Received March 1, 2022, accepted March 20, 2022, date of publication March 25, 2022, date of current version April 1, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3162217

Sliding-Window Forward Error Correction Based on Reference Order for Real-Time Video Streaming

RUI WANG, LIANG SI, AND BIFENG HE
Agora Inc., Hangzhou 310000, China

Corresponding author: Rui Wang (wangrui@agora.io)

This work was supported in part by Agora Inc.

ABSTRACT In real-time video streaming, data packets are transported over the network from a transmitter to a receiver. The quality of the received video fluctuates as the network conditions change, and it can degrade substantially when there is considerable packet loss. Forward error correction (FEC) techniques can be used to recover lost packets by incorporating redundant data. Conventional FEC schemes do not work well when scalable video coding (SVC) is adopted. In this paper, we propose a novel FEC scheme that overcomes the drawbacks of these schemes by considering the reference picture structure of SVC and weighting the reference pictures more when FEC redundancy is applied. The experimental results show that the proposed FEC scheme outperforms conventional FEC schemes.

INDEX TERMS Forward error correction, real-time video streaming, scalable video coding, error correction codes, reference order.

I. INTRODUCTION

Real-time video streaming has been widely used in online education, e-commerce, live broadcasting, and video conferences [1]. Factors such as weak wireless signal strength in wireless connections and congestion at network nodes can cause data packet loss during streaming, which can negatively affect the quality of user experience (QoE). Several FEC schemes can be adopted to mitigate this problem [1]–[5].

In real-time video streaming, FEC schemes can be divided into the following common schemes: frame-level FEC, group of pictures (GOP)-level FEC, expanding-window FEC, and sliding-window FEC. One common approach is to perform Reed–Solomon (RS) coding at the frame level, where the RS coding block contains video packets from the same video frame (i.e., each coding window contains a single video frame and its corresponding FEC redundancies) [6]. Under this constraint, the RS decoder does not need to collect source packets of many frames to recover lost packets; therefore, there is no decoding delay. However, when the number of source video packets generated in each frame is small, FEC is inefficient. In addition, the recovered packets of the current frame cannot help to recover the lost packets of previous frames, and the distortion of the previous frames may propagate to the current and following frames. In GOP-level FEC, the FEC coding window contains all the video frames in a GOP to produce the corresponding FEC redundancies [7], [8]. By utilizing a large coding window, GOP-level FEC is capable of handling burst loss. However, GOP-level FEC brings additional decoding delay when all the video frames in the GOP must be collected for FEC decoding, which is unacceptable in real-time video streaming.

Frame-level FEC allocates repair packets for each frame of the GOP, whereas GOP-level FEC packets are allocated once for the entire GOP, which introduces an additional decoding delay. Expanding-window and sliding-window FECs both eliminate the additional delay by allocating repair packets for each frame. The difference is that expanding-window FEC uses the current video frame and all previous frames to construct the coding window, so the window expands continuously until the entire GOP is covered [10], [11], whereas sliding-window FEC adopts a fixed window that contains only a portion of the data of the previous encoding window [13], [14]. In expanding-window FEC, the coding window size increases linearly within one GOP; for example, the f-th group of FEC packets is generated after all source packets from frame 1 to f are collected. In practical implementations, the computational cost and decoding delay are

The associate editor coordinating the review of this manuscript and approving it for publication was Zilong Liu.

quite high when the size of the GOP is sufficiently large [12]. Thus, the application of expanding-window FEC in real-time video streaming is restricted. By contrast, sliding-window FEC can maintain good recovery performance without over-accumulating coding; it has sufficient recovery performance while maintaining low delay and low complexity. Sliding-window FEC has a sliding coding window that typically consists of a set of frames consecutively in time in the same GOP. In this paper, we call this kind of coding scheme "time-order sliding window FEC" or "continuous sliding window FEC".

In general video coding, low-priority frames (i.e., P-frames or B-frames) are encoded based on the preceding high-priority frames (i.e., I-frames or P-frames) in a video sequence [9]. Therefore, if the reconstruction of a high-priority frame fails, the subsequent low-priority frame is regarded as a reconstruction failure, even if it is successfully recovered by the FEC decoder.

Scalable video coding (SVC) is an extension of the H.264 video compression standard. In SVC, the video frames are grouped into layers: the base layer and one or more enhancement layers. With hierarchical video coding, the SVC can adapt to varying terminal capabilities and needs of the end user, and provides a graceful degradation when the network conditions change. However, the frame dependency is also changed when SVC is adopted, it is no longer a simple one-dimensional frame-by-frame dependency in time order [15], this results in the time-order sliding-window FEC not being optimal.

In this paper, we propose a sliding-window FEC scheme based on video coding dependencies, namely, "reference-order sliding window FEC" or "discrete sliding window FEC". The proposed scheme uses the reference order as a constraint of FEC coding such that there must be a dependency between the frames in the encoding window. The proposed scheme automatically realizes unequal error protection (UEP) and does not need to allocate different amounts of redundancy to frames of different priority, because the scheme itself automatically obtains a higher recovery probability for higher-priority frames. Compared with conventional frame-level FEC and time-order sliding-window FEC, the proposed FEC scheme performs better in terms of the playable frame rate (PFR) and peak signal-to-noise ratio (PSNR).

The remainder of this paper is organized as follows. Section II describes the proposed FEC scheme. Section III provides a theoretical analysis of the proposed FEC scheme in the SVC mode. In Section IV, we present our experimental results for PFR and PSNR. Finally, conclusions are given in Section V.

II. SLIDING-WINDOW FEC BASED ON REFERENCE ORDER

A. CODING WINDOW MANAGEMENT MECHANISM BASED ON TIME ORDER

The window management mechanism of time-order sliding window FEC is based on the frame timestamps [16], [17]; that is, the video frames in the FEC coding window are arranged according to the time when they are generated by the video encoder. In general video coding, the frame dependence is continuous (i.e., frame-by-frame reference), as shown in Fig. 1. Thus, it is feasible to use the time order as the rule for coding window management. As shown in Fig. 3, Window-1, the first FEC coding window, contains the first video frame, which is the intra frame I, and the second FEC coding window (Window-2) contains frame I and frame P1. The third coding window (Window-3) contains frames I, P1, and P2. Considering the general case, let T be the maximum number of frames in the coding window, let frame X be the X-th frame produced by the video encoder, and let the FEC coding window corresponding to frame X be Window-X, containing the preceding frames $X - T$ to X.

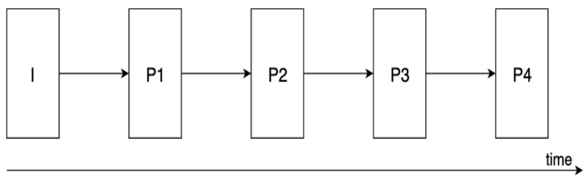


FIGURE 1. Example frame dependency in a GOP (non-SVC case).

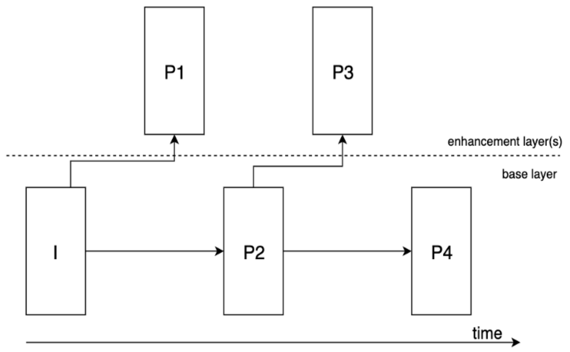


FIGURE 2. Example frame dependency in a GOP (SVC case).

B. THE PROBLEM WITH TIME-ORDER CODING WINDOW MANAGEMENT

SVC adopts a hierarchical structure, in which frames in a GOP are allocated to the base layer or enhancement layer. This approach causes the reference order between frames to change; that is, the previous frame may not necessarily be the reference frame for the current frame. For example, as shown in Fig. 2, frame P2 references frame I instead of frame P1.

If we apply the time-order mechanism in the SVC case, the coding window management will be the same as that in Fig. 3, whereas the coding dependency will follow Fig. 2. Consequently, this mechanism may reduce the recovery probability, particularly under high packet loss rate (PLR) conditions. Suppose that frames I, P1, and P2 correspond to 8, 4, and 4 packets, respectively (the frames are divided into packets for transmission based on the maximum transmission unit (MTU)). Under the conditions of RS coding, we use

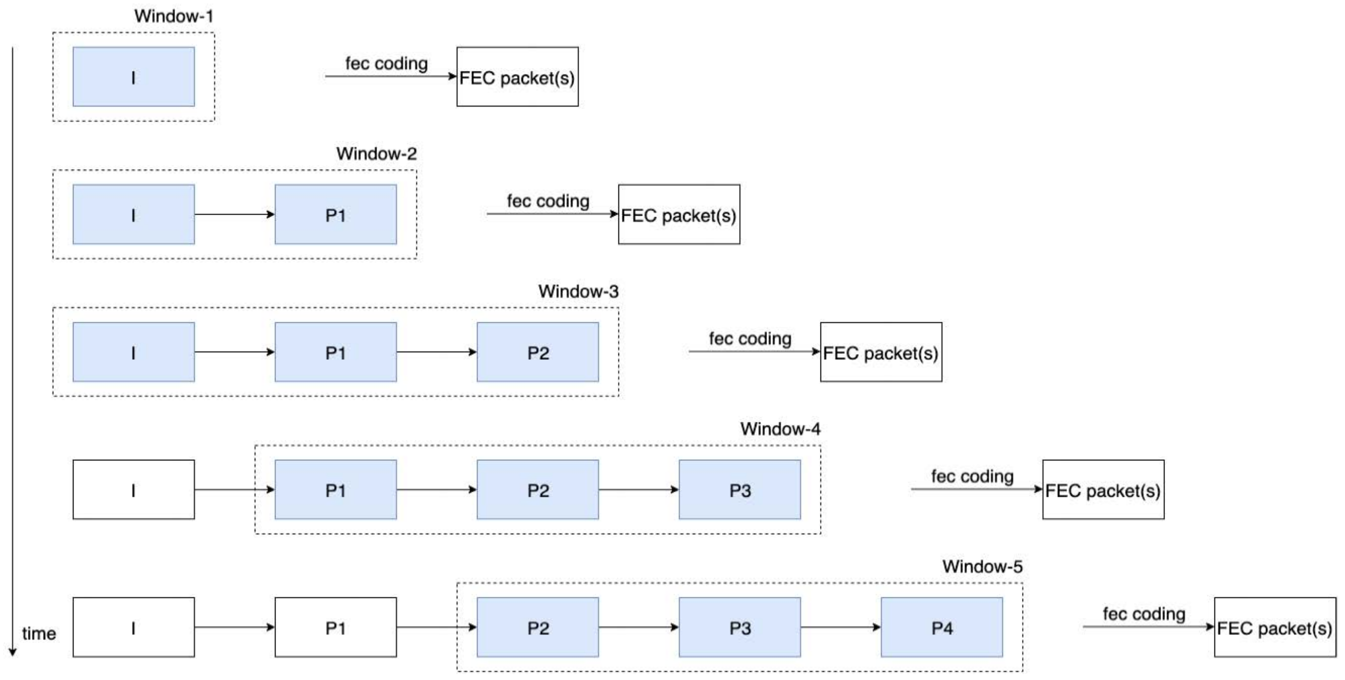


FIGURE 3. Coding window management mechanism based on time order.

RS(16,8) to encode Window-1 to generate 8 repair packets. For frame P1, RS(16,12) is used on Window-2 to generate 4 repair packets; for frame P2, RS(20,16) is used on Window-3 to generate 4 repair packets. Considering that the PLR is 50% and the decoder side is lucky in receiving 8 packets from frame I, Window-1 (containing only frame I) can be successfully decoded with RS coding. Under the further assumption that the decoder side also receives 3 packets from frame P1, this means that 11 packets in total have been received for Window-2, which is insufficient for decoding. As a consequence, at least 5 packets from frame P2 are required to decode Window-3; however, only 4 packets are required to successfully decode frame P2 in frame-level RS coding. Thus, the recovery probability of the time-order sliding window scheme is lower than that of the frame-level FEC. Because frame I is decodable and frame P2 references frame I in the SVC case, the recovered frame P2 can be rendered (which is not possible in frame-by-frame reference patterns); therefore, the PFR is also lower than that of frame-level FEC.

Thus, the window management mechanism based on time order is no longer appropriate in the SVC mode. The proposed coding window management mechanism based on reference order adapts to this scenario by considering the reference picture structure and works well for both SVC and non-SVC encoders.

C. CODING WINDOW MANAGEMENT MECHANISM BASED ON REFERENCE ORDER

Under the coding window management mechanism based on reference order, the frames in the coding window are arranged in accordance with the reference structure. When a new frame is generated and passed to the FEC encoder, the FEC encoder caches the frame and obtains its reference frame. During the FEC coding process, a backward search is performed to find the reference frames of the current frame, and all reference frames are added to the coding window. The backward search stops when the encoding window exceeds the maximum window size or when an intra frame is found. As shown in Fig. 4, Window-1 contains the first frame (i.e., frame I), and Window-2 contains frames I and P1. Because the reference frame of frame P2 is frame I rather than frame P1, Window-3 contains only frames I and P2, whereas under the coding window management mechanism based on time order, Window-3 contains frames I, P1, and P2. Compared with the coding window management mechanism based on time-order sliding window FEC, the new method guarantees that all frames in the FEC coding window have a coding dependency; therefore, whenever an FEC coding block is decodable, the recovered frames are ensured to be decodable by the video decoders in the subsequent video pipeline and can then be successfully rendered. For non-SVC encoders, reference-order coding window management works similarly to the time-order scheme.

The proposed reference-order-based window management method implements an efficient UEP because the higher the priority of a video frame is, the more referenced frames are obtained (either directly or indirectly) and the more times the frame is encoded; thus, the recovery probability is also higher. In the example of coding window management illustrated in Fig. 4, frame I has the highest priority, followed by frame P2.

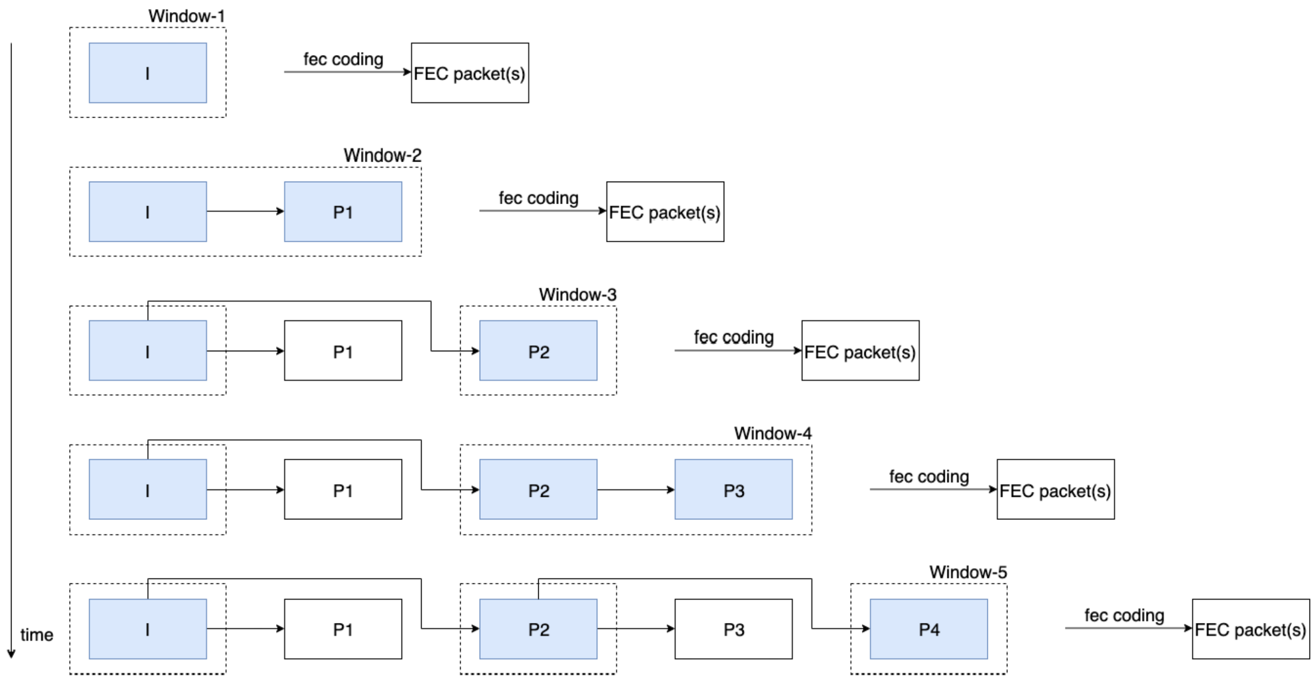


FIGURE 4. Coding window management mechanism based on reference order (SVC case).

Frames P1, P3, and P4 have the lowest priority because these three frames are not referenced by any other frame. In the proposed reference-order encoding process, frame I is encoded five times, and successful decoding of any one of these five coding blocks will recover frame I. Moreover, frame P2 is encoded three times, and frames P1, P3, and P4 are encoded only once, which means that the recovery probability of frame I is higher than that of frame P2, which is, in turn, higher than that of the other frames.

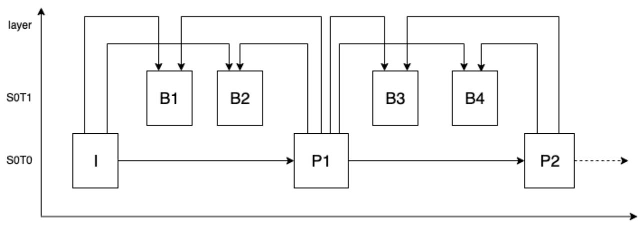


FIGURE 5. Example frame dependency in a GOP (bidirectional prediction structure).

D. CODING WINDOW MANAGEMENT MECHANISM FOR COMPLEX ENCODING STRUCTURES

Fig. 2 shows an example unidirectional prediction structure with 2-layer temporal scalability encoding. This section explores the coding window management based on reference order for more complex encoding structures, e.g., bidirectional prediction structures and prediction structures with spatial layers.

The coding window management mechanism for complex encoding structures follows the same principle; that is, the FEC coding window should contain only the reference frames of the current frame. When FEC coding is performed, the FEC encoder searches backward to find the reference frames of the current frame (namely, the direct reference frames, of which there may be one or more) and the reference frames of the direct reference frames (namely, the indirect reference frames). Then, all reference frames (direct and indirect) are arranged in accordance with their generation timestamps. For the case of spatial SVC, frames with the same timestamp are arranged by their spatial resolution. Suppose that the maximum coding window size is T. Then, to perform FEC coding, the FEC encoder selects the last $T-1$ frames from the sorted queue to construct the encoding window with the current frame.

Fig. 5 shows an example of a bidirectional prediction structure with 1-layer spatial and 2-layer temporal scalability encoding. Following the above principles, when T is set to 4, the coding window for frame B1 is composed of frames I, P1, and B1, the coding window for frame B3 is composed of frames I, P1, P2, and B3, and the coding window for frame P2 is composed of frames I, P1, and P2.

Fig. 6 shows an example prediction structure with 2-layer temporal and 2-layer spatial scalability encoding. Following the same principles, when T is set to 4, the coding window for frame P_{21} consists of frames I_{00}, I_{01}, P_{20}, and P_{21}, and the coding window for frame P_{31} consists of frames P_{20}, P_{21}, P_{30}, and P_{31}. When T is set to 6, the coding window for frame P_{31} is composed of frames I_{00}, I_{01}, P_{20}, P_{21}, P_{30}, and P_{31}.

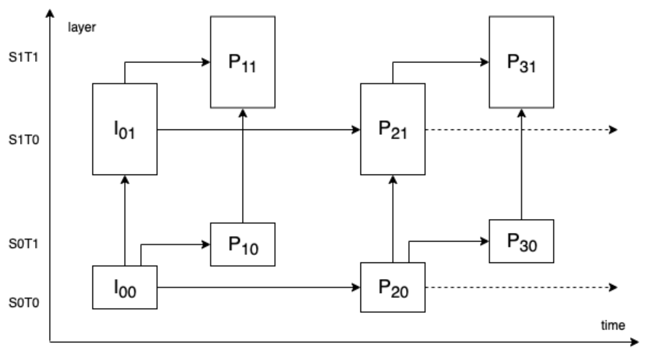


FIGURE 6. Example frame dependency in a GOP (2-layer spatial and 2-layer temporal structure).

III. THEORETICAL ANALYSIS

A. RECOVERY PROBABILITY CALCULATION

In this section, we use the systematic RS erasure code as an example to study the decoding probability of the aforementioned FEC schemes. For simplicity, an independent Bernoulli process is used to model the packet loss. Suppose that there are k source packets in the RS coding window and that k source packets are encoded via RS coding to generate h redundant packets ($k + h = n$). When n packets are sent over the transmission channel, the probability that b packets will be received is as follows:

$$U(n, b) = \binom{n}{b} \times p^{n-b} \times (1-p)^b \tag{1}$$

where p denotes the packet loss probability.

As long as the receiver receives at least k of the n packets, the lost source packets can be recovered by the RS decoder; thus, the probability of the FEC coding window being successfully reconstructed is as follows:

$$S(k, h) = \sum_{i=k}^{k+h} U(k+h, i) \tag{2}$$

To understand the data recovery processes of frame-level FEC, sliding-window FEC, and the proposed sliding-window FEC scheme based on reference order, this section explores the recovery probability for a group of frames consisting of three given frames in the SVC mode (e.g., the first three frames in Fig. 2). Let these three frames be frames I, P1, and P2.

We use k_1, k_2, and k_3 to denote the numbers of original packets from frames I, P1 and P2, respectively, and h_1, h_2, and h_3 to denote the corresponding numbers of redundant packets for each frame. Let X, Y, and Z similarly represent the numbers of packets received from each frame. Thus, we have $0 <= X <= k_1 + h_1, 0 <= Y <= k_2 + h_2, 0 <= Z <= k_3 + h_3$. The recovery probability for Window-i is denoted by Q_i. If Window-1 contains only frame I, then the recovery probability for Window-1 is given by the following equation:

$$Q_1 = P\{X >= k_1\} = S(k_1, h_1) \tag{3}$$

For Window-2, the FEC encoding window contains two frames: frame I and frame P1. To successfully recover Window-2, the following two conditions must be met:

$$\begin{aligned} Q_2 &= P\{Y >= k_2, X + Y >= k_1 + k_2\} \\ &= \sum_{j=k_2}^{k_2+h_2} \sum_{i=k_1+k_2-j}^{k_1+h_1} P\{Y = j, X = i\} \end{aligned} \tag{4}$$

Because X, Y, and Z are independent, the recovery probability for Window-2 can be further expressed as follows:

$$\begin{aligned} Q_2 &= \sum_{j=k_2}^{k_2+h_2} \sum_{i=k_1+k_2-j}^{k_1+h_1} [P\{Y = j\} \times P\{X = i\}] \\ &= \sum_{j=k_2}^{k_2+h_2} \sum_{i=k_1+k_2-j}^{k_1+h_1} [U(k_2 + h_2, j) \times U(k_1 + h_1, i)] \end{aligned} \tag{5}$$

Under the sliding-window FEC scheme based on time order, the frames contained in Window-3 are frames I, P1, and P2, and the recovery probability can be expressed as follows:

$$\begin{aligned} Q_3^t &= P\{Z >= k_3, Y + Z >= a, X + Y + Z >= b\} \\ &= \sum_{m=k_3}^{k_3+h_3} \sum_{j=c}^{k_2+h_2} \sum_{i=d}^{k_1+h_1} [P\{Z = m\} \times P\{Y = j\} \times P\{X = i\}] \end{aligned} \tag{6}$$

where $a = k_2 + k_3, b = k_1 + k_2 + k_3, c = a - m, d = b - m - j$, $P\{Z = m\} = U(k_3 + h_3, m)$, $P\{Y = j\} = U(k_2 + h_2, j)$, and $P\{X = i\} = U(k_1 + h_1, i)$.

In contrast to the conventional sliding-window FEC scheme, the proposed FEC scheme takes the reference order as the criterion for moving and expanding the encoding window. Thus, frame P1 is excluded from Window-3, and the frames contained in Window-3 are frames I and P2. Therefore, the recovery probability for Window-3 is given by the following equation:

$$\begin{aligned} Q_3 &= P\{Z >= k_3, X + Z >= k_1 + k_3\} \\ &= \sum_{m=k_3}^{k_3+h_3} \sum_{i=k_1+k_3-m}^{k_1+h_1} [U(k_3 + h_3, m) \times U(k_1 + h_1, i)] \end{aligned} \tag{7}$$

Given the recovery probabilities for all three windows, the PFR can be calculated. Notably, when Window-1 can be reconstructed, the number of playable frames should be one; when Window-2 or Window-3 can be reconstructed, the number of playable frames should be two; and when Window-2 and Window-3 can both be reconstructed, the number of playable frames should be three instead of four because frame I can be rendered only once, although it is recovered by both windows. Therefore, the expected PFR under the proposed FEC scheme is given by Equation 8.

$$\begin{aligned} \textit{expected PFR} &= 2 \times Q_2 + 2 \times Q_3 + Q_1 \times (1 - Q_2) \\ &\quad + Q_1 \times (1 - Q_3) - Q_2 \times Q_3 \end{aligned} \tag{8}$$

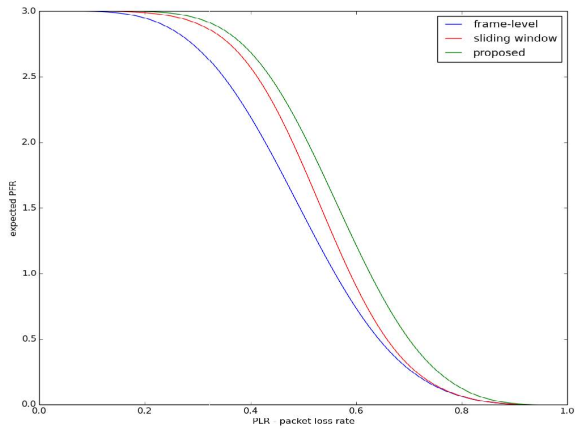


FIGURE 7. Expected PFRs of different FEC schemes given various PLRs in the SVC mode, with $k_1 = k_2 = k_3 = 4$ and $h_1 = h_2 = h_3 = 4$.

Similarly, the expressions for the expected PFRs of the frame-level FEC scheme and the time-order sliding-window FEC scheme can be deduced, as shown in the following equations:

$$\begin{aligned} Q_{b1} &= Q_1\{sliding - window\ FEC\} = Q_1 \\ Q_{b2} &= Q_2\{sliding - window\ FEC\} = Q_2 \\ Q_{b3} &= Q_3\{sliding - window\ FEC\} = Q_3^t \\ &expected\ PFR\ \{sliding - window\ FEC\} \\ &\quad = 3 \times Q_{b3} + 2 \times Q_{b2} \times (1 - Q_{b3}) \\ &\qquad + Q_{b1} \times (1 - Q_{b2}) \times (1 - Q_{b3}) \end{aligned} \tag{9}$$

Based on the above-derived expected PFR expressions, we draw curves to compare the expected PFRs of the different FEC schemes under different PLRs in the SVC mode, as shown in Fig. 7. The proposed FEC scheme outperforms both the time-order sliding-window FEC scheme and the frame-level FEC scheme.

$$\begin{aligned} Q_{a1} &= Q_1\{frame - level\ FEC\} = S(k_1, h_1); \\ Q_{a2} &= Q_2\{frame - level\ FEC\} = S(k_2, h_2); \\ Q_{a3} &= Q_3\{frame - level\ FEC\} = S(k_3, h_3); \\ &expected\ PFR\ \{frame - level\ FEC\} \\ &\quad = Q_{a1} + Q_{a1} \times Q_{a2} + Q_{a1} \times Q_{a3} \end{aligned} \tag{10}$$

B. DELAY ANALYSIS COMBINED WITH ARQ TECHNOLOGY

In practical real-time video streaming applications, FEC and automatic repeat request (ARQ) technologies are often used together to minimize the impact of network packet loss. In frame-level FEC, when the number of packets lost in a video frame exceeds the number of corresponding redundant packets, the receiver can only wait for a retransmission to obtain the lost source packets. The retransmission delay is related to the round-trip time (RTT) and the PLR. In high-RTT networks, the retransmission delay can be too large to wait for the retransmitted packets for real-time applications. In the proposed FEC scheme, the subsequent redundant packets can help recover the packet loss of previous frames. As long as the recovered packet arrives earlier than the retransmitted packet, the receiver can use that packet and cancel the repeat request, which is helpful in reducing retransmission delay and bandwidth consumption.

IV. PERFORMANCE EVALUATION

The video sequences used in this experiment are commonly used video test sequences in the 4:2:0 YUV format (e.g., "*Foreman*", "*Flower*", and "*Silent*"). The video codec used is OpenH264. Each GOP contains 15 frames; the frame sequence in each GOP is "$IP_eP_bP_eP_bP_eP_bP_eP_bP_eP_bP_eP_bP_eP_b$", where P_e is a P frame in the enhancement layer and P_b is a P frame in the base layer. The spatial resolution of the test sequences is 352 * 288.

At the encoder, we set the frame rate to 30 fps. The target bitrate was set to 500000 bps. The OpenH264 encoder's usage type was "*CAMERA_VIDEO_REAL_TIME*". To observe the error correction performance of the different FEC methods in the SVC mode, we set the temporary layer number to 2. The remaining settings of the OpenH264 encoder followed the default configuration. For the decoder, the default configuration of OpenH264 was adopted.

The PFRs and PSNRs under different PLRs were experimentally analyzed using (1) the frame-level FEC scheme [7], (2) the conventional sliding-window FEC scheme based on time order (referred to simply as the sliding-window scheme for brevity) [18], and (3) the sliding-window FEC scheme based on reference order (the proposed scheme).

A. PFR

The PFR is a significant indicator for evaluating FEC performance in real-time video communication that reflects the efficiency of real-time error correction [19].

Fig. 8 compares the PFR ratios of the different FEC schemes under various PLRs in SVC mode. As shown in Fig. 8, when the PLR is less than or equal to 6%, the three FEC schemes all exhibit high performance. When the PLR is 8%, 10%, 12%, 14%, 16%, 18%, and 20%, the PFR is respectively 0.939, 0.8841, 0.8535, 0.811, 0.7733, 0.7199, and 0.6823 for frame-level FEC [7] and 0.9729, 0.9421, 0.911, 0.8643, 0.8439, 0.791, and 0.7029 for time-order sliding-window FEC [18]. When the proposed FEC scheme is used, the PFR is further increased to 0.992, 0.9891, 0.9886, 0.9825, 0.9783, 0.9493, and 0.8982, respectively.

Thus, the proposed FEC scheme outperforms both the time-order sliding-window FEC scheme [18] and the frame-level FEC scheme [7], particularly at high PLRs.

B. PSNR

The PSNR is an indicator for evaluating video quality that represents the ratio of the maximum possible signal power to the destructive noise power. The higher the PSNR is, the higher the video quality [20]. In other words, the PSNR

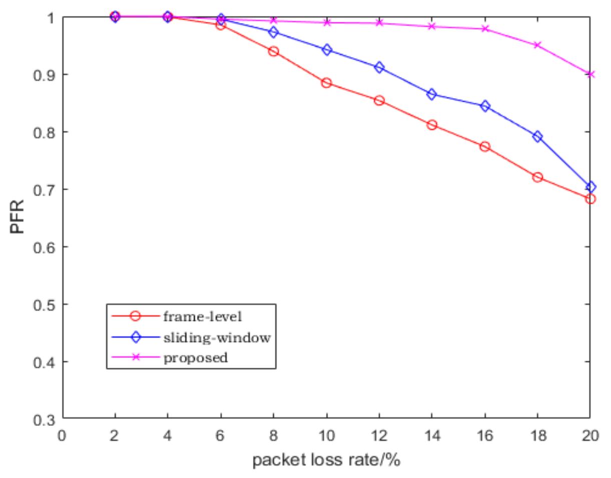


FIGURE 8. PFR ratios of different FEC schemes under various PLRs in SVC mode.

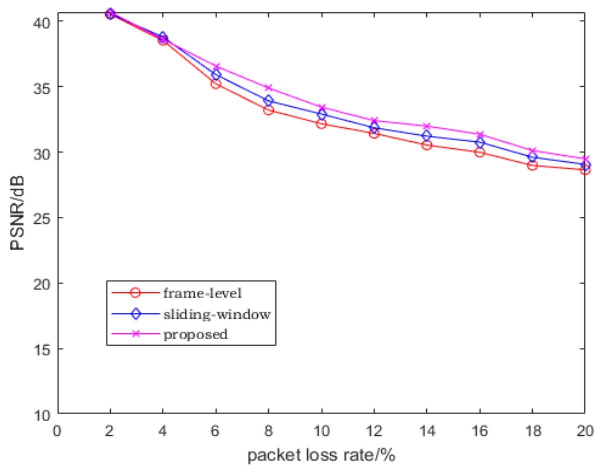


FIGURE 9. PSNRs of different FEC schemes under various PLRs in SVC mode.

reflects the extent to which video communication is affected by packet loss.

Fig. 9 shows the PSNRs of the different FEC schemes under various PLRs in SVC mode. When the PLR is less than or equal to 4%, the three FEC schemes exhibit similar performance because it is not easy for sudden and continuous packet loss to occur under the condition of a low PLR. When the PLR is 6%, 8%, 10%, 12%, 14%, 16%, 18%, and 20%, the PSNR is respectively 35.22 dB, 33.21 dB, 32.17 dB, 31.43 dB, 30.54 dB, 29.99 dB, 28.98 dB, and 28.65 dB for frame-level FEC [7] and 35.92 dB, 33.91 dB, 31.86 dB, 31.22 dB, 30.76 dB, 29.61 dB, 29.06 dB, and 28.64 dB for sliding-window FEC [18]. When the proposed FEC scheme is used, the PSNR is further increased to 36.58 dB, 34.90 dB, 33.45 dB, 32.41 dB, 31.99 dB, 31.37 dB, 30.12 dB, and 29.49 dB, respectively.

The above experimental data illustrate that the proposed method shows good performance in maintaining video quality under a high PLR in SVC mode.

V. CONCLUSION

In this paper, we have studied sliding-window FEC in real-time video streaming. A sliding-window FEC scheme based on reference order is proposed to improve the error correction performance for SVC encoders. The proposed scheme considers the frame dependency characteristics of the video encoder (source coding) in coding window management for sliding-window FEC (channel coding) and achieves an optimal UEP solution. Experimental results show that compared with frame-level FEC and time-order sliding-window FEC, the proposed FEC scheme achieves notable improvements in terms of the PFR for SVC encoders.

REFERENCES

[1] A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forwarding error correction," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, pp. 819–828, Jun. 2000.

[2] H. Wu, M. Claypool, and R. Kinicki, "Adjusting forward error correction with temporal scaling for TCP-friendly streaming MPEG," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 1, no. 4, pp. 315–337, 2005.

[3] J. S. Plank and L. Xu, "Optimizing Cauchy Reed–Solomon codes for fault-tolerant network storage applications," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Jul. 2016, pp. 173–180.

[4] J. Xiao, T. Tillo, C. Lin, Y. Zhang, and Y. Zhao, "A real-time error resilient video streaming scheme exploiting the Late- and early-arrival packets," *IEEE Trans. Broadcast.*, vol. 59, no. 3, pp. 432–444, Sep. 2013.

[5] C. I. Kuo, C. K. Shieh, W. S. Hwang, and C. H. Ke, "Performance modeling of FEC-based unequal error protection for H.264/AVC video streaming over burst-loss channels," *Int. J. Commun. Syst.*, vol. 30, no. 1, 2017, Art. no. e2826.

[6] Y. Xu and T. Zhang, "Variable shortened-and-punctured Reed–Solomon codes for packet loss protection," *IEEE Trans. Broadcast.*, vol. 48, no. 3, pp. 237–245, Sep. 2002.

[7] Y. Liu, J. Liu, S. Ci, and Y. Ye, "Joint video/depth/FEC rate allocation with considering 3D visual saliency for scalable 3D video streaming," in *Proc. Vis. Commun. Image Process. (VCIP)*, Nov. 2014, pp. 1–6.

[8] M. Hussain and A. Hameed, "Adaptive video-aware forward error correction code allocation for reliable video transmission," *Signal Image Video Process.*, vol. 12, pp. 161–169, Jan. 2018.

[9] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Commun. Mag.*, vol. 44, no. 8, pp. 134–143, Aug. 2006.

[10] Y. Geng, X. Zhang, C. Zhou, and Z. Guo, "Unequal error protection for real-time video streaming using expanding window Reed–Solomon code," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 3763–3767.

[11] J. Xiao, T. Tillo, and Y. Zhao, "Real-time video streaming using randomized expanding Reed–Solomon code," *IEEE Trans. Circ Syst. Video Technol.*, vol. 23, no. 11, pp. 1825–1836, Feb. 2013.

[12] S. Cheng, L. Huang, X. Zhang, and Z. Guo, "Accelerated expanding-window FEC using RS-code for real-time video streaming," in *Proc. SPIE2nd Int. Conf. Hot Inf.-Centric Netw. (HotICN)*, Dec. 2019, pp. 63–65, doi: 10.1109/HotICN48464.2019.9063209.

[13] M. Lalam, K. Amis, and D. Leroux, "Sliding encoding-window for Reed–Solomon code decoding," in *Proc. 4th Int. Symp. Turbo Codes Rel. Topics 6th Int. ITG-Conf. Source Channel Coding*, Apr. 2006, pp. 1–6.

[14] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-window raptor codes for efficient scalable wireless video broadcasting with unequal loss protection," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1491–1503, Jun. 2010.

[15] J. Xu, X. Yang, S. Zheng, and L. Song, "Group-of-pictures-based unequal error protection for scalable video coding extension of H.264/AVC," *Opt. Eng.*, vol. 48, no. 6, 2009, Art. no. 060502.

[16] M. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-window digital fountain codes for streaming of multimedia contents," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 3467–3470.

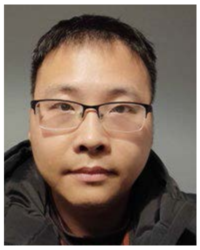
[17] V. Roca and B. Teibi, *Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME*, document IETF RFC 8681, 2020. [Online]. Available: www.rfc-editor.org/rfc/rfc8681.txt

[18] Y. Weng, C. Shih, and Y. Chou, ''Sliding-window forward error correction using Reed–Solomon code and unequal error protection for real-time streaming video,'' *Int. J. Commun. Syst.*, vol. 31, no. 1, 2018, Art. no. e3405.

[19] C. Shih, C. Kuo, and Y. Chou, ''Frame-based forward error correction using content-dependent coding for video streaming applications,'' *Comput. Netw.*, vol. 105, pp. 89–98, Aug. 2016.

[20] Q. Huynh-Thu and M. Ghanbari, ''Scope of validity of PSNR in image/video quality assessment,'' *Electron. Lett.*, vol. 44, no. 13, pp. 800–801, Jun. 2008.

RUI WANG was born in Zhoukou, China. He received the bachelor's degree in communication engineering from the Harbin Institute of Technology, in 2010. Then he participated in the software development of real-time audio and video streaming products at Huawei Technologies Company Ltd., and later led the development of WebRTC media servers at Hikvision Digital Technology Company Ltd. He is currently engaged in the development of video transmission and error correction coding at Agora.io. His research interests include real-time audio and video transmission, error correction coding, and WebRTC.



LIANG SI was born in Hangzhou, China. He received the bachelor's and master's degrees in control science from the Harbin Institute of Technology, in 2007 and 2009, respectively. After graduation, he worked as a Software Engineer at Huawei Technology Company Ltd., and later became the Manager of the Advanced Network System Engineering Department, Agora.io. His research interests include packet loss resistance and congestion control of real-time video and audio streams.

BIFENG HE was born in Wenzhou, China. He received the B.S. and M.E. degrees in telecommunication engineering from Hangzhou Dianzi University, China, in 2014 and 2018, respectively. From 2021 to 2022, he worked as a Video Engine Development Engineer at Agora.io. His research interests include image processing and forward error correction technique.