ON PACKETIZATION OF JPEG2000 CODE-STREAMS IN WIRELESS CHANNELS

Hua Cai [1] *and Bing Zeng* [2]

[1] Internet Media Group, Microsoft Research Asia
[2] Dept. of Electrical & Electronic Engineering, The Hong Kong University of Science & Tech.

ABSTRACT

The JPEG2000 standard adopts an error-resilient structure to organize its code-stream. Meanwhile, this robust structure also provides potential optimization space for packetization algorithms. In this paper, we study the optimal packetization issue for JPEG2000 code-streams over wireless channels. We first discuss the best strategy for packetizing bits from an individual code-block into channel packets. Then, we extend it to bits from a group of code-blocks. Experimental results indicate that our scheme is outperforming the normal method by significant margins.

1. INTRODUCTION

With the progress of digital communications, transmitting multimedia contents over wireless channels achieved wide commercial and research interests in the last few years. As wireless channels have low bandwidth constraints, source compression techniques should be used to reduce the amount of data to be transmitted. Recently, the JPEG2000 still image compression standard [1][2] has been established to offer a superior compression performance compared with its predecessor (JPEG). Undoubtedly, it will soon be adopted in many application areas, including the wireless multimedia communications.

However, wireless channels present many challenges for communication, due to their error-prone, bandwidth-limited, and time-varying characteristics. In Particular, the compressed bitstreams are highly sensitive to channel errors, making the multimedia communications more difficult. To combat with bit errors, data bits must be protected with channel coding algorithms. Usually, block codes (such as binary Hamming code [3], Reed-Solomon code [3], cyclic code [3], and RCPC codes [4], etc.) are used for error correction. By encoding a message block of k symbols into a new n-symbol block ($k \leq n$), the original message can tolerate a certain degree of errors. For example, four different channel coding schemes, CS-1 $\sim$ CS-4, are defined in the General Packet Ratio Service (GPRS) system [5]. The number of data bits in each scheme is 181, 268, 312, and 428 bits respectively, whereas the coded block size (including overhead bits) is fixed at 456 bits.

In above wireless communication systems, data bits should be filled into channel coding blocks for the purpose of error protection. Notice if we regard each channel coding block as one channel packet[1], this bits-filling process equals to packetizing bits into fix-length packets. The packet size here is pretty small (e.g., 181 $\sim$ 428 bits in GPRS) compared with that of the network packet (which could contain several channel packets). Normally, as in most of existing systems, bits in compressed source file are packetized into channel packets continuously without any rearrangement. This normal packetization method is quite simple and straightforward. However, since it treats the bitstream as a simple file without any structure, the error-resilient feature of JPEG2000 code-streams could not be exploited fully. As a result, the normal method is not robust to errors.

In this paper, we study the optimal packetization issue for transmitting JPEG2000 code-streams over wireless channels. As a JPEG2000 code-stream consists of bits from many independently encoded code-blocks, we study the problem upon the code-block basis. We first discuss the best strategy for packetizing bits from an individual code-block into channel packets. Then, we extend the above strategy to bits from a group of code-blocks. Good performance can be achieved by our proposed scheme, meanwhile, no overhead bits are required.

The rest of the paper is organized as follows. We first discuss the JPEG2000 code-stream structure in Section 2. Next, in Section 3 we present our packetization scheme, from one code-block to a group of code-blocks. Experimental results are shown in Section 4. Finally, some conclusions are drawn in Section 5.

2. JPEG2000 CODE-STREAM STRUCTURE

In the JPEG2000 codec, all image samples are transformed into spatial frequency subbands with the aid of Discrete Wavelet Transform (DWT). Each subband is partitioned into small source-sample blocks, known as *code-blocks*. Each code-block is then encoded independently with arithmetic coding.

After encoding, coded bits of each code-block need to be selected to meet the target bit rate, and then organized into code-stream. To perform bits organization, each resolution in the DWT is partitioned into the so-called *precincts*. Precinct are not a partition of image data, but of the compressed code-block bits, which are used to reconstruct that resolution. Compressed data from each precinct is collected into the code-stream packets. Each code-stream packet consists of a packet head and a packet body. Fig. 1 presents a typical structure of JPEG2000 code-stream, which demonstrates that a code-stream could contain several layers, while each layer consists of code-stream packets from different precincts (resolutions) and each packet is further composed of bits from code-blocks of relevant precinct.

As the code-stream packet header contains the length of bits from each code-block, bits from different code-blocks can be well synchronized given the correct header part. Moreover, in conjunction with some other error resilience tools such as the *start-of-packet* (SOP) marker and the *end-of-packet-header* (EPH) marker, the resultant code-stream achieves even stronger error resilience.

[1]To differentiate the name *packet* that is used in JPEG2000 code-stream, channel coding, and network transport layer, we call it as *code-stream packet*, *channel packet*, and *network packet*, respectively.

0-7803-8554-3/04/$20.00 ©2004 IEEE.

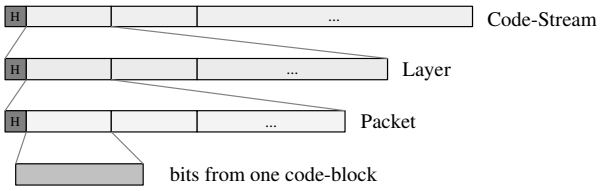

Figure 1: A typical structure of JPEG2000 code-stream (H stands for header part).

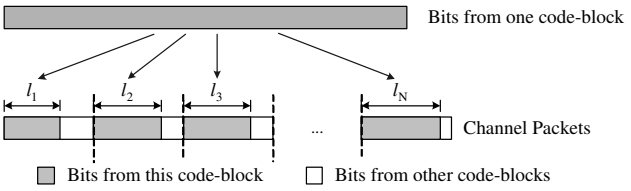

Figure 2: Packing a code-block of bits into N channel packets.

3. PROPOSED PACKETIZATION STRATEGY

We know from Section 2 that the JPEG2000 code-streams are organized with a robust structure which is resilient to channel errors. As a result, a code-stream consists of packets from different quality layers and different precincts. Each packet is further composed of bits from a group of code-blocks and some header information of each code-block. Moreover, bits from the same code-block are causally dependent (due to embedded coding), whereas bits from different code-blocks are independent to each other.

Meanwhile, this robust structure also provides potential optimization space for packetization algorithms. Let's consider the problem of packetizing a JPEG2000 code-stream into channel packets. As a code-stream consists of bits from many independently encoded code-blocks, the problem can be studied upon the code-block basis naturally. Moreover, since code-blocks are grouped into code-stream packet, we constrain the optimization within a code-stream packet. Thus, the problem can be defined as:

Given a certain code-stream packet that consists of an L_H-bits header part and a body part containing K code-blocks (with L_i-bits length for the i^{th} code-block, $1 \leq i \leq K$), how to put these bits into a list of M-bits channel packets so that the highest robustness can be achieved.

In this section, we first discuss the best strategy for packetizing bits from an individual code-block into channel packets. Then, we extend the above strategy to a more complex case, where optimization is considered for bits from a group of code-blocks.

3.1. Strategy for bits from an individual code-block

Let's analyze the expected number of contaminated bits [2] of a certain code-block, which is placed into N channel packets with l_i ($l_i \leq M$) bits for the i^{th} packet (shown in Fig. 2). Let $\hat{r}(i)$ be the number of contaminated bits when the i^{th} channel packet is wrong. Let $\hat{r}(i_1, i_2)$ be the number of contaminated bits when both of the i_1^{th} and i_2^{th} packets ($i_1 \neq i_2$) are wrong, and so on. Moreover, to simplify the analysis, we assume that all M-bits data will become useless if errors in the channel packet cannot be recovered (erroneous packet will be simply discarded in many wireless communication systems). This is because errors in the wrong packet will contaminate most of the bits of that packet.

Because of the causal dependency, a single error will contaminate all of the subsequent bits of the same code-block. Hence, the performance of a certain code-block is only determined by the position of its first erroneous bit. Therefore, we can get:

$$\hat{r}(i) = \sum_{j=i+1}^{N} l_j$$

$$\hat{r}(i_1, ..., i_n) = \hat{r}(\min(i_1, ..., i_n)) - \left[\sum_{j=1}^{n} l_{i_j} - l_{\min(i_1,...,i_n)} \right] \tag{1}$$

In above, $\hat{r}(i)$ is counted from $i+1$ since we are only interested in the number of contaminated bits, not including the erroneous bits in the i^{th} channel packet themselves.

Clearly, for a given channel packet error probability p_e, the expected number of contaminated bits of the above code-block is:

$$\mathcal{E}[\hat{R}] = \mathcal{E}[\mathcal{E}[\hat{R}_n]] = \sum_{n=0}^{N} \mathcal{E}[\hat{R}_n] \times p_e^n \times (1-p_e)^{N-n} \tag{2}$$

where $\mathcal{E}[\hat{R}_n]$ denotes the expected number of contaminated bits when there are n out of N channel packets are wrong,

$$\mathcal{E}[\hat{R}_0] = 0$$

$$\mathcal{E}[\hat{R}_1] = \sum_{i=1}^{N} \hat{r}(i)$$

$$\mathcal{E}[\hat{R}_n] = \sum_{i=1}^{N-(n-2)} \left\{ \hat{r}(i) \times \left[1 - n + \prod_{j=i}^{i+n-2} (N-j) \right] \right\} \times n, \quad \text{for } n \geq 2. \tag{3}$$

From Eqn. (3), it can be seen that $\mathcal{E}[\hat{R}_n]$ is only determined by $\hat{r}(i)$. Actually, $\hat{r}(i)$ is more dominating than $\hat{r}(i+1)$, since $\hat{r}(i)$ is weighted unequally in $\mathcal{E}[\hat{R}_n]$ (for $n \geq 2$) when i increases. Therefore, minimizing the expected number of contaminated bits, $\mathcal{E}[\hat{R}]$, is equivalent to minimizing $\hat{r}(i)$ for all i ($1 \leq i \leq N$).

Note $\hat{r}(i) = \sum_{j=i+1}^{N} l_j$, thus minimization of $\hat{r}(i+1)$ is based on that of $\hat{r}(i)$. In other words, when the sum of $\hat{r}(i)$ is minimized, each $\hat{r}(i)$ can also achieve its minimum value. Therefore, we can use the sum of $\hat{r}(i)$ as the objective function for optimization:

$$\mathcal{J} = \sum_{i=1}^{N} \hat{r}(i) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} l_j = \sum_{i=1}^{N} (i-1) \times l_i \tag{4}$$

It is clear from Eqn. (4) that, for a best packetization strategy (with the smallest $\mathcal{J}$), the number of bits filled into each channel packet must be in a non-increasing fashion, i.e., $l_i \geq l_{i+1}$. Since the maximum number of data bits for a channel packet is M, the best strategy for L-sized bits from one code-block should be:

[2] Contaminated bits are the bits corrupted by some erroneous data. These bits themselves are correct, however, they cannot be decoded correctly since some of their depended data are wrong. The less the contaminated bits, the higher the error robustness.

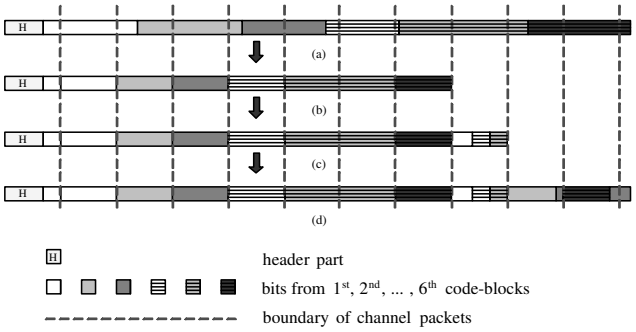

Figure 3: An example of packetizing a packet of bits.

Put the first $M \times \lfloor L/M \rfloor$ bits to $\lfloor L/M \rfloor$ channel packets, with M bits for one packet; and for the remaining $L\%M$ bits, put them to one packet (where $\lfloor x \rfloor$ represents the positive part of x, and $x\%y$ represents the remainder of x after being divided by y (x and y are both integers)).

3.2. Strategy for bits from a group of code-blocks

We've discussed the best packetization strategy in Section 3.1 for bits from one code-block. When using this strategy directly to packetize a code-stream, each code-block of bits can achieve the highest robustness to channel errors. However, as the last channel packet for each code-block of bits is often not fully filled (i.e., the rest of data bits of that channel packet will be wasted), it is not efficient in bandwidth utilization. Hence, to attain high robustness as well as efficient bandwidth utilization, we extend the above strategy to bits from more code-blocks.

Moreover, at the receiver end, the packetization process should be performed inversely to extract bits of each code-block from channel packets. Obviously, the receiver must know the length of bits from each code-block as well. Note in JPEG2000 the lengths information have already been included in the header of each code-stream packet. To obtain the lengths information, the header part should be packetized normally without any rearrangement. Once the header part is extracted, the receiver can repeat the packetization process inversely to know the exact position of each code-block in the channel packets. After that, bits are reorganized as that of the original code-stream packet, and finally, source decoding will be performed.

We consider the optimization within a code-stream packet since it is an independent unit containing bits from a group of code-blocks and their corresponding lengths information. When bits from a group of code-blocks are packetized, each code-block of bits are first treated individually using the best strategy proposed before. After that, some channel packets are fully filled with bits from the same code-block, while others are partially filled only. We then search through these partially filled packets iteratively, try to find some combinations of them that can be merged into a single packet. Note for a large group that contains many code-blocks (e.g., several hundred code-blocks per code-stream packet in JPEG2000 if a large precinct size is chosen), it is very likely that many of partially filled packets can be merged. Ideally, if all of them are merged together, then every code-block of bits could achieve the highest robustness. If there are still some partially filled packets left, then some of them have to be split into two or more parts to be merged with other partially filled packets. Finally, for those split parts, their lengths should be adjusted to satisfy the non-increasing constraint, as specified in Section 3.1.

The procedure of packetizing a code-stream packet is summarized below, which consists of six steps for optimizing the overall performance:

Step 1 Start from the first channel packet, put the header part continuously.

Step 2 If the channel packet containing header is not full, then put bits into that channel packet continuously until either that packet is full, or there are no bits left.

Step 3 If there is any remaining code-block (with length L) larger than the packet size M, then put its first $M \times \lfloor L/M \rfloor$ bits to $\lfloor L/M \rfloor$ number of packets.

Step 4 Search through the remaining code-blocks, try to find n (initially, $n = 2$) code-blocks with the sum of remaining lengths equal to M, and put the found code-blocks into one packet. Repeat this step until there is no code-blocks match to the condition. Then increase n by 1 and repeat this step until n reaches a pre-defined value (e.g., when $n > 5$, stop searching).

Step 5 From the remaining code-blocks, find the largest one first, and then keep on finding the smallest one until the sum of lengths exceeds M. Split the last code-block to fit M, and put the found code-blocks into one packet. Repeat this step until there are no bits left.

Step 6 For bits put in Step 5, if there is any code-block whose bits are put to different packets without satisfying the non-increasing length property, swop the two parts.

To better explain the above procedure, an example of packetizing a code-stream packet of bits is shown in Fig. 3. In the example, the code-stream packet consists of a 88-bits header and body bits from 6 code-blocks, each with length 216, 240, 192, 168, 296, and 232 bits, respectively (shown in Fig. 3-(a)). These bits should be packetized into a list of 128-bits channel packets. If the bits are packetized normally, it is the same as in Fig. 3-(a). On the contrary, through the proposed procedure, the packet header is first put, followed by the first 40 bits of the 1^{st} code-block. Then, the remaining bits of each code-block are put into integer number of packets, as shown in Fig. 3-(b). Next, through searching, the remaining bits of the 1^{st}, the 4^{th}, and the 5^{th} code-blocks are found matching to the packet size, hence they are put into one packet, as shown in Fig. 3-(c). Finally, the left bits are put according to Step 6, as shown in Fig. 3-(d). We also compared the value of the objective function $\mathcal{J}$ defined in Eqn. (4) for the above example. For the normal packetization, the value is 1056; whereas for the proposed packetization, the value is 808 only.

4. EXPERIMENTAL RESULTS

We simulate a Binary Symmetric Channel (BSC) with Bit Error Rate (BER) around 10^{-2}, which is typical for a bad channel condition in wireless communications. To combat with the bit errors, channel coding algorithm based on the Reed-Solomon code of 8 bits/symbol are used in the experiments. Three standard test images *Lena*, *Goldhill*, and *Peppers* (512×512, monochrome, 8 bits/pixel) of different features are used for testing.

Table 1: Comparison of average PSNRs when BER equals 10^{-2}.

Image	Code-block size	Normal Pack	Proposed Pack
Lena	32×32	29.76	30.52
	16×16	31.17	32.13
Goldhill	32×32	28.82	29.36
	16×16	29.96	30.51
Peppers	32×32	29.29	30.10
	16×16	30.84	31.45

In our experiments, each image is first encoded to a size corresponding to 0.71 bits/pixel with the JPEG2000 codec [6]. As the channel is quite noisy, two small code-block sizes, 32×32 and 16×16, are used for source coding for the purpose of error resilience. Since headers are important, they should be protected with higher priority. This is attained by merging all of the packets' headers together with the main header and the tile part header (header merging is allowed in JPEG2000), and then protecting them with a channel coding scheme RS$(63, 39)$. The packets' bodies are protected with RS$(63, 45)$. After channel coding, the total rate (including source and channel rate) equals 1.0 bit/pixel.

Since all headers are merged together, the whole code-stream can be looked as one code-stream packet naturally. Packetization is thus performed on this big packet. Notice the header part is protected with RS$(63, 39)$ while the body part is protected with RS$(63, 45)$. Equivalently, the channel packet size is $39 \times 8 = 312$ and $45 \times 8 = 360$ bits, respectively. For our proposed scheme, the body part bits are packetized according to the proposed procedure. As for the normal packetization, the code-stream is filled into two kinds of channel packets continuously without any rearrangement.

The average PSNR values (averaging over $50,000$ independent runs) of the two packetization schemes over a BER range are compared in Fig. 4 (for space limitation, only the results for *Lena* are provided). It can be seen that, for the *Lena* image, the improvement is about 1.0 dB when BER $\geq 10^{-2}$. Quantitatively, the average PSNR values when BER equals 10^{-2} are shown in Table 1. These results indicate that our scheme is outperforming the normal method by significant margins.

5. CONCLUSIONS AND DISCUSSIONS

In this paper, we study the optimal packetization issue for transmitting JPEG2000 code-streams over wireless channels. We first discuss the best strategy for packetizing bits from an individual code-block into channel packets. Then, we extend it to bits from a group of code-blocks. Good results have been observed from the testing. Meanwhile, no overhead bits are required.

Additionally, the proposed packetization scheme could also work for other standards such as JPEG and MPEG-2, etc, as long as their produced bitstreams can be separated into many independent parts (e.g., slice structure for MPEG-2). Of course, the corresponding length of each part should also be delivered.

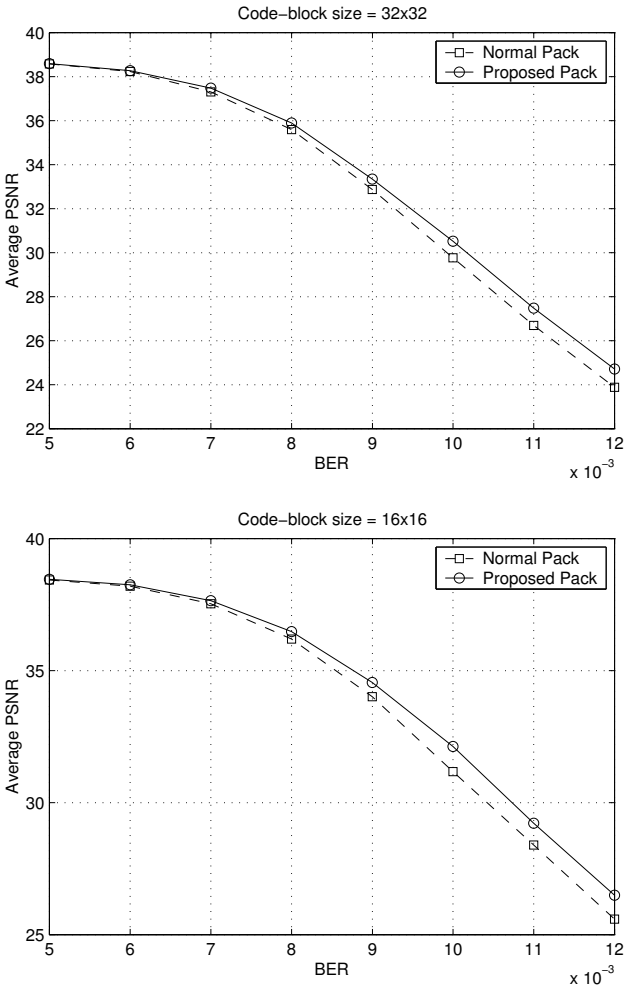

Figure 4: Comparative evaluation of the proposed packetization scheme for the *Lena* image with two code-block sizes.

6. REFERENCES

[1] David Taubman and Michael Marcellin, "JPEG2000: Standard for interactive imaging," *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1336–1357, Aug. 2002.

[2] ISO/IEC 15444-1, "Information technology–JPEG2000–Image coding system–Part 1: Core coding system," 2000.

[3] John G. Proakis, *Digital communications*, McGraw-Hill, 4th edition, 2001.

[4] Joachim Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 389–400, Apr. 1988.

[5] 3GPP TS 03.64 v8.8.0, "Overall description of the GPRS radio interface; Stage 2," Apr. 2001.

[6] ISO/IEC JTC1/SC29/WG1, "JPEG2000 Verification Model 7.2," May 2000.