes the FCT of QUIC applying ART.

Setup. We conducted comprehensive testing of two key scenarios: RPC transmission and small file downloading. In the RPC transmission scenario, we analyzed data from 3.3 million QUIC connections spanning an entire day on a 4G Mobile Network. This data provided valuable insights into the performance and efficiency of the RPC requests. For the small file transfer scenario, where each file size was approximately 3MB, we collected real-network data from users located in diverse geographical areas. The data was gathered using the Bonree [25] platform and comprised over 26,000 requests spanning a week. Notably, these clients generated transfer requests at an impressive rate of 100 times per second.

Result. Figure 21(a) presents the average FCT improvement proportion of ART over native QUIC in an RPC transmission scenario under varying packet loss rates. The graph demonstrates that the optimization effect becomes more pronounced as the packet loss rate increases, with improvements reaching approximately 34%. This phenomenon can be attributed to the diminished effectiveness of the loss recovery algorithm at higher packet loss rates, and ART effectively compensates for this deficiency. Figure 21(b) provides a comparison of FCT across different quantiles. Additionally, in the small file downloading scenario, we observed an overall reduction in FCT of about 2%. Our deployment experience further shows that this reduction is especially significant for users with higher access bandwidth.

VIII. Related Work

FEC-based loss recovery. The field of packet loss recovery has seen numerous works related to FEC [6], [26]–[28]. However, most of them can not effectively address the issue of burst packet loss [26]–[28]. Although some state-of-the-art work [6] has provided some solutions to handle burst packet loss, they involve a high implementation complexity and require corresponding modifications on the client side, making them unsuitable for multi-vendor scenarios. In contrast, ART is a lightweight design that only requires server-side (sender-side) modification.

ARQ-based loss recovery. The TCP stack has proposed a series of ARQ-based loss recovery improvements [16], [22], [29]–[34]. Among them, FACK [29], RACK-TLP [16], and TACK [22] have been proven to significantly reduce loss detection time in many cases [35]. In addition to these sender-side loss detection algorithms, QUIC [15] and TCP-TACK [34] share the same idea of detecting loss at the receiver side according to the monotonically increasing packet number. These works, however, mainly rely on rapid loss detection (i.e., optimizing T_{single} in Equation (4)). In contrast, this paper focuses on reducing retransmission round (i.e., optimizing N in Equation (4)), which is also a key influence factor in ARQ-based loss recovery.

Combining FEC and ARQ. There is an increasing trend of combining FEC and ARQ [36], [37]. However, most of them fall into the category of dynamically switching between the two paradigms according to application requirements or network conditions. ART shares the same idea of combining FEC and ARQ but operates in a different way. Particularly, ART applies redundancy to only lost packets instead of unlost packets. There also exist some studies that employ the idea of redundancy, most of which are applied in wireless sensor networks [38]–[41]. Furthermore, their redundancy level is often set to a fixed value [42]. The most related work to ART is OR3 [17], which adopts a rule-based way to decide the redundancy level according to the retransmission round. However, our comparison experiments show that there is still room to reduce the recovery time and redundant traffic due to the dynamics of packet loss in the wild.

IX. Conclusion

Retransmission itself does not hinder slow recovery, it is the loss of retransmission that becomes the real obstacle. Simple but useful, ART is proposed as an ARQ-enhanced scheme that selectively incorporates FEC for lost packets without any modifications to the receiver/client side. The primary goal of ART is to address data reassembling starvation and receiving buffer starvation by considerably reducing the recovery time of lost packets without imposing significant redundancy costs. The real-world deployment experience reaffirms the viability of ART as an advantageous option for CDN vendors seeking to enhance their competitiveness in a diverse and competitive market landscape with multiple suppliers.

Acknowledgment

We thank the anonymous reviewers and our shepherd, Toru Hasegawa, for their valuable comments. We are also grateful for the conversations with and feedback from Wei Li, Jupeng Zhang, Zhi Long, and Kezhi Wang.

REFERENCES

[1] Y. Zhang, K. Xu, H. Wang, Q. Li, T. Li, and X. Cao, "Going fast and fair: Latency optimization for cloud-based service chains," *IEEE Network*, vol. 32, no. 2, pp. 138–143, 2017.

[2] Y. Zhao, K. Xu, H. Wang, B. Li, and R. Jia, "Stability-based analysis and defense against backdoor attacks on edge computing services," *IEEE Network*, vol. 35, no. 1, pp. 163–169, 2021.

[3] B. Wu, T. Li, C. Luo, C. Ouyang, X. Du, and F. Wang, "Autoplex: inter-session multiplexing congestion control for large-scale live video services," in *ACM SIGCOMM Workshop (NetAI)*, 2022, pp. 1–6.

[4] S.-H. Chan, X. Zheng, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Video loss recovery with fec and stream replication," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 370–381, 2006.

[5] F. Michel, Q. De Coninck, and O. Bonaventure, "QUIC-FEC: Bringing the benefits of forward erasure correction to QUIC," in *IEEE IFIP Networking*, 2019, pp. 1–9.

[6] M. Rudow, F. Y. Yan, A. Kumar, G. Ananthanarayanan, M. Ellis, and K. Rashmi, "Tambur: Efficient loss recovery for videoconferencing via streaming codes," in *USENIX NSDI*, 2023, pp. 953–971.

[7] D. BertsekasandR, "Gallager, data networks," *Prentice-Hall*, vol. 1, no. 99, p. 2, 1992.

[8] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Improving performance of TCP over wireless networks," in *IEEE ICDCS*, 1997, pp. 365–373.

[9] D. Barman, I. Matta, E. Altman, and R. El Azouzi, "TCP optimization through FEC, ARQ, and transmission power tradeoffs," in *Springer WWIC*, 2004, pp. 87–98.

[10] J. Chen, W. Tan, L. Liu, X. Hu, and F. Xu, "Towards zero loss for TCP in wireless networks," in *IEEE IPCCC*, 2009, pp. 65–70.

[11] T. Li, K. Zheng, and K. Xu, "Acknowledgment on demand for transport control," *IEEE Internet Computing*, vol. 25, no. 2, pp. 109–115, 2021.

[12] Y. Arda and J.-C. Hennet, "Inventory control in a multi-supplier system," *IJPE*, vol. 104, no. 2, pp. 249–259, 2006.

[13] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, p. 58–66, 2017.

[14] D. Baltrunas, A. Elmokashfi, A. Kvalbein, and Alay, "Investigating packet loss in mobile broadband networks under mobility," in *IFIP Networking*, 2016, pp. 225–233.

[15] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The quic transport protocol: Design and internet-scale deployment," in *ACM SIGCOMM*, 2017, p. 183–196.

[16] Y. Cheng, N. Cardwell, N. Dukkipati, and P. Jha, "The RACK-TLP Loss Detection Algorithm for TCP," RFC 8985, Feb. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8985

[17] H. Xie and T. Li, "Revisiting loss recovery for high-speed transmission," in *IEEE WCNC*, 2022, pp. 1987–1992.

[18] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *USENIX ATC 18*, 2018, pp. 731–743.

[19] L. Tech, "LSQUIC," https://github.com/litespeedtech/lsquic/commit/850b0a3d100b2abe89cf054a9f8d13054fac34a3, 2022.

[20] LitongLab, "QUIC-ART," https://github.com/litonglab/quic-art, 2023.

[21] R. Netravali, A. Sivaraman, K. Winstein, S. Das, A. Goyal, and H. Balakrishnan, "Mahimahi: A lightweight toolkit for reproducible web measurement," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, p. 129–130, 2014.

[22] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, and K. Tan, "Tack: Improving wireless transport performance by taming acknowledgments," in *ACM SIGCOMM*, 2020, pp. 15–30.

[23] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *ANRW*, 2018, p. 19.

[24] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM*, vol. 8, no. 2, pp. 300–304, 1960.

[25] bonree, "Bonree," https://www.bonree.com, 2008.

[26] K. Park and W. Wang, "Afec: an adaptive forward error correction protocol for end-to-end transport of real-time traffic," in *IEEE ICCCN*, 1998, pp. 196–205.

[27] M. Koul and K. Rao, "An n+1 redundant gop based fec algorithm for improving streaming video quality due to packet loss and delay jitter," in *ICET*, 2007, pp. 102–107.

[28] H. Lundqvist and G. Karlsson, "Tcp with end-to-end fec," in *IZS*, 2004, pp. 152–155.

[29] M. Mathis and J. Mahdavi, "Forward acknowledgement: refining tcp congestion control," *Acm Sigcomm Computer Communication Review*, vol. 26, no. 4, pp. 281–291, 1996.

[30] E. Blanton, "Rfc 4653: Improving the robustness of tcp to non-congestion events," *IETF*, 2006.

[31] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig, "Rfc 5827: Early retransmit for tcp and stream control transmission protocol (sctp)," *IETF*, 2010.

[32] P. Hurtig, A. Brunstrom, A. Petlund, and M. Welzl, "Rfc 7765: Tcp and stream control transmission protocol (sctp) rto restart," *IETF*, 2016.

[33] E. Blanton, M. Allman, L. Wang, I. Jarvinen, M. Kojo, and Y. Nishida, "Rfc 6675: A conservative loss recovery algorithm based on selective acknowledgment (sack) for tcp," *IETF*, 2012.

[34] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, and K. Tan, "Revisiting acknowledgment mechanism for transport control: Modeling, analysis, and implementation," *IEEE/ACM TON*, vol. 29, no. 6, pp. 2678–2692, 2021.

[35] E. Blanton, M. Allman, K. Fall, and L. Wang, "Rfc 3517: A conservative selective acknowledgment (sack)-based loss recovery algorithm for tcp," *IETF*, 2003.

[36] F. Michel, A. Cohen, D. Malak, Q. De Coninck, M. Médard, and O. Bonaventure, "Flec: Enhancing quic with application-tailored reliability mechanisms," *IEEE/ACM TON*, vol. 31, no. 2, pp. 606–619, 2023.

[37] T. Porter and X.-H. Peng, "Effective video content distribution by combining tcp with adaptive fec coding," in *IEEE BMSB*, 2010, pp. 1–5.

[38] H. Wen, C. Lin, F. Ren, Y. Yue, and X. Huang, "Retransmission or redundancy: Transmission reliability in wireless sensor networks," in *IEEE MASS*, 2007, pp. 1–7.

[39] H. Wen, C. Lin, F. Ren, H. Yang, T. He, and E. Dutkiewicz, "Joint adaptive redundancy and partial retransmission for reliable transmission in wireless sensor networks," in *IEEE IPCCC*, 2008, pp. 303–310.

[40] T. Li, J. Liang, D. Wang, Y. Ding, K. Zheng, X. Zhang, and K. Xu, "On design and performance of offline finding network," in *IEEE INFOCOM*, 2023, pp. 1–10.

[41] Y. Ding, T. Li, J. Liang, and D. Wang, "Blender: Toward practical simulation framework for ble neighbor discovery," in *ACM MSWiM*, 2022, pp. 103–110.

[42] J. Zhu and S. Roy, "An adaptive two-copy delayed sr-arq for satellite channels with shadowing," in *IEEE VTC*, 2002, pp. 849–853.