Optimal Caching Mechanism for JPEG 2000 Communications

Marcela Iregui, Philippe Chevalier and Benoît Macq *

ABSTRACT

In this paper we study an optimal transmission mechanism along with a memory caching strategy in a server-client architecture for large JPEG2000 (J2K) images browsing. The system takes advantage on the scalability introduced by J2K codestreams. The idea is that users accessing the server could navigate in a seamless way in one or several images. They could change the displaying requirements, ask for a desired resolution, quality or region of interest (ROI) with no need to receive the whole codestream. We propose a smart data streaming strategy to minimize the latency time and support interactive browsing. Our strategy takes into account several constraints as delays, memory, bandwidth, etc., for permitting fast adaptation to image browsing with optimal employment of resources.

1 Introduction

Consider a server-client architecture where the server holds a JPEG2000 image database. The client can access and browse images in a seamless way taking advantage of the J2K codestream scalability [1], [4]. At the beginning, one image or a ROI is displayed in low quality or low resolution according to the user preferences. If a request from the client is received (i.e. next quality, resolution or region), the server parses the codestream, seeks the relevant packets and sends them. However, as explained in [3], limitations on the available bandwidth reduce the transmission rate and only a portion of the image can be displayed at a particular time. Moreover, in dynamic browsing interfaces, the displayed image regions shall change whenever the user changes the request. In consequence, the response time must be minimized and the most important packets sent at each time. The selection of the relevant J2K packets must take into account the complex characteristics of the JPEG2000 codestream, i.e. J2K packets have different associated utilities and dependencies.

*M. Iregui and B. Macq are with the Communications and Remote sensing laboratory and P. Chevalier with the POMS unit, all at the Université Catholique de Louvain, Belgium. E-mail: iregui@tele.ucl.ac.be. This work was funded by the IST project PRIAM (IST-28646).

If the available resources are underused, the server could predict the user behavior in the session and decide accordingly to send additional data as proposed by [3]. However, in this article we don't address this case and we will consider only constrained cases. If the server deals with several clients with different constrains in delays and bandwidth, a data streaming strategy is necessary to optimize the data flow and a caching mechanism to hold data at server and client sides. A similar approach is presented in [2] for video streaming. In this work we make allowance for rate constraints and we propose an optimal data stream mechanism based on the optimization of an utility function for J2K packets streaming.

2 System Model

The system consists of a server, which encodes and archives images, and a client, which requests some image data, decodes it and manages the displaying as shown in Figure 1. At the server side, images are compressed using the J2K algorithm. The parser is an intelligent entity in the server, which decides what portion of the codestream must be sent to meet the user requirements. At the client side the received data are decoded and the results held within a cache memory, ready to be used by the viewer. In order to allow a progressive transfer of coded data, the system will use an interactive protocol adapted to JPEG 2000.

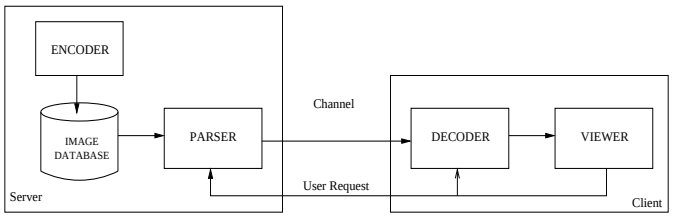

Figure 1: System block diagram.

2.1 Encoder

Images are encoded using the JPEG 2000 encoding algorithm [1], [4]. One of the most powerful features

of JPEG 2000 is the scalability of the generated codestream i.e. it can be arranged by quality (SNR), resolution, component and/or position. By SNR, The quality layers in the codestream are decoded progressively giving at each time a displayed image improved in quality. By resolution, the image can be accessed by levels, i.e. starting from a thumbnail, at each level the displayed image is bigger. Scalability by position is possible because the codestream is composed of packets that contain the coded data for a given precinct (group of codeblocks) which can be mapped onto a specific image region. Five possible progression orders are allowed:

1. Layer-Resolution-Component-Position,
2. Resolution-Layer-Component-Position,
3. Resolution-Position-Component-Layer,
4. Position-Component-Resolution-Layer,
5. Component-Position-Resolution-Layer.

Another important feature of J2K is that it is possible to predefine a ROI encoded with better quality than the background, and placed at the beginning of the codestream.

2.2 Parser

The parser is the intelligent entity of the server. It looks for the relevant parts of the codestream according to the user requests and the channel bandwidth. It maintains a model of the decoder state in order to determine the next data to send. The Parser fills a buffer with a maximum quantity of data that maximizes the user satisfaction. The buffer size depends on the memory, bandwidth and delay constraints imposed by the network or the client.

2.3 Decoder

The decoder is a J2K "scalable" decoder, able to decode progressively portions of the codestream. In order to actualize at each time the decoded data, a copy of the received packets is maintained in the decoder cache memory. At the output the decoder will put the image information in an image cache memory ready to be used by the viewer.

3 Data streaming strategy

As said above, the parser must be able to seek the relevant parts of the J2K codestream according to the user requests. The idea is to maximize the user satisfaction and efficiently manage resources: cache memory and/or bandwidth. By user satisfaction we mean displaying the required image region at the maximum quality and resolution in a minimum delay. To minimize the delay, the available bandwidth must be optimally used in order to maximize the rate in each transmission interval. The problem of budget-constrained allocation in a bandwidth limited transmission can be formulated as:

if G_i is the state of the client cache at time i, we want to find the optimal set of J2K packets S_i to be sent at interval i, such that the total rate is such that:

$$R_i = \sum_{k \in S_i} r_k \leq B_i \tag{1}$$

and will produce the largest utility $U(G_i \cup S_i)$.

In equation (1) r_k is the rate of the packet k and B_i is the available bandwidth (or cache memory). R_i is the effective bandwidth which is the sum of rates of the J2K packets that maximize the resources and the utility function for the interval i. The state of the client cache G_i is updated by: $G_{i+1} = G_i \cup S_i$

This problem is an extension of the 0-1 knapsack problem, where the utility metric will determine the value of each packet and the bandwidth (memory) constraint corresponds to the capacity of the sack. The 0-1 knapsack problem can be easily solved using greedy algorithms or dynamic programming [6].

3.1 Utility function

The definition of the utility metric, U, is based upon the user preferences. This function depends on the packet dependencies and the way the user navigates in the image. Thus:

1. We suppose that at the beginning of the session the user selects a preferred progression order. Depending on the selection, a weight is given to each variable of the display. For example, if the user selects a Layer-Resolution-Component-Position progression order, it means that he or she wants an overview of a overall region, giving priority to position over components, resolutions and layers in that order. In this case, the user considers that the important thing is to have an overview as soon as possible of the selected region or the whole image.

2. The packet value is given by a measure of the importance of its corresponding layer, resolution, component and position. In the case of layers, lower layers have priority over higher ones,the same for resolutions and components. For example, packets from higher layers have 0 value if the corresponding lower layers have not been sent or previously selected as candidates.

3. For the position, the criterion we use is related with the relevance of the given precinct regarding the region of the image selected by the user, and additionally by its location between the selected area. We give more weight to such precincts located in the center of the region because normally, users tend to focus at the center of a zoomed region rather than at the borders.

If we define $s \subset S_i$ as the set of packets that has been already selected to be sent and $k_{l,r,c,p}$ as the packet corresponding to layer l, resolution r, component c and precinct p, the utility function of a packet k is given by:

$$U_i(k_{l,r,c,p}) = u_{i,l}u_{i,r}u_{i,c}u_{i,p} \quad (2)$$

In (2), $u_{i,l}$, $u_{i,r}$, $u_{i,c}$ and $u_{i,p}$ represent the utility of the layer, resolution, component and precinct respectively. Thus, if $x > 0$ is a measure of the distance from the center of the ROI, $a_i \in \{1, 2, 3, 4\}$ the weight given to each dimension by the user and l,r,c the weight of each layer, resolution or component; we define utility variables as:

$$u_{i,l} = \begin{cases} 0 & \text{if } k_{l-1,r,c,p} \notin (G_i \cup s),\ l > 0, \\ a_1 l & \text{otherwise} \end{cases} \quad (3)$$

$$u_{i,r} = \begin{cases} 0 & \text{if } k_{l,r-1,c,p} \notin (G_i \cup s),\ r > 0 \\ a_2 r & \text{otherwise} \end{cases} \quad (4)$$

$$u_{i,c} = \begin{cases} 0 & \text{if } k_{l,r,c-1,p} \notin (G_i \cup s),\ c > 0 \\ a_3 c & \text{otherwise} \end{cases} \quad (5)$$

$$u_{i,p} = \begin{cases} 0 & \text{if } p \notin \text{ROI}(i) \\ \frac{a_4}{x} & \text{otherwise} \end{cases} \quad (6)$$

The parser keeps track of the client access history to determine which data must be sent. The server receives feedback from the client when data has been received or discarded due to memory constraints. This mechanism allows adaptation to the navigation and server adaptation to the client status.

If no track of the client status is held, the client must always drive the navigation, so each time, a request must be sent indicating specifically the needed packets. For some applications this mechanism is not optimal because it supposes the client downloads a codestream index for each image he is dealing with, even if he's just looking at low resolutions when seeking a given image in a database.

4 Client memory caching mechanism

Clients accessing the server can have an heterogeneous set of rates and distinct hardware configurations, decoder capabilities and memory capacities. Thus, for constrained applications a memory caching strategy is necessary to manage data. Due to the dependencies between packets, when cache storage is full, the decoder must determine which packets can be discarded to make room and which are mandatory to decode the next incoming data. This decision depends on the user navigation in the image. Several caching techniques, taking into account the importance of each packet, have been proposed. A soft caching scheme has been proposed by [5], which takes advantage of the a priori information of each image and resolution. However, this globally optimized cache allocation strategy is not necessarily optimal for particular local access operation.

For some applications, the cache management must be done dynamically and we need to adapt the resources of the client during the navigation process. This will permit an adaptation to the navigation and an optimal sharing of resources between several images (i.e. a medical exam). So, the caching mechanism used for our system is based on the soft caching for dynamic access proposed by [3].

5 Simulation results

In this section we illustrate the results showing a simple example in a LRCP navigation scenario of a 1792x2816 image. The image is J2K compressed with two quality layers, 3 resolution levels and 77 precincts/resolution. Supposing that at time i the client has received the whole layer 0, so he has an overview of the image. Knowing that he has limited resources, he chooses a region to zoom in as shown in figure 2(a) and sends a request to the server. The server has two possibilities, either it sends the corresponding packets following the raster order of the precincts as they are in the codestream or sends the packets which efficiently utilize the channel bandwidth given by the utility function presented in section 3. We show in figures 2(b) and 2(c) the same image region displayed at the t_{i+1} instant without and with optimization respectively. In image 2(b) we can see that the top has been updated but the center which is normally the target region is still fuzzy. Hence, from these two pictures we can infer that normally the user will be more satisfied by the second one. If he is just interested in identifying the subjects in the picture, in the first case he must wait more to get the packets he is interested in.

6 Complexity issues

It is well known that dynamic programming is time demanding when data sizes are very large [6]. For the 0-1 Knapsack problem, the classical solution is $O(nb)$, where n is the number of packets and b the available bandwidth. If b is extremely large, the algorithm can take a lot of time and even with no consideration of the packet dependencies which introduce an additional delay, this algorithm may easily become un-feasible. However, in cases where b is large, it is not necessary an optimization algorithm because the user can get data very fast. Hence, it should be selectively applied in cases for which the bandwidth is very limited, for it is here that utility is maximal. So, because our algorithm is optimal when focusing the center of a region, what we propose is to use it only in constrained cases where the bandwidth is two times smaller than the codestream size for a given image

(a) t_i (b) t_{i+1} (c) t_{i+1}, optimal

Figure 2: Boat2([4]) image displayed at t_i and t_{i+1}

or region. If even in that conditions the computational time is critical, there exist greedy solutions, which are less time demanding but give sub-optimal solutions to the problem.

7 Conclusions

In this paper we have presented a strategy to manage the data streaming in a server-client architecture to allow seamless navigation in J2Klarge images by efficiently managing resources. The presented data flow mechanism allows the minimization of the server response time according to the user preferences. This method is adaptable to different clients with different resources. In this way the constrained applications will receive the maximum quantity of data according to their capacities. This technique should be combined with a cache memory management at the client side to allow the optimal employment of resources. We propose a dynamic mechanism to control the memory caching at the client side. This data flow mechanism can be complemented with a technique to predict the user behavior in the navigation in order to prefetch packets to the client.

References

[1] Martin Boliek, Christipoulos Charilaos, and Eric Majani. *JPEG200: Part I Final Draft International Standard.* ISO/IEC 15444-1, March 2001.

[2] Jia-Ru Li, Xia Gao, Qian Leiming, and Vaduvur Bharghavan. Goodput control for heterogeneus data stream. In *Proc. of the 10th Inal. workshop on Network and Operating System Support for Digital Audio and Video*, NOSSDAV, June 2000.

[3] Chengjiang Lin and Yuan F. Zheng. Fast browsing of large scale images using server prefetching and client caching techniques. In *Applications of Digital Image Processing XXII*, SPIE, pages 376–387, July 1999.

[4] David Taubman and Michael Marcellin. *JPEG200 Image Compression Fundamentals, Standards and Practice.* Kluwer Academics Publishers, November 2001.

[5] C. Weidmann, M. Vetterli, A. Ortega, and F. Carignano. Soft caching: Image caching in a rate-distortion framework. In *IEEE conference on Image Processing (ICIP 97)*, 1997.

[6] Laurence Wolsey. *Integer programming.* Wiley Interscience, 1998.