QRST: A QUIC-Enabled Robust Streaming Transmission Framework for Ultra Large-Scale LEO Satellite Networks

Mengyang Zhang, *Student Member, IEEE*, Zitian Zhang, *Senior Member, IEEE*, Ting Ma, *Member, IEEE*, Xiaoyu Liu, *Student Member, IEEE*, Jinqiang Chen, Sheng Fang, Li Zhang, and Haibo Zhou *Senior Member, IEEE*

***Abstract*—With the development of Low Earth Orbit (LEO) satellites, Ultra Large-Scale LEO Satellite Networks (ULSLSNs) hold immense potential to provide high-speed and reliable services in future communication systems. It offers substantial promise to meet emerging Internet traffic demands, such as remote real-time applications with stringent delay constraints (deadline). However, the complexity of ULSLSNs is significantly amplified by the satellite mobility, limited bandwidth, and more packet losses. The complex and dynamic network can greatly increase block transmission latency and may further degrade user's QoE for real-time applications. To reduce block transmission latency and deliver more blocks before deadline for real-time applications, we propose a QUIC-enabled Robust Streaming Transmission (QRST) framework deployed in ULSLSNs. This framework comprises four components and mainly involves an adaptive Forward Erasure Coding (FEC) scheme and a deadline-driven block scheduler. The FEC scheme dynamically allocates redundancy based on current network conditions to reduce extra retransmission delay and balance bandwidth overhead. The block scheduler selects blocks for transmission to deliver more blocks before deadline, especially for high-priority blocks. Finally, we implement QRST over the network of Kupier K3 Shell simulated by NS-3 and living video streaming applications are transmitted. The experiment results show that QRST can significantly enhance the transmission performance of video streaming applications in ULSLSNs compared with other mechanisms.**

***Index Terms*—Ultra Large-Scale LEO Satellite Networks, QUIC, forward erasure correction, scheduler, real-time applications.**

I. Introduction

Nowadays, with the rapid increasing in user-equipment and advancement of emerging applications, existing terrestrial networks face significant pressures in efficiently serving growing traffic demands [1]. Especially for real-time applications, it has stringent delay constraints, which means a block may become useless if it's received after deadline [2]. Due to limited capacity, long distance and restricted coverage, it's challenging for terrestrial networks to guarantee stable delay requirements in remote/non-terrestrial areas [3], [4], [5]. Fortunately, with the developments of Low Earth Orbit (LEO) satellite technologies, the Ultra Large-Scale LEO Satellite Networks (ULSLSNs) are expected to provide seamless coverage, high-speed data rate and low latency [6], [7] such as Kupier [8], Telesat [9] and Starlink [10]. It's being considered as a promising solution to meet the increasing demands of modern communication systems and is anticipated to become a predictable and achievable paradigm in future communication networks [11]. However, some challenges are also posed for application transmission in ULSLSNs due to weather interference, antenna attenuation and satellite mobility, etc [12]. Specifically, data packets are more possible to experience losses compared with terrestrial networks, which is caused by atmospheric interference, weather and attenuation, etc [13]. And with the high-speed movement of LEO satellites, the distance and visibility between satellites and grounds are changing dynamically [14], [15], resulting in variable end-to-end latency. For real-time applications, packet losses and unstable network environments can greatly increase block transmission latency, and blocks missing deadline can further degrade user's Quality of Experience (QoE) [16].

To ensure better transmission performance, transport layer protocol serves as an important part to provide reliable and low-latency services. Quick UDP Internet Connection (QUIC) is a User Datagram Protocol (UDP)-based transport protocol proposed by Google in 2013 [17]. QUIC can provide the services which combine the functionality of Hypertext Transfer Protocol (HTTP), Transmission Control Protocol (TCP) and Transport Layer Security (TLS), built on top of UDP. It is mainly designed to reduce establishment and transmission latency, enabling faster data exchange. And with the characteristics like stream multiplexing, data encryption and reliability, QUIC is flexible to handle diverse application demands and accounts for increasing traffic for global Internet. It holds great promise to support rapid and reliable data transmission in LEO satellite networks [18]. However, the retransmission mechanisms of QUIC protocol can introduce much extra latency for block transmission when more packet losses occur in ULSLSNs, which is undesirable for real-time applications. Forward Error Correction (FEC) [19] is an effective technique to handle packet losses. It can recover lost packets by sending redundant data without the need of additional retransmission, which is highly deployed in high-error and delay-constrained scenarios. Due to the benefits of FEC, IETF are ongoing discussing to integrate FEC technique into QUIC protocol to

Mengyang Zhang, Xiaoyu Liu and Haibo Zhou (Corresponding author) are with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China. Email: {Kolimn_Zhang, xyliu0119}@163.com; haibozhou@nju.edu.cn.

Zitian Zhang is with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China. Email: zitian.zhang@mail.zjgsu.edu.cn.

Ting Ma is with the School of Electronic and Optical Engineering, Nanjing University Of Science and Technology, Nanjing 210094, China. Email: tingma@njust.edu.cn.

Jinqiang Chen, Sheng Fang and Li Zhang is with the Huawei Technologies Co., Ltd. Email: {chenjinqiang, fangsheng, zhangli344}@huawei.com.

enhance its performance over the scenarios with packet losses [20]. Although some good results of QUIC with FEC are presented, the inappropriate amount of redundant data can lead to bandwidth wastage and significantly increase block delivery time. Moreover, QUIC schedules packets by the principle of First-In-First-Out (FIFO), without considering the block properties in real-time applications, such as priority, delay constraints. The block completion ratio and block priority can significantly impact the user's QoE, which should be taken into account specially. As a result, it's still challenging to satisfy the transmission requirements of real-time applications in ULSLSNs.

In this paper, we propose a QUIC-enabled Robust Streaming Transmission (QRST) framework to enhance blocks transmission performance and improve user's QoE for real-time applications in ULSLSNs. We consider the scenario as shown in Fig. 1, living video streaming is transmitted between two ground stations via ULSLSNs. The QRST framework is mainly deployed on sender side, which involves two key modules, an adaptive FEC scheme and a deadline-driven block scheduler. On sender side, the video frame is selected from sender buffer by the deadline-driven block scheduler and subsequently is encoded by the adaptive FEC scheme to transmit via ULSLSNs. The reverse process occurs on receiver side. To be specific, the adaptive FEC scheme dynamically allocates redundant packets (Repair Symbols) for stream packets (Source Symbols) based on network conditions to reduce block transmission latency and save bandwidth overhead at the same time. When with higher loss rate, more Repair Symbols are generated to recover packet losses. Otherwise, fewer Repair Symbols are allocated to avoid excessive bandwidth utilization. Furthermore, to improve the block completion ratio before deadline for real-time application, the deadline-driven block scheduler arranges the blocks order considering the properties of deadline, priority and block size. It aims to prioritize to send high-priority and urgent-deadline blocks and delay the transmission of blocks which are unlikely to be delivered before deadline. Finally, we integrate these framework components into QUIC protocol and implement it over Kupier K3 shell network simulated by NS-3. The main contributions are summarized as follows:

- We propose an adaptive FEC scheme, this mechanism aims to allocate appropriate redundancy to reduce block transmission latency and balance bandwidth overhead at the same time according to current network conditions.
- Furthermore, a deadline-driven block scheduler is presented to improve the block completion ratio and user's QoE for real-time applications. The scheduler arranges the blocks sending order considering their deadline, priority and size. It prioritizes to send the blocks with higher priority and try to meet the deadline requirements for more blocks .
- We integrate these components into QUIC protocol and implement it over Kupier K3 shell network simulated by NS-3. We examine the performance of QRST under different network configurations with living video streaming application.

The remaining of this paper is organized as follows: Section II discusses the related work. Section III presents the architecture and algorithms of our framework. Section IV shows the details of the implementation for our ideas. Section V conducts the evaluations to examine the performance of QRST and Section VI concludes the paper.

II. Related Works

A. FEC Techniques in QUIC

The standard QUIC protocol defined by IETF [21] uses two thresholds to detect the packet losses: time threshold and packets threshold. If a packet remains unacknowledged over a period of time before the newly acknowledged packet or its sequence number is less than a certain amount than the newly acknowledged packet, it is considered to be lost. Once the packet is detected as lost, QUIC will retransmit this packet with a new sequence number and it takes at least an extra RTT to be received, which can bring much extra latency when more losses occur [22]. With the benefits of FEC technique, Google integrates a XOR-based FEC method in early versions of QUIC [23]. With the XOR-based FEC coding, sender can deliver one more Repair Symbols for the same coding window to protect Source Symbols from losses. However, it can only recover a single lost Source Symbol within that window and cannot recover any one if multiple symbols are lost. QUIC-FEC [24] proposes an extension to support several FEC techniques in QUIC, such as XOR, Reed-Solomon and Convolutional RLC error-correcting codes. It shows great results in small files transfers but may be harmful to long data transfers. Yu *et al.* [25] proposes a proactive streaming coding mechanism to support low-latency transmission in non-terrestrial networks by FEC technique. This mechanism adjusts the FEC redundancy to make it close to the measured loss rate and evaluate it in Geostationary Earth Orbit (GEO) satellite networks. rQUIC [26] presents an adaptive mechanism to adjust the number of Repair Symbols based on channel conditions. It defines a parameter through the ratio of retransmission packets number and acknowledged packets number to represent the channel conditions. However, this mechanism is a gradual process and may not adapt to dynamic LEO satellite networks. Furthermore, FLEC [27] presents a innovative reliability mechanism both considering the channel conditions and application requirements. It enables applications to select the appropriate reliability mechanism based on diverse requirements for better transmission performance. This mechanism is highly inspiring and can accommodate different demands for good results. However, the properties of real-time applications such as block size, priority are not the focal points in FLEC. And the LEO satellite environments are not fully considered. In our early work, a framework named QRVTS [28] is presented which adjusts redundancy ratio based on residual loss rate with the model of Bernoulli process. This framework is effective for the reduction of transmission latency but lack the optimization for overall performance.

B. Block Scheduler

Since the blocks missing deadline significantly impact user's QoE for real-time applications, many efforts are focused on the scheduling methods to deliver more blocks before deadline. Shi *et al.* [2] proposes a deadline-aware transport protocol

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3540322

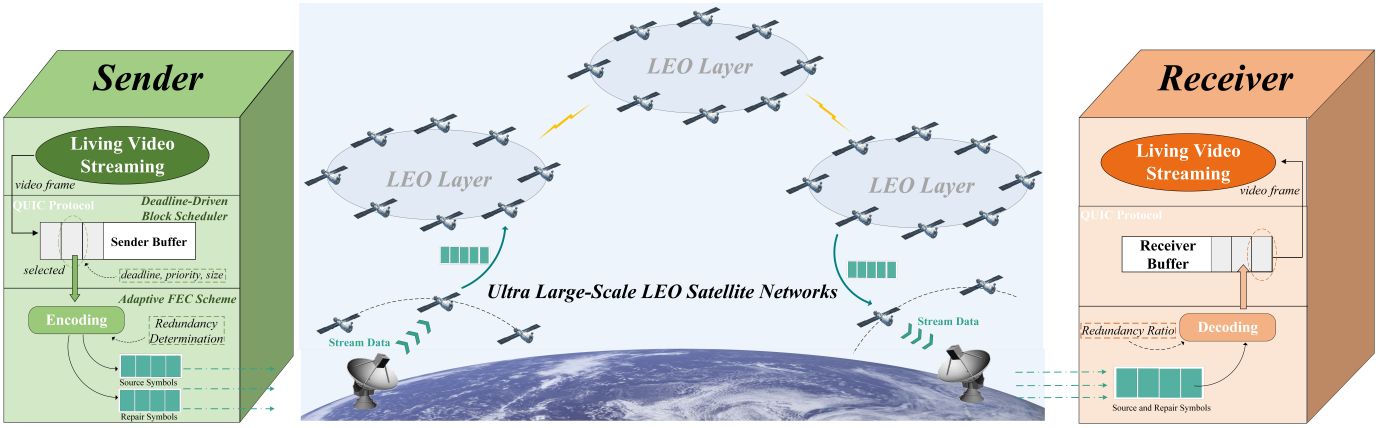

Fig. 1. Living Video Streaming Transmission in Ultra Large-Scale LEO Satellite Networks.

called DTP implemented in QUIC. It recognizes the delay constraint of blocks specified by applications and attempts to ensure more timely delivery. Tetris [29] proposes a near-optimal scheduling algorithm based on multipath deadline-aware transport protocol. It divides different blocks to diverse tasks and schedules them by the comprehensive priority. Zhang *et al.* [30] defines a cost of path considering RTT and packets queuing delay. It chooses the path with minimum cost to transmit packets under the demands of real time video streaming. PATON [31] presents a priority-aware frame selection algorithm to minimize the sum of total distortion under bandwidth and delay constraints for high-definition mobile video delivery. MP-DASH [32] takes the user preferences, video chunk size and deadline into consideration, and aims to determine the best fetch strategy over multipath with the delay constraints. DAMS [33] presents a deadline-aware multipath transmission protocol to arrange the sending order of blocks based on their properties and scheduled them into multiple paths to reduce the waste of bandwidth and improve block completion ratio before deadline. Although there are many excellent works related to blocks scheduling, they mainly focus on multipath scheduling or are not targeted for the adaptability of LEO satellite networks. The performance of these mechanisms can be degraded due to the dynamic round-trip time, higher loss rate and limited bandwidth. To this end, we propose a QUIC-enabled Robust Streaming Transmission (QRST) framework that joint exploits the adaptive FEC scheme and the deadline-driven block scheduler to optimize the blocks transmission performance and improve user's QoE for real-time applications in ULSLSNs.

III. QRST Architecture and Design

QRST aims to ensure timely block delivery and improve the completion ratio, enhancing the poor transmission performance in USLSNs for real-time applications. It divides the functionality into two parts: using FEC coding to reduce the impact of packet losses on block transmission latency, and employing block scheduling to deliver more blocks before deadline. By integrating these mechanisms, QRST can enhance the overall transmission performance for real-time applications. The QRST framework is mainly deployed on the sender side. As illustrated in Fig. 2, this framework comprises four components: the deadline-driven block scheduler, adaptive FEC scheme, transmission module and feedback module. On the sender side, QRST begins with delivering blocks to sender buffer by applications. Subsequently, the block scheduler selects the block for transmission considering multiple block properties to meet deadline requirements and improve user's QoE. Once block scheduling is completed, the adaptive FEC scheme dynamically introduces Repair Symbols based on current network conditions to enhance the data protection. Then the block data is encoded into Source Symbols and encapsulated within QUIC streams for transmission over the network by transmission module. Furthermore, the feedback module appends network condition details to the other components through the acknowledgement packets. On the receiver side, the process mainly accounts for packets decoding and sending ACK packets. Due to space limitations, the description of this process is omitted. In following contexts, we will provide more details of the four components.

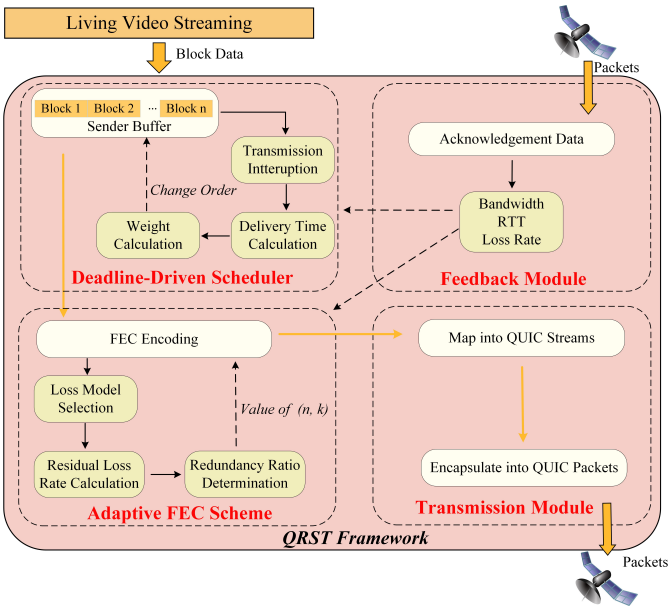

Fig. 2. The Framework and Working Components of QRST.

A. Transmission and Feedback Modules

To enhance the performance of blocks transmission, a rational approach is required to provide feedback on network

conditions. On the sender side, when receiving acknowledgement packets from receiver, information such as bandwidth, RTT and loss rate are fed back into next round decision-making process. The bandwidth and RTT are linked to blocks scheduling, while packet losses information is utilized for redundancy determination. Moreover, to bolster the extensibility of video transmission, we map one or more blocks into a QUIC stream by transmission module. It means a QUIC stream can accommodate the data of multiple blocks, which can brings much adaptability for the scheduler. Simultaneously, in order to facilitate the management of application data transmission, we align the number of streams with the number of blocks. Several QUIC streams may serve the same incomplete block and the block number is transmitted to receiver encapsulated in QUIC packets.

B. Deadline-Driven Block Scheduler

In QUIC protocol, the default scheduler follows a FIFO approach, prioritizing the transmission of blocks based on arriving order. However, due to the limited bandwidth resources and dynamic network, it is worth considering an enhanced block scheduler to ensure more blocks delivered before deadline for real-time applications. In this subsection, we propose a deadline-driven block scheduler that aims to improve the block completion ratio before deadline and prioritize the transmission of high-priority blocks. The scheduler arranges the order of blocks based on their size, deadline and priority. The details are summarized in Algorithm 1 and the working steps are as follow.

The scheduler starts with the decision of whether to interrupt the transmission of current block scheduled in last round. Specifically, if the data of current block has not been fully transmitted and the remaining data is considered to be not able to be delivered before deadline, the scheduler may interrupt and delay the transmission of current block. Because transmitting expired data would unnecessarily waste bandwidth and delay the sending of other blocks. As shown in Algorithm 1, the function $Delayed(block_t)$ return true when the remaining data of the $block_t$ is deemed to exceed the deadline. However, in order to utilize bandwidth efficiently, we are hesitant to cancel the transmission of a block easily. Therefore, we intentionally underestimate remaining delivery time in this function to prevent degradation caused by inaccurate estimation.

If the transmission of last-round block is delayed or it has been fully transmitted, the scheduler enters the blocks selection step. To ensure timely delivery, only blocks that are anticipated to be delivered before deadline are considered in this process. To determine which block is eligible for selection, function $CheckArriveOnTime(block_t)$ is to access whether a given block can complete transmission before deadline based on current network conditions. The detailed calculation for this function is shown as follow. The available time for $block_t$ is calculated by Equation (1), where now and $O\hat{W}D$ represent the current time and one-way delay, respectively. The $unsent\_size$ is the remaining data size of $block_t$ and $B\hat{W}$ is the estimation of bandwidth. We get these parameters from feedback module as mentioned before. Besides, we add

Algorithm 1 Deadline-Driven Block Scheduling

Input: $blocks$, the blocks in sender buffer
Input: $block_l$, the scheduled block in last round
Input: $size, prio, ddl$, unsent size, priority and deadline of a block
Input: $Delayed(block_t)$, return true if delaying the transmission of $block_t$
Input: $CheckArrivOnTime(block_t)$, return true if $block_t$ can be delivered before deadline
Input: $GetMinExpire(blocks)$, return the most recently arrived block of $blocks$
Input: $CompreSche(blocks)$, return comprehensive scheduling results as illustrated in Algorithm 2
Output: $block$, the block to be scheduled for transmission

if $remaing\ size\ of\ block_l\ != 0$ **then**
  if $Delayed(block_l) = false$ **then**
    $return\ block_l$
  end if
end if
$blocks' = blocks$
for $block\ in\ blocks'$ **do**
  if $CheckArrivOnTime(block) = false$ **then**
    $remove\ block\ from\ blocks'$
  end if
end for
if $blocks' = NULL$ **then**
  $return\ GetMinExpire(blocks)$
end if
return $CompreSche(blocks')$

a safeguard for deadline to reduce the influence of dynamic network conditions as shown in Equation (2). The Δt is used as the half of median deviation of RTT and γ is used to describe the aggressiveness of this safeguard according to diverse user demands. If the available time is less than 0, it is inferred that this block cannot complete transmission before deadline and is consequently removed from the scheduling queue. If all the blocks are removed from scheduling queue, it does mean no block can complete transmission before deadline at current time, so we select the most recently arrived block from sender buffer, which is most likely to meet the deadline requirements. Otherwise, we start to determine which block to be sent from scheduling queue.

$$avail\_time = \overline{Deadline} - now - O\hat{W}D - \frac{unsent\_size}{B\hat{W}}. \quad (1)$$

$$\overline{Deadline} = Deadline - \gamma \times \Delta t. \quad (2)$$

In this circumstance, the block scheduler performs comprehensive scheduling by considering the properties of priority, deadline and size as shown in Algorithm 2. Given a block of priority $prio$ and available time $avail\_time$, the final weight is obtained by Equation (3).

$$weight = \frac{1}{\alpha \times \frac{avail\_time}{deadline} + \beta \times prio}. \quad (3)$$

The parameters α and β can be determined according to application demands and user preferences. Obviously, the block with higher priority and more urgent deadline is prioritized to transmit. However, considering the importance of block priority, if the block with highest weight is of low priority, we will examine whether it will impact the completion of other high-priority blocks in scheduling queue if sending it at current time. Specifically, all high-priority blocks are extracted from the scheduling queue to check whether they can be delivered before deadline after completing the transmission of the block with highest weight. If the answer is negative, the scheduler will discard this block and select another from scheduling queue again with the same method shown in Algorithm 2, until the highest weight block is high-priority or its transmission has no impact on the receiving of other high-priority blocks. The final weight is obtained by function $GetPrioBlock(blocks)$ with Equation (3).

Algorithm 2 Comprehensive Block Scheduling

Input: $blocks$, the blocks from Algorithm 1
Input: $size, prio, ddl$, unsent size, priority and deadline of a block
Input: $GetPrioBlock(blocks)$, return the highest weight block from $blocks$
Input: $CheckArrivOnTime(block_t)$, return true if $block_t$ can be delivered before deadline
Input: $CheckOnTime(block_t, block)$, return true if $block_t$ can be delivered before deadline after completing the transmission of $block$
Output: $block$, the block to be sent for transmission

$stop = 0$
while $!stop$ **do**
  $stop = 1$
  $block = GetPrioBlock(blocks)$
  if $priority\ of\ block = high$ **then**
    return $block$
  end if
  for $block_t\ in\ blocks$ **do**
    if $priority\ of\ block_t = high$ **then**
      if $CheckOnTime(block_t, block) = 0$ **then**
        remove $block$ from $blocks$
        $stop = 0$
      end if
    end if
  end for
end while
return $block$

C. Adaptive FEC Scheme

As mentioned in Section II, coding in QUIC protocol does bring great benefits for data transmission when more packet losses occur. But it is challenging to determine the trade-off between bandwidth overhead and reliability especially in dynamic network environments. On one side, increasing the number of Repair Symbols can protect the Source Symbols from losses and prevent the extra delay brought by retransmission at the expense of bandwidth. On the other side, fewer Repair Symbols account for less bandwidth utilization but cannot recover any Source Symbol at receiver if more packet lose than anticipated, resulting in bandwidth wasting. Therefore, we introduce the residual loss rate, referring to the packet loss rate after FEC protection, as a mean to control FEC redundancy and generate just enough Repair Symbols to ensure the residual loss rate less than specific threshold. In this case, our goal is not to theoretically recover all lost symbols on the receiver side, as excessive protection would greatly consume bandwidth. We are inspired by the fact that, for example, the global loss rate is 10%, indicating one symbol lose among every ten symbols on average. However, adding one Repair Symbol after nine Source Symbols generally cannot recover any one in the same coding window, as the residual loss rate for this case is usually unignorable and more symbols can be lost than anticipated. Therefore, we aim to control the residual loss rate just below a specified threshold in order to achieve the desired trade-off between bandwidth overhead and reliability. As a result, we can allocate appropriate number of Repair Symbols to recover the lost packets and reduce retransmission time in different scenarios.

Firstly, we give the analysis and calculation of residual loss rate with different loss models deployed in our framework. In QRST, we treat each packet as a Source Symbol or a Repair Symbol and consider the packet loss models of Bernoulli process and Gilbert loss model, to describe random losses and burst losses, respectively.

The Bernoulli process is a statistical model used to describe binary classification problems. It assumes that each loss can be viewed as a random variable independently sampled from a Bernoulli distribution. Given the packet loss rate p, the probability for m packets lost among n packets is:

$$Pr(N_{lost} = m) = \binom{n}{m} p^m (1-p)^{n-m}. \tag{4}$$

In the context of a FEC block consisting of k Source Symbols and $n-k$ Repair Symbols, if more than $n-k$ Source Symbols are lost in the same window there are no symbols can be recovered. And the probability for this case can be derived from Equation (5) and the residual loss rate can be calculated by Equation (6).

$$Pr(N_{lost} > n-k) = \sum_{m=n-k+1}^{n} \binom{n}{m} p^m (1-p)^{n-m}. \tag{5}$$

$$P_{res}(p) = \frac{1}{n} \sum_{m=n-k+1}^{n} m \binom{n}{m} p^m (1-p)^{n-m}. \tag{6}$$

The Gilbert loss model is a two-state Markov chain used to characterize burst packet losses as shown in Fig. 3 [34]. In this model, the good state represents a successful reception of a packet, while the bad state represents a packet loss at the receiver side.

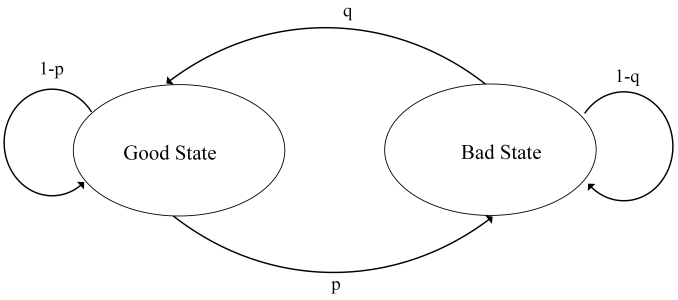

Fig. 3. Gilbert Loss Model.

In this figure, p is denoted as the probability from good to bad state and q is denoted as the probability from bad to good state. A simple trace estimation for p and q are [35]:

$$\hat{p} = \frac{n_{01}}{n_0}. \tag{7}$$

$$\hat{q} = \frac{n_{10}}{n_1}. \tag{8}$$

where n_{01} is the number of packet losses after a good reception and n_{10} is the number of good receptions after a packet loss in the trace window. We further assume the loss process matches a renewal error process. We use $P(n) = Pr(0^{n-1}|1)$ to denote the probability that at least consecutive $n-1$ packet losses after a successful reception and $p(n) = Pr(0^{n-1}1|1)$ is the probability that consecutive $n-1$ packet losses between two successful receptions, where 1 represents a packet loss and 0 represents a successful reception. The probability is calculated by Equation (9) and Equation (10), respectively.

$$P(i) = \begin{cases} 1, & \text{if } i = 1 \\ q(1-p)^{i-2}. & \text{otherwise} \end{cases} \tag{9}$$

$$p(i) = \begin{cases} 1-q, & \text{if } i = 1 \\ q(1-p)^{i-2}p. & \text{otherwise} \end{cases} \tag{10}$$

So the probability $R(m,n)$ that $m-1$ packet losses in the next $n-1$ packets after a bad state is:

$$R(m,n) = \begin{cases} P(n), & \text{if } m=1 \text{ and } n \geq 1 \\ \sum_{i=1}^{n-m+1} p(i)R(m-1,n-i). & \text{if } \ 2 \leq m \leq n \end{cases} \tag{11}$$

Similarly, we use $Q(n) = Pr(1^{n-1}|0)$ and $q(n) = Pr(1^{n-1}0|0)$ to denote the probability that at least $n-1$ consecutive packets received successfully after a packet loss and consecutive $n-1$ packets received successfully between two packet losses, calculated by Equation (12) and Equation (13). And the probability $S(m,n)$ that $m-1$ packets are successfully received in the next $n-1$ packets can be obtained by Equation (14).

$$Q(i) = \begin{cases} 1, & \text{if } i = 1 \\ p(1-q)^{i-2}. & \text{otherwise} \end{cases} \tag{12}$$

$$q(i) = \begin{cases} 1-p, & \text{if } i = 1 \\ p(1-q)^{i-2}q. & \text{otherwise} \end{cases} \tag{13}$$

$$S(m,n) = \begin{cases} Q(n), & \text{if } m=1 \text{ and } n \geq 1 \\ \sum_{i=1}^{n-m+1} q(i)S(m-1,n-i). & \text{if } \ 2 \leq m \leq n \end{cases} \tag{14}$$

So the residual loss rate after FEC protection can be obtained, the state (good state or bad state) of last packet drives to the next process differently. And the residual loss rate P_{res} can be calculated by

$$\begin{aligned} P_{res} = & \frac{\pi}{k} \sum_{i=1}^{k} iR(i,k) \sum_{j=\max(n-k+1-i,0)}^{n-k} R(j+1, n-k+1) \\ & + \frac{1-\pi}{k} \sum_{i=1}^{k-1} (k-i)S(i,k) \sum_{j=0}^{k-1-i} S(j+1, n-k+1). \end{aligned} \tag{15}$$

where π represents the global packet loss rate and it is derived from

$$\pi = \frac{p}{p+q}. \tag{16}$$

and the average burst length is

$$len = \frac{1}{q}. \tag{17}$$

Therefore, we get the residual loss rate in different loss models. We utilize the Bernoulli process in the case of random packet losses and utilize the Gilbert loss model in the case of burst losses. The adaptive FEC scheme is summarized in Algorithm 3. We determine the FEC redundancy based on the threshold of residual loss rate. This threshold is associated with the global loss rate, which is set to $\eta \cdot \pi$. π is the global loss rate and η is a user-defined factor between 0 and 1, which can change dynamically along with network conditions. Subsequently, we calculate the residual loss rate according to Bernoulli process or Gilbert loss model under different situations. The value of (n,k) is to determine the coding rate and we carefully select it to ensure the residual loss rate is slightly below the threshold, avoiding unnecessary redundancy. The parameters n_1 and n_2 are to limit the value range of n and they can be specified by the network conditions or application demands. And if Gilbert loss model is used, we need to promise the redundancy length is greater than the burst loss length as shown in Equation (17).

IV. Implementation

The key components of QRST framework to implement are the deadline-driven block scheduler and the adaptive FEC scheme. We implement the two parts based on QUIC default scheduler and generic FEC technique. To reconstruct it more easily, it is more ideal to extract these two modules from the original code separately, ensuring minimal redundancy with the underlying code. Therefore, we have decided to implement QRST by PQUIC [36], a flexible framework that allows for dynamic exchange of protocol plugins between clients and servers. This framework enables us to extend the protocol on a per-connection basis and modify protocol functions (e.g, FEC coding, block scheduling) through protocol plugins. We

Algorithm 3 Adaptive FEC Algorithm

Input: π, global loss rate
Input: $update\_interval$, the updating interval
Input: $model$, Bernoulli process or Gilbert loss model
Input: $CalRes(n_i, k_i, model, \pi)$ get residual loss rate with $model$
Output: value of (n, k)

if $time > update\_interval$ **then**
  $Res\_t = \eta \cdot \pi$
  Initialize $n = 2, k = 1$
  $n_i = n_1, k_i = \lceil \frac{n_i}{2} \rceil$
  while $n_1 \leq n_i \leq n_2$ **do**
    while $\lceil \frac{n_i}{2} \rceil \leq k_i \leq n_i$ **do**
      $res = CalRes(n_i, k_i, model, \pi)$
      if $res \leq Res\_t$ and $\frac{k_i}{n_i} > \frac{k}{n}$ **then**
        $n = n_i, k = k_i$
      end if
      $k_i = k_i + 1$
    end while
    $n_i = n_i + 1$
  end while
  if $model = Gilbert$ and $n - k < \frac{1}{q}$ **then**
    $n = \lceil \frac{\frac{1}{q}}{n-k} \cdot n \rceil$
    $k = \lceil \frac{\frac{1}{q}}{n-k} \cdot k \rceil$
  end if
end if

implement the behavior of the key components by redefining fec plugin and the scheduling functions. The scheduling functions are mainly to read the block data from applications and arrange them into a specific sending order. Based on that, we construct a $real-time\ scheduling$ plugin used to better handle the scheduling requirements for real-time applications. And for the implementation of FEC, Random Linear Codes (RLC) is employed in our framework. RLC is not only easy to implement or modify but also enables the recovery of multiple packets within the same window. Furthermore, in the initial stage, we adopt a mapping strategy where each block is assigned to a specific stream in QUIC. When the transmission begins, if there is an interruption for a block, we append the data of another block to the same stream associated with the interrupted block. To ensure an equal number of streams and blocks for simplified management as mentioned above, we create additional streams for each interrupted block, facilitating the transmission of the remaining block data.

Finally, to simplify the experiments and get more generalized conclusions, we only use the Gilbert loss model in our simulations. While the Bernoulli process is also implemented to handle random packet losses.

V. Evaluation

A. Experiment Setup

1) Network Simulation: To involve QRST implementation into LEO satellite constellation, we utilize the NS-3 discrete-event network simulator [37] with the Direct Code Execution (DCE) module [38] for our experiments. The NS-3 simulator allows us to build the LEO satellite networks and the DCE module enables the execution of the actual QRST code within NS-3. In order to accurately construct a real LEO constellation, we employ the Kupier K3 Shell network within NS-3 simulator. Kuiper constellation is a project initiated by Amazon and aims to deploy a large satellite Internet and provide low-latency broadband Internet connectivity. The Kupier constellation configurations are shown in Table I. The adjacency and positions for satellites and ground stations are calculated according to the network configuration. And the packet losses are added into each data link with the pattern of burst loss by using the function of NS-3.

Shell	*Height (km)*	*Orbits*	*Sats/Orbit*	*Inclination*
K1	630	34	34	51.9°
K2	610	36	36	42°
K3	590	28	28	33°

TABLE I: Shell Configurations for Kupier Constellation

2) Video Trace: We select the living video streaming as the real-time application. In order to simulate the delay constraint of video blocks, we parse the video into frames by FFmpeg library. The video frames have three different types: I, P, B, where I frames are assigned with high priority and B,P frames are assigned with low priority. Each frame is treated as a block and the deadline is set to 350ms. The frame rate is 30FPS.

3) Metrics: To obtain more convincing results, we present the performance metrics from different perspectives. We evaluate QRST and other schemes with the same source video trace and propose to compute the all frames completion ratio and high-priority frames completion ratio before deadline to assess the overall transmission performance. Moreover, the average bitrate and rebuffering time are also taken into account to reflect the user's QoE. Noticeably, only frames delivered before deadline are considered as the effective data. As a result, the average bitrate is obtained by $\frac{the\ amount\ of\ effective\ data}{video\ length}$ and the rebuffering time is $\frac{missing-deadline\ block\ number}{frame\ rate}$. We also consider the delivery time and overhead as the metrics to evaluate the benefits of adaptive FEC scheme. For delivery time, this duration is recorded from the start of frame transmission to the time when it is received entirely. And overhead refers to the percentage of redundant data compared to all transmitted data.

4) Reference Schemes:

- QUIC. The regular QUIC protocol is implemented by PQUIC. It incorporates the techniques of fast retransmission or Retransmission Timeout (RTO) to handle packet losses and the default FIFO block scheduler to determine blocks sending order.
- QUIC-FEC (FEC). The Random Linear Codes is integrated into QUIC protocol as a reference. For the FEC coding, the redundancy is selected as 12.5% ($n = 8, k = 7$) and 25% ($n = 8, k = 6$) to match low redundancy and high redundancy. And they are represented by FEC (8,7) and FEC (8,6), respectively.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3540322

- Deadline-First Scheduler (DDL) [27]. The DDL scheduler selects the non-expired frame with the closest deadline for transmission.
- Deadline-First Scheduler with FEC (DDLF). The DDLF scheduler combines the DDL scheduler and FEC mechanism simultaneously. It operates by selecting the nearest non-expired block for transmission and encoding it at a fixed redundancy ratio.

B. Deep Analysis of QRST

In order to better explore the performance of QRST, we first dissect it into two independent components: the Adaptive FEC Scheme (AFS) and the Deadline-Driven Block Scheduler (DDBS) and examine them individually to observe their effectiveness on video transmission. The bandwidth is set to 8Mbps in this subsection.

Fig. 4 shows the delivery time of video frames under different loss conditions. The evaluation results are presented with the loss rate of data link ranging from 0.1% to 0.6%, covering both light and heavy loss conditions. AFS achieves the lowest delivery time in all emulation scenarios due to the ability that FEC redundancy can be adjusted dynamically based on different network conditions. This strategy can greatly reduce block transmission delay without causing excessive bandwidth wastage in different scenarios. Besides, for the other FEC schemes, they cannot always outperform QUIC in all conditions and the performance greatly depends on the packet loss rate of data link. The reason is that, in some conditions, fixed redundancy can appropriately accommodate packet losses to recover packets without excessively bandwidth occupying. However, if too much redundancy is introduced or the redundancy is insufficient to recover packets in other conditions, bandwidth can be significantly wasted and it will severely increase the delivery time of video frames as shown in Fig. 4. As we can see, for the 0.1% loss rate of data link, FEC (8,7) gets better performance due to the appropriate redundancy compared with QUIC. However, when loss rate increases, FEC (8,7) sends insufficient redundancy and fails in recovering, which results in bandwidth wastage and generally degrades the performance compared with QUIC protocol. For FEC (8,6), it can effectively recover packet losses and reduce retransmission time with more introduced redundancy. As the loss rate increases, this advantage becomes more obvious due to the less bandwidth wasting. However, in situations of heavier packet losses, there may not be enough capacity to compensate for lost packets even with the 25% redundancy, leading to a degradation in performance.

Fig. 5 illustrates the redundancy overhead for FEC (8,6), FEC (8,7) and AFS. It is evident that AFS demonstrates the ability to dynamically adjust coding rate according to different network environments, the redundancy overhead is particularly related with network conditions. Moreover, it is notable that the redundancy ratio for FEC(8,6) or FEC(8,7) remains relatively stable, lacking the ability to adapt to dynamic networks, occasionally resulting in performance degradation. Besides, as shown in Fig. 5, in case of heavier packet losses, AFS can introduce nearly 30% overhead for redundant data which is even more than that of FEC(8,6). Although it partially enhances the performance in terms of frame delivery time, it also introduces a considerable amount of redundancy, which may significantly affects the delivery of other video frames. This also underscores the necessity for a global block scheduler to manage overall frames transmission.

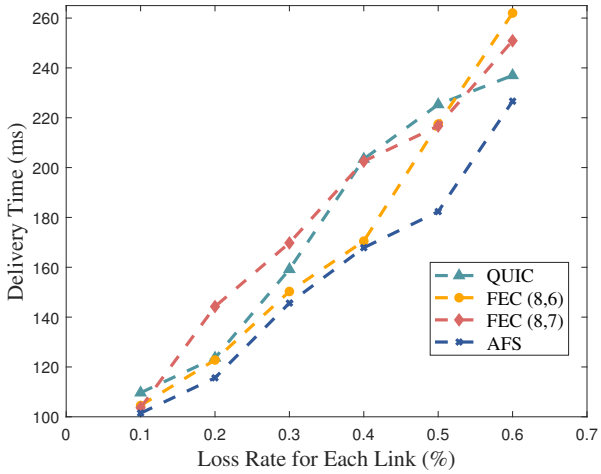

Fig. 4. Delivery time of video frames.

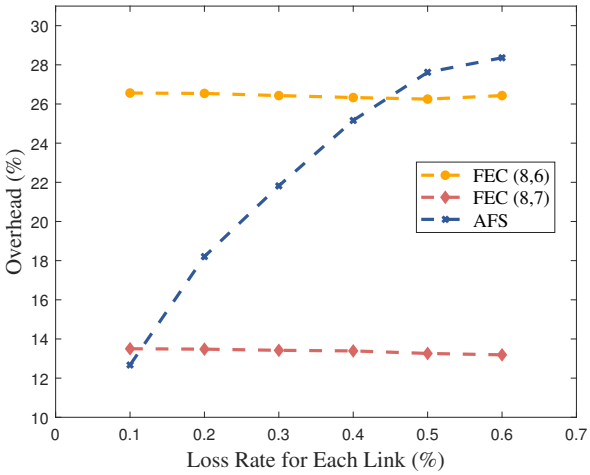

Fig. 5. Overhead for different schemes.

Apart from AFS, we also separately evaluate the performance of DDBS with other schedulers to obtain more generic conclusions. Fig. 6 and Fig. 7 show the all video frames and high-priority video frames completion ratio before deadline in different emulation scenarios. Compared with other schedulers, DDBS substantially improves the completion ratio of both all frames and high-priority frames in nearly all conditions. We also observe that DDL performs best for all frames completion ratio when the loss rate is 0.1%. The reason is that bandwidth is relatively sufficient to transmit video frames when less packet losses occur and DDL can achieve considerable performance in this situation even only considering the deadline. DDBS takes the frame priority into account and achieves a trade-off between all frames completion and high-priority frames completion, and have a more conservative judgment to select which frame to send.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3540322

As a result, DDBS gets a little lower completion ratio for all frames compared with that of DDL but much higher high-priority frames completion ratio due to the aware of priority when the loss rate is 0.1%. As the loss rate increases, packet losses cause substantial fluctuations in the network and significantly increase the transmission latency of individual frames, which leads to more queuing delay for subsequent frames. Those frames that are originally able to be received before deadline will be affected, resulting in a decrease for the completion ratio. DDBS overcomes this disadvantage by effectively estimating the block transmission time and delaying the transmission of blocks which are unlikely to be delivered before deadline. While DDL only considers individual frames deadline, ignoring current network conditions and fluctuations. Therefore, when packet loss rate is high, the performance of DDL is not as good as DDBS as shown in Fig. 6. For the original QUIC protocol, it utilizes the default FIFO scheme for frames scheduling. When packet losses occur, it requires additional retransmission time to complete the transmission of the entire frame. The additional delay will affect the all subsequent frames like the behaviors of DDL. Furthermore, the deadline constraint is ignored in QUIC protocol and it creates a vicious cycle, resulting in a dramatic degradation in completion ratio performance as demonstrated in Fig. 6. For high-priority frames completion ratio, DDBS takes block priority into account for the scheduling and achieves the best performance in all loss conditions as shown in Fig. 7. It further demonstrates the effectiveness of DDBS scheduler when many types of frames exist in sender buffer with limited available bandwidth.

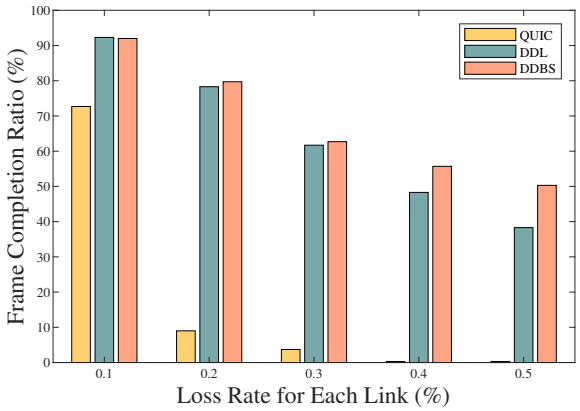

Fig. 6. All frames completion ratio before deadline.

Fig. 8 and Fig. 9 display the performance of regular QUIC protocol, AFS, DDBS and QRST, in case of all frames and high-priority frames completion ratio. As shown in the figures, QRST consistently performs best in all scenarios, demonstrating higher completion ratio for both high-priority and all frames. It clearly illustrates the significant advantages that DDBS can gain from the reduction of individual frames transmission time facilitated by AFS compared with any single component and it is necessary to integrate two components together for overall performance optimization. Noticeably, in Fig. 4, the delivery time is always lower than 350ms in all scenarios. However, the delivery time is recorded from the start of video frame transmission to its completion without the consideration of queuing time in sender buffer. Actually both FEC and retransmission schemes will introduce extra bandwidth consumption when packet loss rate increases, which introduces additional queuing delay for other video frames in sender buffer. This also explains why, even with the less delivery time, AFS still struggles to achieve a great frames completion ratio in higher loss rate as shown in Fig. 8.

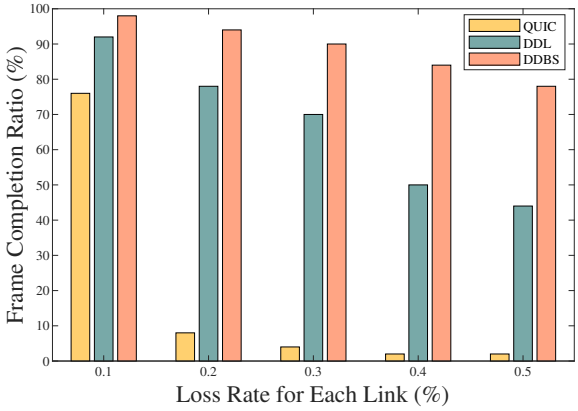

Fig. 7. High-priority frames completion ratio before deadline.

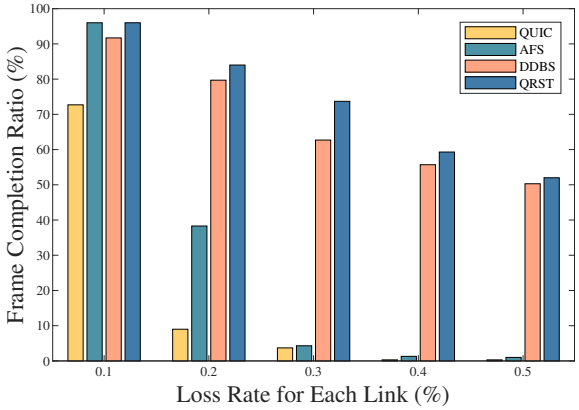

Fig. 8. Completion ratio of all frames.

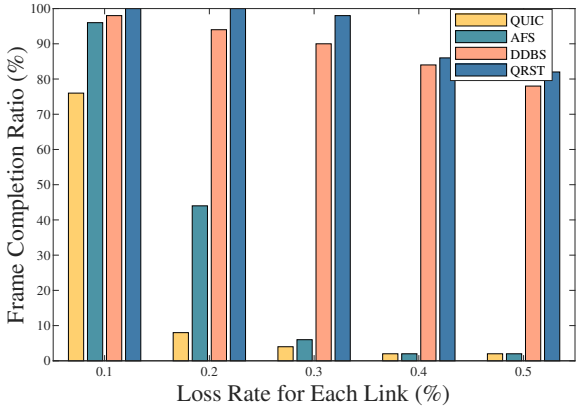

Fig. 9. Completion ratio of high-priority frames.

C. Overall Performance

Now we explore the overall performance of QRST in different scenarios compared with other reference schemes:

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3540322

QUIC, DDLF (8,6) and DDLF (8,7). We first assess the average bitrate and rebuffering time for these schemes, which are the key QoE metrics. In these experiments, the bandwidth is set to 8Mbps. As shown in Fig. 10, QRST performs the best average bitrate in all scenarios compared with others with limited bandwidth. And it achieves nearly 550Kbps in lowest loss rate and about 350Kbps in highest loss rate. As the loss rate increases, the advantage of QRST becomes more pronounced. Moreover, Fig. 11 illustrates the rebuffering time for these schemes. The results are consistent with the average bitrate, QRST achieves the lowest rebuffering time in all scenarios. Notably, DDLF (8,6) and DDLF (8,7) also outperform original QUIC protocol due to the FEC protection and deadline-aware scheduling, they can deliver more blocks before deadline and get lower rebuffering time. However, neither of the two schemes is inherently superior, which depends on the adaption to specific scenarios. As a result, the two metrics which are closely related to frame completion ratio can significantly benefit from adaptive FEC protection and overall block scheduling in QRST. Especially in ULSLSNs, it can effectively reduce the impact of network instability on block transmission and more blocks can be ensured to be delivered before deadline, therefore user's QoE is enhanced.

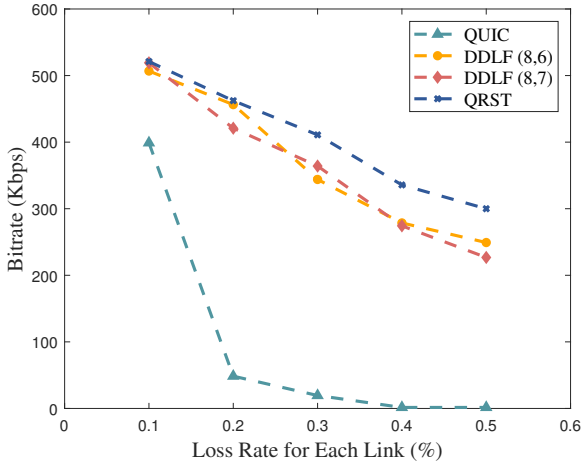

Fig. 10. Average bitrate.

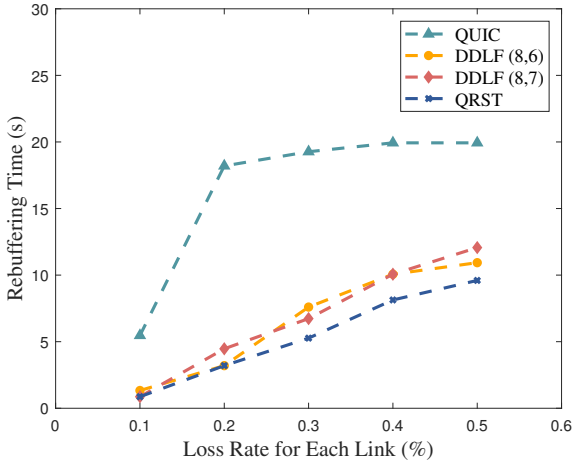

Fig. 11. Rebuffering time.

To further observe the performance of QRST in detail, we give the comprehensive performance analysis with metrics of completion ratio for all and high-priority frames as shown in Fig. 12 and Fig. 13. The bandwidth is 8Mbps. For all frames, QRST achieves the highest completion ratio under all loss rate and demonstrates the robust ability even with higher loss rate. Moreover, it exhibits a more pronounced advantage in terms of high-priority frames completion. Specifically, QRST can achieve the completion ratio of high-priority frames for more than 80% under higher loss rate and 100% under lower loss rate, which are much more than that of all frames. And for other schemes, the high-priority frames completion ratio is similar with that of all frames. The reason is that QRST considers the importance of critical frames sufficiently while other schemes treat all frames equally, nearly identical completion ratio are displayed.

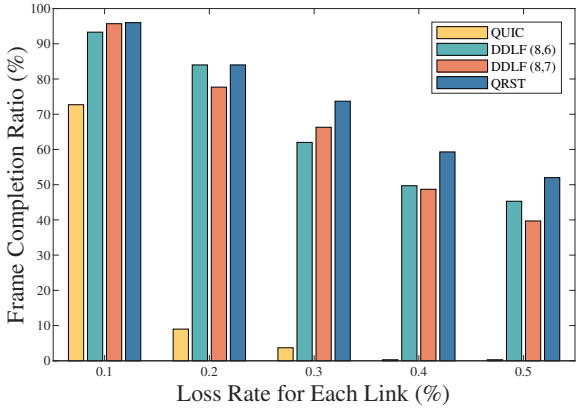

Fig. 12. Completion ratio of all frames.

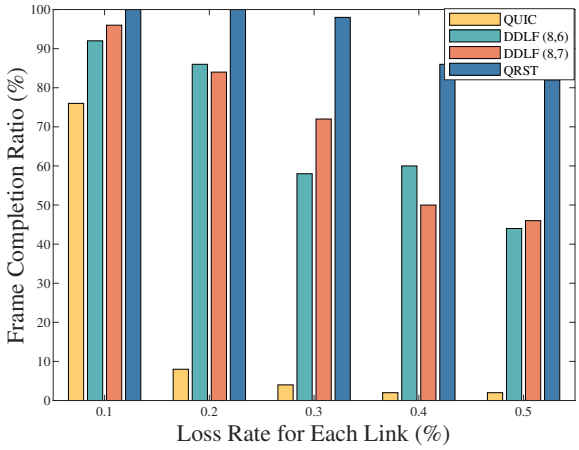

Fig. 13. Completion ratio of high-priority frames.

In order to reveal the performance of QRST in more scenarios, we present the evaluation results with the bandwidth of 6Mbps and 10Mpbs referring to less and more bandwidth resources, respectively. Metrics we selected are still the all and high-priority frames completion ratio to show the transmission performance of our framework for real-time applications. As shown in Fig. 14 and Fig. 15, the bandwidth is 6Mbps which is more limited and all schemes generally degrade performance compared with that of 8Mbps bandwidth. However, QRST still

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3540322

maintains considerable improvement in both completion ratios due to the adaptability to different scenarios. Specifically, QRST achieves more than 70% and 90% under lower loss rate, and more than 40% and 60% under higher loss rate in terms of the all frames and high-priority frames completion ratio, respectively. The adaptability of QRST is benefited from the robustness of both DDBS and AFS. Even with limited bandwidth, it's able to make practical decisions based on network conditions, thereby improving the transmission performance of video application. However, it is worth noting that DDLF (8,7) exhibits the best performance in terms of all frames completion ratio when the loss rate is 0.1%. This can be attributed to the fact that QRST strikes a balance between the completion ratio of all frames and high-priority frames, as illustrated earlier.

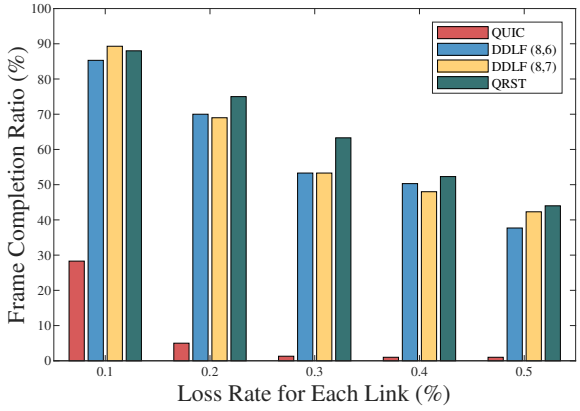

Fig. 14. Completion ratio of all frames with bandwidth 6Mbps.

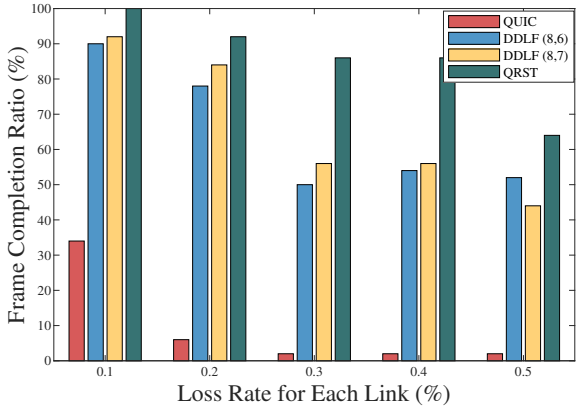

Fig. 15. Completion ratio of high-priority frames with bandwidth 6Mbps.

Fig. 16 and Fig. 17 illustrate the transmission performance of various schemes with the bandwidth of 10Mbps. The results are similar to the scenarios with the bandwidths of 6Mbps and 8Mbps. However, with higher bandwidth, QRST and other schemes can achieve more optimal improvements. And QRST continues to exhibit the best performance across both all frames completion ratio and high-priority frames completion ratio. It further demonstrates the robustness of the QRST framework.

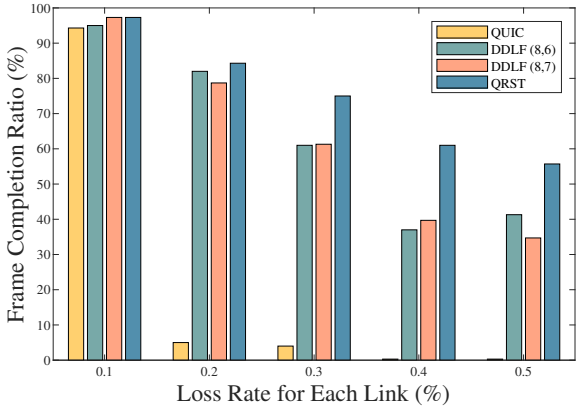

Fig. 16. Completion ratio of all frames with bandwidth 10Mbps.

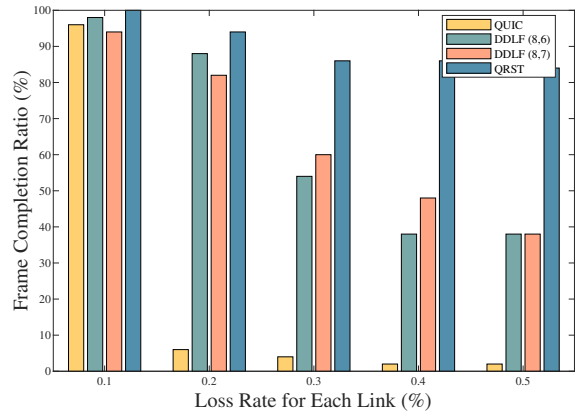

Fig. 17. Completion ratio of high-priority frames with bandwidth 10Mbps.

VI. Conclusion

In this paper, we have proposed a QUIC-enabled Robust Streaming Transmission (QRST) framework for Ultra Large-Scale LEO Satellite Networks. Our proposed QRST mainly involves two key components: an adaptive FEC scheme and a deadline-driven block scheduler. The FEC scheme can dynamically determine the ratio between the number of Source Symbols and Repair Symbols based on the current network conditions to reduce retransmission latency and balance bandwidth overhead. Our designed block scheduler considers various block properties, including deadline, size and priority for scheduling. It can prioritize the transmission of the block which has more urgent deadline and higher priority in sender buffer. Our approach has improved the completion ratio of all blocks, with a particular focus on high-priority blocks. We have implemented QRST by PQUIC and conducted the experiments over the Kupier K3 Shell network simulated by NS-3. The verified results have shown that QRST can effectively reduce the transmission time of individual blocks, improve user's QoE and the completion ratio of all and high-priority blocks for real-time applications in LEO satellite networks.

VII. Acknowledgment

This work is partial supported from the National Key R&D Program of China under Grant 2020YFB1806104 and the Natural Science Foundation of Jiangsu Province Youth Project (BK20180329).

References

[1] F. Tang, H. Hofner, N. Kato, K. Kaneko, Y. Yamashita, and M. Hangai, "A deep reinforcement learning-based dynamic traffic offloading in space-air-ground integrated networks (sagin)," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 276–289, 2021.

[2] H. Shi, Y. Cui, F. Qian, and Y. Hu, "Dtp: Deadline-aware transport protocol," in *Proceedings of the 3rd Asia-Pacific Workshop on Networking*, ser. APNet '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–7. [Online]. Available: https://doi.org/10.1145/3343180.3343191

[3] F. Tang, C. Wen, L. Luo, M. Zhao, and N. Kato, "Blockchain-based trusted traffic offloading in space-air-ground integrated networks (sagin): A federated reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3501–3516, 2022.

[4] X. Qin, T. Ma, Z. Tang, X. Zhang, H. Zhou, and L. Zhao, "Service-aware resource orchestration in ultra-dense leo satellite-terrestrial integrated 6g: A service function chain approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 9, pp. 6003–6017, 2023.

[5] Y. Cao, S.-Y. Lien, Y.-C. Liang, and D. Niyato, "Toward intelligent non-terrestrial networks through symbiotic radio: A collaborative deep reinforcement learning scheme," *IEEE Network*, pp. 1–1, 2024.

[6] T. Ma, B. Qian, X. Qin, X. Zhang, N. Cheng, and H. Zhou, "Joint subchannel allocation and beamforming for multicast in ultra-dense leo backbone network," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 1625–1630.

[7] Z. Jia, M. Sheng, J. Li, D. Niyato, and Z. Han, "Leo-satellite-assisted uav: Joint trajectory and data collection for internet of remote things in 6g aerial access networks," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9814–9826, 2021.

[8] A. Boyle. (2019) Amazon to offer broadband access from orbit with 3,236 satellite 'project kuiper' constellation. [Online]. Available: https://www.geekwire.com/2019/amazonproject-kuiper-broadband-satellite/

[9] Telesat. (2020) Telesat: Global satellite operators. [Online]. Available: https://www.telesat.com/

[10] (2017) SpaceX Starlink. [Online]. Available: https://www.spacex.com/webcast

[11] T. Ma, B. Qian, X. Qin, X. Zhang, L. X. Cai, and H. Zhou, "Resource scheduling for high-capacity multicast service in ultra-dense leo satellite networks," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 2, pp. 2468–2481, 2024.

[12] F. Tang, C. Wen, X. Chen, and N. Kato, "Federated learning for intelligent transmission with space-air-ground integrated network toward 6g," *IEEE Network*, vol. 37, no. 2, pp. 198–204, 2023.

[13] S. Ma, Y. C. Chou, H. Zhao, L. Chen, X. Ma, and J. Liu, "Network characteristics of leo satellite constellations: A starlink-based measurement from end users," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

[14] X. Yuan, F. Tang, M. Zhao, and N. Kato, "Joint rate and coverage optimization for the thz/rf multi-band communications of space-air-ground integrated network in 6g," *IEEE Transactions on Wireless Communications*, vol. 23, no. 6, pp. 6669–6682, 2024.

[15] Q. Chen, L. Yang, Y. Zhao, Y. Wang, H. Zhou, and X. Chen, "Shortest path in leo satellite constellation networks: An explicit analytic approach," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 5, pp. 1175–1187, 2024.

[16] O. Habachi, H.-P. Shiang, M. Van Der Schaar, and Y. Hayel, "Online learning based congestion control for adaptive multimedia transmission," *IEEE transactions on signal processing*, vol. 61, no. 6, pp. 1460–1469, 2013.

[17] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. R. 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," *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:2768765

[18] S. Yang, H. Li, and Q. Wu, "Performance analysis of quic protocol in integrated satellites and terrestrial networks," in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2018, pp. 1425–1430.

[19] L. Vicisano, M. Watson, and M. Luby, "Forward Error Correction (FEC) Building Block," RFC 5052, Aug. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc5052

[20] I. Swett, M.-J. Montpetit, V. Roca, and F. Michel, "Coding for QUIC," Internet Engineering Task Force, Internet-Draft draft-swett-nwcrg-coding-for-quic-03, Jul. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-swett-nwcrg-coding-for-quic/03/

[21] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[22] J. Iyengar and I. Swett, "QUIC Loss Detection and Congestion Control," RFC 9002, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9002

[23] I. Swett, "Quic wg charter: Fec initially out of scope," IETF Document, iETF99, 2017.

[24] F. Michel, Q. De Coninck, and O. Bonaventure, "Quic-fec: Bringing the benefits of forward erasure correction to quic," in *2019 IFIP Networking Conference (IFIP Networking)*, 2019, pp. 1–9.

[25] J. Yu, S. Pan, R. Gao, and Y. Li, "Low-delay transmission for non-terrestrial networks based on fec and reinforcement learning," in *2023 IEEE/CIC International Conference on Communications in China (ICCC)*, 2023, pp. 1–6.

[26] M. Zverev, P. Garrido, F. Fernandez, J. Bilbao, O. Alay, S. Ferlin, A. Brunstrom, and R. Aguero, "Robust quic: Integrating practical coding in a low latency transport protocol," *IEEE Access*, vol. 9, pp. 138 225–138 244, 2021.

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

[28] M. Zhang, T. Ma, Z. Zhang, H. Zhou, and L. Zhao, "A quic-enabled reliable video transmission scheme in ultra-dense leo satellite networks," in *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, 2023, pp. 1–6.

[29] Y. Liu, W. Su, and L. Tan, "Tetris: Near-optimal scheduling for multipath deadline-aware transport protocol," in *2021 International Conference on Networking and Network Applications (NaNA)*, 2021, pp. 34–40.

[30] S. Zhang, W. Lei, W. Zhang, Y. Guan, and H. Li, "Congestion control and packet scheduling for multipath real time video streaming," *IEEE Access*, vol. 7, pp. 59 758–59 770, 2019.

[31] J. Wu, B. Cheng, M. Wang, and J. Chen, "Priority-aware fec coding for high-definition mobile video delivery using tcp," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 1090–1106, 2017.

[32] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "Mp-dash: Adaptive video streaming over preference-aware multipath," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 129–143. [Online]. Available: https://doi.org/10.1145/2999572.2999606

[33] X. Zuo, Y. Cui, X. Wang, and J. Yang, "Deadline-aware multipath transmission for streaming blocks," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 2178–2187.

[34] E. N. Gilbert, "Capacity of a burst-noise channel," *The Bell System Technical Journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[35] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings.*, vol. 1, 1999, pp. 345–352 vol.1.

[36] Q. De Coninck, F. Michel, M. Piraux, F. Rochet, T. Given-Wilson, A. Legay, O. Pereira, and O. Bonaventure, "Pluginizing quic," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19, 2019, p. 59–74. [Online]. Available: https://doi.org/10.1145/3341302.3342078

[37] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_2

[38] D. Camara, H. Tazaki, E. Mancini, T. Turletti, W. Dabbous, and M. Lacage, "Dce: Test the real code of your protocols and applications over simulated networks," *IEEE Communications Magazine*, vol. 52, no. 3, pp. 104–110, 2014.