Real-time Video Streaming Exploiting the Late-arrival Packets

Jimin Xiao*§, Tammam Tillo§, Chunyu Lin**, Yao Zhao**
* Dept. of Electrical Engineering and Electronics, University of Liverpool, Liverpool, UK
Email: jimin.xiao@liverpool.ac.uk
§ Dept. of Electrical & Electronic Engineering Xi'an Jiaotong-Liverpool University, Suzhou, China
Email: tammam.tillo@xjtlu.edu.cn
** Institute of Information Science, Beijing Jiaotong University,
Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044
Email: {chunyulin,yzhao}@bjtu.edu.cn

***Abstract*—For real-time video applications, such as video telephony service, the allowed maximum end-to-end transmission delay is usually fixed. The packets arriving at the destination out of the maximum end-to-end delay are treated as *late-arrival* packets, and these packets are discarded in traditional video transmission systems. In this paper, in order to improve the system performance, we propose to exploit these packets to update the decoder reference buffer. Two schemes are proposed to exploit the *late-arrival* packets, one scheme is to use sliding-window updating, where the updating window is moving; another scheme is to use fixed-window update together with systematic Reed-Solomon code. The effectiveness of the two schemes are validated by simulation results without adding extra delay. It is found that in both schemes, the updating window size plays an important role on the system performance.**

***Index Terms*—systematic Reed-Solomon code, error resilience, error propagation, H.264/AVC, late-arrival packet**

I. Introduction

When transmitting video packets over unreliable networks, $i.e.$, Internet or wireless networks, some packets are dropped by the underlying networks, so they never arrive at the destination. Meanwhile, some packets may arrive at the destination after a long delay, which is larger than the maximum allowed end-to-end delay for the applications. In general, these packets are also regarded as lost packets by the traditional video applications; to distinguish these packets from the physically dropped packets ($i.e.$ *physical lost* packets), we will refer to them as *late-arrival* packets in the following. How many packets arrive at the destination after the display deadline is a function of the end-to-end delay, which is an application-dependent parameter. For the real-time video conferencing/telephony applications, the acceptable end-to-end delay is between 150 ms and 400 ms according to ITU-T G.114 [1].

In the current error resilient methods, the *physical lost* packets and *late-arrival* packets are treated in the same way. So, these methods mainly aim to mitigate the effects of the losses, and one commonly used approach is to insert some intra macroblocks to cut off the propagation paths of the distortion caused by the losses. For example, in [2], the end-to-end distortion is estimated at the encoder side, then the intra macroblocks are optimally inserted based on the end-to-end RD optimization. However, the coding efficiency of the Intra macroblock refreshment approaches is dramatically compromised, because the coding efficiency of intra mode is much lower than inter mode. On the other hand, feedback-based error control methods [3] as Automatic Repeat reQuest (ARQ) and feedback-based Reference Picture Selection (RPS), which also aim to stop the propagated errors, cause one round-trip time (RTT) delay, this makes them unsuitable for real-time video applications. Whereas, redundant slice/picture coding with equal or lower quality [4] and multiple description coding (MDC) [5][6] usually causes no additional delay. However, when the redundant version is used to replace the primary one, a mismatch error will appear in the decoding loop, and in general it will propagate to the following frames until the end of the Group of Pictures (GOP). One more set of effective approaches to combat packet losses is based on the use of Forward Error Correction (FEC) codes. In [7], the error resilient performance of the Redundant Slice MDC (RS-MDC) [5] has been shown to be less performing than the Unequal Error Protection coding scheme based on FEC (ULP-FEC) [7] applied at GOP level. However, in [7], the advantage of the ULP-FEC approach over RS-MDC is obtained because the Reed-Solomon (RS) coding block includes the whole GOP, and consequently one GOP of delay is caused, whereas RS-MDC causes no delay.

Unlike all the previous error resilient approaches, the *physical lost* packets and the *late-arrival* packets are treated differently in this paper, and we propose to exploit the information of the *late-arrival* packets. To do this, two schemes are proposed, one scheme is to use sliding-window updating, which means the *late-arrival* packets within the sliding-window are re-decoded, and the reference buffer of the decoder will be updated with the re-decoded information. Another scheme is to use fixed-window updating together with systematic Reed-Solomon code, where all the *late-arrival* packets within the fixed-window will be used by the RS decoder to try to recover other unavailable packets, then together with the recovered packets (if any), the *late-arrival* packets will be re-decoded to update the decoder reference buffer. In both the two schemes, it is found that the updating window size plays an important role on the system performance.

The rest of the paper is organized as follows. In Section II, the proposed two schemes are presented; In Section III some simulation results validating the proposed schemes are given. Finally, some conclusions are drawn in Section IV.

II. The Proposed Two Schemes

In this section, the proposed two schemes of exploiting the *late-arrival* packets are presented in detail, with their strength and weakness clarified.

A. Sliding-Window Updating Scheme

The sliding-window updating scheme is used to exploit the information contained in the *late-arrival* packets, where the *late-arrival* packets arriving at the destination within the sliding-window are decoded and used to update the decoder reference buffer. The size of the sliding-window is fixed, and it includes several consecutive frames before the current frame. Figure 1 shows one example to explain the sliding-window updating scheme, where the sliding-window size is

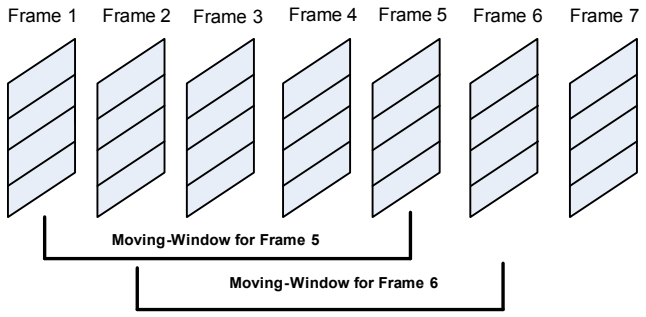

Figure 1. One example of the sliding-window updating scheme, with a sliding window size of 5.

5. For easy demonstration, let us assume the maximum end-to-end delay is 150 ms, the frame rate is 30 fps, which means that the time interval between two adjacent frames is 33.3 ms. In this example, as the sliding-window size is 5, the updating window for Frame 6 starts from Frame 2. So, before decoding and displaying Frame 6, any *late-arrival* packets belonging to frames [2-6] will be decoded and used to update the decoder reference buffer. Let us assume the end-to-end transmission delay for one source packet in the Frame 2 is 270 ms, being larger than the maximum end-to-end delay, so this packet is considered as *late-arrival* packet. In this case, this packet will arrive at the receiver side before the display deadline of Frame 6. As the sliding-window for Frame 6 includes frames [2-6], this packet will be re-decoded and used to update the decoder reference buffer. However, in this example, if the end-to-end transmission delay for this packet is more than $(150 + 33.3 * 4)$ ms, the arrival of this packet would be beyond the updating sliding-widow, therefore it will be discarded.

B. Fixed-Window Updating with Forward Error Correction

Instead of using sliding-window updating, we use fixed-window updating in this scheme, where the windows are not overlapped. The benefit of fixed window is that, systematic Reed-Solomon parity packets can be easily added at the end of each window to further protect the video stream. For the systematic RS code (N, K), for every K source packets, $N - K$ parity packets are introduced to make up a codeword of packets. As long as a client receives at least K out of the N packets, it can recover all the source packets. If the received packet number is less than K, the received source packets can still be used, because they have been kept intact by the systematic RS encoding process. In general, for the same parity packet rate $\mu = (N - K)/K$, the larger the value of K is, the higher the performance of the systematic RS code can be. To show the effects of K on the error correction performance of the systematic RS code, Table I lists the remaining packet loss rate after the systematic RS code correction, for different values of K. It shows that for the same network packet loss rate and redundancy, large RS code block size can make the systematic RS code more efficient.

Figure 2 shows one simple example of how the fixed-windows and the RS parity packets are allocated. In this example, each fixed-window contains video packets of three frames, each frame generates 4 packets, and the parity packet rate is $\mu = 25\%$, so RS code $(15, 12)$ is used to protect the video packets.

For easy demonstration, we use the first window as an example. As previously assumed, the maximum end-to-end delay is 150 ms. The

Table I
THE REMAINING PACKET LOSS RATE AFTER THE SYSTEMATIC RS CODE CORRECTION WITH PARITY PACKET RATE 20%, WHERE RS CODING BLOCK SIZE IS $K = 5, 10, 15, 20, 30$, NETWORK PACKET LOSS RATE IS $p = 5\%, 10\%, 15\%$

p	$K = 5$	$K = 10$	$K = 15$	$K = 20$
5%	1.13%	0.51%	0.25%	0.13%
10%	4.10%	3.03%	2.38%	1.93%
15%	8.34%	7.62%	7.20%	6.91%

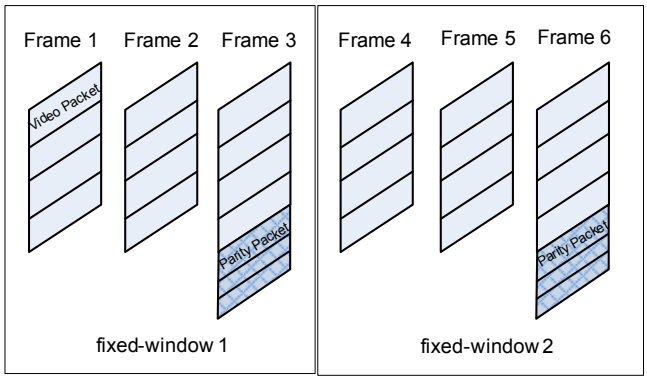

Figure 2. One example of the fixed-window updating scheme, with a fixed window size of 3.

end-to-end transmission delays for all the packets of the first fixed-window are listed in Table II, and Table III lists all the available packets by the display deadline of each frame, based on Table II and the maximum end-to-end delay (150 ms). By the display deadline of the first frame eight packets belonging to the first fixed-window are received, among them three packets belong to the first frame, with $S_{1,2}$ being unavailable [1]. In this case the $(15, 12)$ RS code used to protect the first Sub-GOP is not strong enough to recover the missed packets. Therefore, the decoder will decode the first frame with slice $S_{1,2}$ being concealed. At the display deadline of the second frame, the amount of received packets is also not enough to recover the lost packets, and consequently slice $S_{2,3}$ and $S_{2,4}$ will be concealed during the decoding process. However, slice $S_{1,2}$ is available now, so the first frame will be re-decoded (with slice $S_{1,2}$) to update the decoder reference buffer before decoding the second frame. By the display deadline of the third frame, the totally received packet number is 12, being enough to recover all the unavailable packets, so the RS decoder will recover all the unavailable source packets in this fixed-window, and re-decode all the frames in this fixed-window, in this case, there will be no concealment distortion in the third frame, and no concealment distortion propagates to the following frames.

The fixed-window size is one important parameter in this scheme. On one hand, in order to improve the RS code performance, large RS code block size (also means large fixed-window size) is preferred, and large fixed-window size also means that more *late-arrival* packets can be exploited. On the other hand, if the fixed-window contains too many frames, by the display deadline of the beginning frames of the fixed-window, the probability of receiving the packets for the end frames of the fixed-window and the parity packets is quite low. Therefore, for these frames, the probability of receiving enough packets to recover the lost or delay ones is quiet low. Consequently, video quality of these frames will degrade significantly, which will

[1] $S_{i,j}$ is used to denote the j-th slice in the i-th frame.

Table II
THE TRANSMISSION DELAY (MICROSECONDS) OF ALL THE PACKETS FOR THE FIRST FIXED-WINDOW. i IS THE FRAME INDEX; j IS THE PACKET INDEX; D MEANS THAT THIS PACKET IS PHYSICALLY LOST

Frame(i) \ Packet(j)	1	2	3	4	5	6	7
1	120	160	140	90	-	-	-
2	130	70	D	170	-	-	-
3	80	70	60	140	160	80	D

Table III
AVAILABLE PACKETS $S_{i,j}$, BY THE DISPLAY DEADLINE OF FRAME i; j IS THE PACKET INDEX.

$Frame$	Available packets
1	$S_{1,1}$, $S_{1,3}$, $S_{1,4}$, $S_{2,2}$, $S_{3,1}$, $S_{3,2}$, $S_{3,3}$, $S_{3,6}$
2	$S_{1,1}$, $S_{1,2}$, $S_{1,3}$, $S_{1,4}$, $S_{2,1}$, $S_{2,2}$, $S_{3,1}$, $S_{3,2}$, $S_{3,3}$, $S_{3,6}$
3	$S_{1,1}$, $S_{1,2}$, $S_{1,3}$, $S_{1,4}$, $S_{2,1}$, $S_{2,2}$, $S_{2,4}$, $S_{3,1}$, $S_{3,2}$, $S_{3,3}$, $S_{3,4}$, $S_{3,6}$

make the overall video quality low.

In this scheme, the RS parity packets will be evenly allocated for all the video packets, which means if the RS code block size includes more video packets, more RS parity packets will be dedicated to this block, and vice versa. Let us assume there are L frames in one Group of Pictures (GOP), the source packet number for each frame is $K(i), 1 \le i \le L$, the parity packet rate is μ, and each fixed-window includes packets of W frames. In this case, the parity packet number allocated for Frame i is denoted as $R(i), 1 \le i \le L$

$$R(i) = \begin{cases} \left\lceil \mu \sum_{j=1}^{kW} K(j) \right\rceil - \left\lceil \mu \sum_{j=1}^{(k-1)W} K(j) \right\rceil & \\ \quad \text{for } i = kW, \ k = 1, 2, \ldots & \\ \left\lceil \mu \sum_{j=1}^{L} K(j) \right\rceil - \left\lceil \mu \sum_{j=1}^{\lfloor \frac{L}{W} \rfloor W} K(j) \right\rceil & \\ \quad \text{for } i = L & \\ 0 \quad \text{otherwise} & \end{cases} \tag{1}$$

where the operation $\lceil X \rceil$ is used to get the minimum integer number greater than or equal to X, whereas operation $\lfloor X \rfloor$ is used to get the maximum integer number smaller than or equal to X.

C. Pros and Cons of Two Schemes

For the proposed two schemes, the *late-arrival* packets are exploited in two different fashions, both of which can achieve good performance without inducing extra delay. The one with sliding-window is standard compatible, and there is no need to add the Reed-Solomon encoder/decoder, which makes this scheme easy to be implemented. Moreover, as no protection packets are introduced in this scheme, the generated bit rate of the video stream will not change dramatically due to the inclusion of the RS parity packets. Whereas, for the scheme with fixed-window updating and systematic RS code, as shown in the following simulation part, its performance is higher with the penalty of relative higher complexity than the sliding-window scheme. However, for this scheme, as the RS parity packets are introduced, the bit rate of the video stream surges at the end of each fixed-window.

III. SIMULATION RESULTS

Our simulation environment is built on the JM14.0 [8] H.264/AVC codec. The GOP structure is IPPP..., the frame rate is 30 frames per second, the GOP length is 30 frames, and only the previous frame is used as the reference frame for the motion prediction. As for packetization, fixed slice length in term of byte is used, taking the MTU of wireless network into account, we set the target slice length as 400 bytes, one slice is encapsulated in one network packet. The average luminance PSNR is used to assess the objective video quality, and the mean PSNR is averaged over 100 trials. The packet loss and delay pattern for the Internet experiments specified in Q15-I-16r [9] is used to emulate the real Internet environments. In the following simulations, the maximum end-to-end delay is set to be 150 ms, this is because an end-to-end latency of 150 ms is barely noticeable [1].

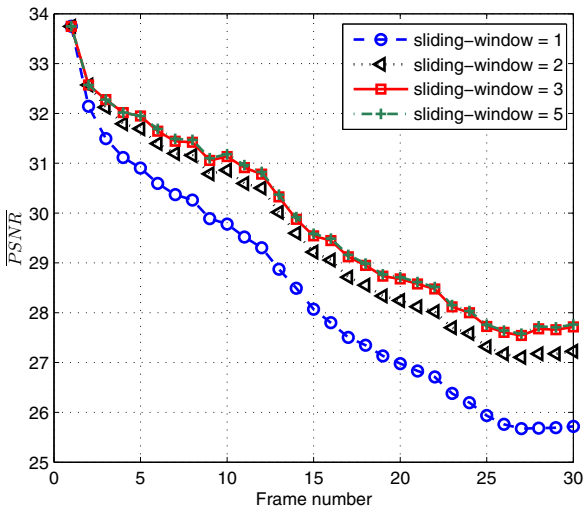

Figure 3. Effects of sliding-window updating size on the first scheme, CIF Foreman sequence; QP = 28; packet loss and delay pattern file f3.

The first experiment is carried out to validate the sliding-window updating scheme. The packet loss and delay pattern file f3 in Q15-I-16r is used, where the *physical* packet loss rate is 3.25%, and the average delay is 125 ms, the percentage of *late-arrival* packets with delay more than 150 ms is 2.14%. Figure 3 reports the results with updating sliding-window size of 1, 2, 3 and 5 frames. Note that for the case of updating sliding-window size 1, the *late-arrival* packets will not be exploited. It is interesting to note that, with large sliding-window, the frame by frame PSNR in one GOP drops more slowly than that with small sliding-window. The PSNR of the last frame in one GOP for the case of sliding-window size 3 is nearly 2 dB higher than that of sliding-window 1. This is because large sliding-window can exploit more information in the *late-arrival* packets than small ones. It is also observed that the performance of using sliding-window size 3 and 5 is equivalent, this is because for the packet loss and delay pattern file f3, a sliding-window size of 3 is enough to exploit the *late-arrival* packets. In this case, taking the system complexity into consideration, sliding-window size of 3 is more reasonable for a network environment which is similar to that of file f3.

Figure 4 reports the effects of the window size of the fixed-window scheme on the error resilient performance. When the fixed-window only contains video packets of one frame, the video quality degrades dramatically with the frame number, with the last frame PSNR of the GOP being 4 dB lower than that with fixed-window size of 4. This is because in this case, the source packet number of the RS code is not big enough to make the RS code efficient, meanwhile, for small fixed-window size, the *late-arrival* packets cannot be exploited efficiently. On the other hand, for the case of fixed-window size of 6, the video quality fluctuates significantly, and its average PSNR

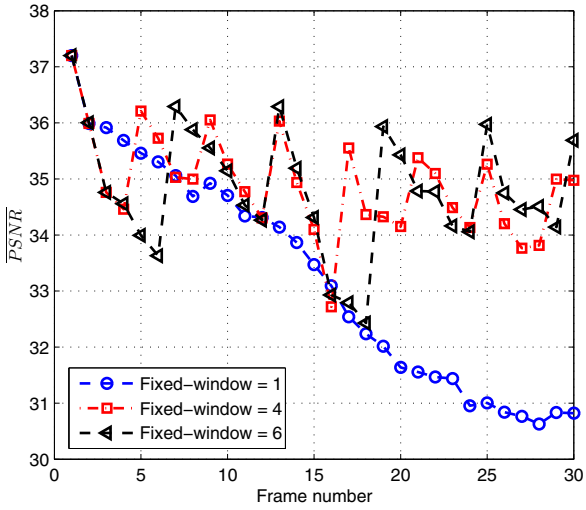

Figure 4. Effects of fixed-window size on the second scheme; CIF Foreman sequence; QP = 28; parity packet rate is 20%; packet loss and delay pattern file f3.

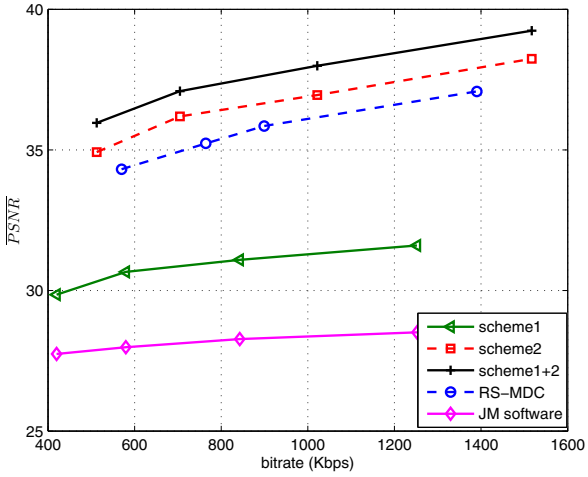

Figure 5. Average PSNR versus bitrate for the first and second scheme, jointly using the first and second scheme, RS-MDC [5] and JM software [8]; CIF Foreman sequence is used; the packet loss and delay pattern is file f3.

over the whole GOP is lower than that of fixed-window size 4. This phenomenon is because when the fixed-window includes too many frames, for the beginning frames of the fixed-window, with high probability the RS code cannot recover the unavailable packets. Based on these observations, the importance of selecting proper fixed-window size for the second scheme is demonstrated.

The average PSNR versus bitrate for the CIF Foreman sequence is shown in Figure 5. In this figure, the proposed first scheme is implemented with a sliding-window size of 5 frames; the second scheme is implemented with the fixed-window size $W = 4$. In this figure, the results of jointly using the first and second scheme are also reported; for this case, the RS parity packets are allocated using Equation 1 with fixed-window size 4 and parity packet rate 20%, whereas for the *late-arrival* updating, sliding-window with size of 5 is applied. The results for the JM software and RS-MDC [5] are also reported in this figure as a benchmark, where the *late-arrival* packets are not exploited, which means those packets with end-to-end delay larger than 150 ms are considered as dropped packets. RS-MDC [5] is selected because it is a real-time scheme which causes no additional delay in the encoding process, and its error resilient performance is competitive. It is observed that the second scheme outperforms the RS-MDC, while the the performance of the first scheme is lower than it. This phenomenon indicates that only exploiting the *late-arrival* packets without adding redundant information cannot leads to good error resilient performance. Nevertheless, although the sliding-window scheme requires no redundant information, its performance is about 2 dB higher than the JM software. Moreover, as the sliding-window scheme can exploit the *late-arrival* packets better than the fixed-window scheme, jointly using the two schemes provides the best performance among all the schemes.

IV. Conclusions

In this paper, two real-time video streaming schemes that exploiting the *late-arrival* packets have been proposed. Unlike the traditional video transmission systems, in the proposed schemes, the *late-arrival* packets are not simply discarded, but they are used to boost the reconstructed video quality at the receiver side. In order to further improve the error resilient performance of the proposed scheme, we propose to add FEC protection packets to the video stream. The simulation results have validated the proposed schemes, meanwhile the results demonstrate that the FEC protection window size and the updating window size has an impact on the system performance. Our future work would be investigating the framework to calculate the optimal FEC protection window size and the updating window size based on the network conditions, the maximum end-to-end delay and other video sequence characteristic.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No.60972085, No.60903066), the Sino-Singapore JRP (No.2010DFA11010) and National Science Foundation of China for Distinguished Young Scholars (No.61025013).

References

[1] *Recommendation ITU-T G.114, One-Way Transmission Time*, ITU, 1996.
[2] R. Zhang, S. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 6, pp. 966 –976, Jun. 2000.
[3] B. Girod, N. Farber, "Feedback-Based Error Control for Mobile Video Transmission," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1707 –1723, Oct.1999.
[4] C. Zhu, Y.-K. Wang, M. Hannuksela, and H. Li, "Error resilient video coding using redundant pictures," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 1, pp. 3 –14, 2009.
[5] T. Tillo, M. Grangetto, and G. Olmo, "Redundant slice optimal allocation for h.264 multiple description coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 1, pp. 59 –70, 2008.
[6] I. Radulovic, P. Frossard, Y.-K. Wang, M. Hannuksela, and A. Hallapuro, "Multiple description video coding with h.264/AVC redundant pictures," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 1, pp. 144 –148, 2010.
[7] T. Tillo, E. Baccaglini, and G. Olmo, "Unequal protection of video data according to slice relevance," *Image Processing, IEEE Transactions on*, vol. 20, no. 6, pp. 1572 –1582, june 2011.
[8] HHI Fraunhofer Institute, H.264/AVC Reference Software, available at http://iphome.hhi.de/suehring/tml/download/.
[9] Y.-K. Wang, S. Wenger, and M. M. Hannuksela, "Common conditions for SVC error resilience testing," *JVT document P206*, Aug 2005.