An Efficient Technique for Remote Browsing of JPEG 2000 Images on the Web

J.P. Ortiz, V.G. Ruiz and I. García

Abstract— **This paper presents an evaluation of the existing techniques for remote browsing of JPEG 2000 images and a new proposal that solves many of the observed deficiencies of these techniques for Web based systems. Up to now the use of the JPEG 2000 standard is not being very popular because of the existing solutions are complex and sometimes inappropriate. The main goal of this work is to offer an optimal and easy solution for this kind of systems.**

Keywords— **JPEG 2000, JPIP, HTTP, data-bin.**

I. Introduction

The JPEG 2000 compression standard, thanks to its powerful features, allows to implement efficient systems for remote browsing of images. But nowadays, this kind of systems is not very popular, specially on the Web, because of the main disadvantages of the existing solutions. It is necessary a new approach that increases the JPEG 2000 standard popularity on the Web, making possible to exploit its desirable features in order to be able to browse efficiently remote images.

The rest of this paper is structured as follows: in the Section II, we present the fundamentals of the JPEG 2000 compression standard; in the Section III, the existing techniques for remote browsing of JPEG 2000 images are presented, explained and analyzed, showing their main advantages and disadvantages; in the Section IV, it is exposed our proposal, detailing its operation; in the Section V, we analyze and evaluate our proposed system, comparing ir with the existing solutions; finally, in the Section VI, we conclude with the pertinent conclusions.

II. JPEG 2000

JPEG 2000 is a recent image compression standard [1], emerged from the Joint Photographic Expert Group (JPEG) within the International Standards Organization (ISO). It is based on the Discrete Wavelet Transform (DWT) and the Embedded Block Coding with Optimized Truncation (EBCOT) [2]. One of its main features, apart from the efficient compression, is its high scalability, allowing, for example, quality, resolution and spatial scalability. This feature makes it ideal for remote browsing of images, where the client needs to view an specific region of a remote image, at a necessary resolution level (WOI).

In this section we explain basic concepts related to the JPEG 2000 standard in order to be able to present properly the different techniques for remote browsing of JPEG 2000 images.

This work was supported by the MCyT under contracts TIC2002-00228.

Since the DWT is used, the original image is decomposed in $1 + 3n_r$ spatial frequency subbands, where n_r is the number of resolution levels. The EBCOT paradigm requires to divide each subband in rectangular blocks with the same size, called "code-blocks". The code-blocks are grouped into rectangular groups called "precincts", whose dimension can be change in each resolution level.

Each code-block is independently coded into an embedded bit-stream. The code-block bit-streams are divided in n_l segments of different length, where n_l is the number of quality layers.

The compressing process is applied independently to each component of each tile. Tiling will not be explained in this paper, and we will assume the images do not have tiles, that is, they will have only one tile, occupying the entire image. The tiling is only used in specific applications because, in general, it decreases the quality of the reconstruction.

The bit-streams of each code-block of a precinct, associated with the same quality layer, are grouped in a "packet". In this way, every packet is associated with a tile, a component, a resolution level, a precinct and a quality layer. The generated packets when compressing an image can be interleaved in one of the progression orders defined in the standard: LRCP, RLCP, RPCL, PCRL and CPRL. In relation to the progression names, "L" is for quality layer, "R" is for resolution level, "C" is for component and "P" is for precinct.

The Part 1 of the JPEG 2000 standard defines the core coding system, establishing the basic format of a compressed image, the code-stream. The code-stream contains all the packets generated by the compressing process and a set of markers that can mark specific sections. If a marker contains additional information, is called marker segment. Both markers and marker segments are named with a 3 letter name.

In the Figure 1 it is shown the basic structure of a code-stream. There is a main header (composed of markers and marker segments) and the packets of all the image tiles. The set of packets of every tile is partitioned in subsets called tile-parts. Each tile-part has a header too.

A file with a simple code-stream stored normally has the ".j2c" extension, and it is commonly called "raw file". In order to provide more functionality and flexibility, in the Part 2 of the standard [3] it is defined a more complex image file format, that is based on "boxes" (JP2 boxes) of information. In this format, a code-stream would be included within a box. Other defined box is, for example, a box

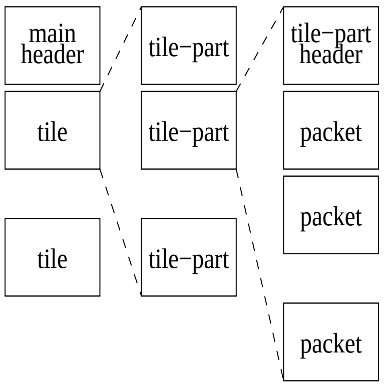

Fig. 1. Code-stream structure.

for the palette information (there is no defined any marker segment for storing this information). The primary image file format based on boxes is the JP2 file, but there are another image file formats defined in the standard following the same structure. The image files with a format based on boxes are typically called "JP2-family files". The JP2 boxes can be super-boxes, containing another boxes.

III. Remote browsing techniques

In this section we are going to review the main existing techniques for remote browsing of JPEG 2000 images to be able to evaluate and compare at the end the proposed and implemented system with these. Is is important to bear in mind that the required aim is to browse images remotely on Internet, mainly on the Web. Some techniques here presented were designed for a generic use, so they have several disadvantages in this context, although in many others they are very efficient.

A. JPIP

JPIP is a protocol defined in the Part 9 of the JPEG 2000 standard [4][5], and is a evolution of the JPIK protocol proposed by D. Taubman. At the beginning, although the JPEG 2000 still image compression had many features oriented to implement efficient applications for remote browsing, the standard did not define any mechanism to do it. Later, in the Part 9, it was defined the JPIP protocol with the purpose of covering this kind of applications.

In the Figure 2 we can see the typical structure of a system for remote browsing of images, using the JPIP protocol. The system is a typical client/server system, where the clients request WOIs of remote images to the server, and this sends the necessary information for reconstructing them. In the client side, there are four modules: i) the user interface, or browser, which transforms the user interactions into WOI requests, ii) the JPIP client, which receives the WOI requests form the browser and sends them to the server, through a communication channel, iii) the client cache, where the JPIP client stores the information received from the server, and iv) the module which is decompressing and rendering iteratively the cache data for reconstructing the requested WOIs, and sending the imagery to the browser for showing it to the user. In the server side, there are JPEG 2000 images and a JPIP server that processes the WOI requests, extracts the appropriated information from the local images, transcoding them if necessary, and sends it to the client. It is possible, although it is not required, that the server maintains a cache model of the client to avoid send information already sent. This cache model can be explicitly modified by the client.

The JPIP protocol has been designed for being implemented over almost any other protocol, like HTTP, TCP or UDP, although the most commonly used protocol is HTTP, due to that the JPIP requests and responses can be easily encapsulated in HTTP messages. It makes that the communication between JPIP clients and servers can be established on the Web infrastructure, passing through the possible existing Web proxies.

Currently, the JPIP system is in general the most efficient option for implementing applications for remote browsing of images, exploiting the features offered by the JPEG 2000 standard. But, if we need to exploit these features in a common Web infrastructure, that is, placing the images in a Web server, this system has several disadvantages. The first disadvantage is the requirement of a specific server when, currently, there are not implementations for all the computer architectures. This does not occur if you use a standard server, like a HTTP server, because you do not have any problem to find an implementation and, in many cases, the implementation is free (like the Apache server) A specific server is not always possible in Web systems, because in many cases we have to use an external server machine and, either the owner refuses a strange and not well-known external server or the security rules used do not allow to use other port different from the standard ports (HTTP, SMTP, etc.).

Another disadvantage of the JPIP protocol is the processing overload of the server machine. In general we will be able to use only one computer, so we would have to integrate the Web server and the JPIP server in the same machine. The JPIP philosophy admits that the images stored in the server were of any type (with or without tiles, with any precinct dimension, with any kind of progression, etc.), so the server must transcode "on-the-fly" the images to transform them in a more efficient format for transmission. This overload is not very important for a machine without any other important process, but in a computer intended for Web serving, where there is a Web server with other high processing overload, this is unacceptable.

B. Indexed JP2-family files

As has been explained, the data of a compressed image is partitioned in bit-streams called packets, with diverse lengths. If we want to reconstruct only a certain region of an image, we need to access to those necessary packets for this region. To do it, we

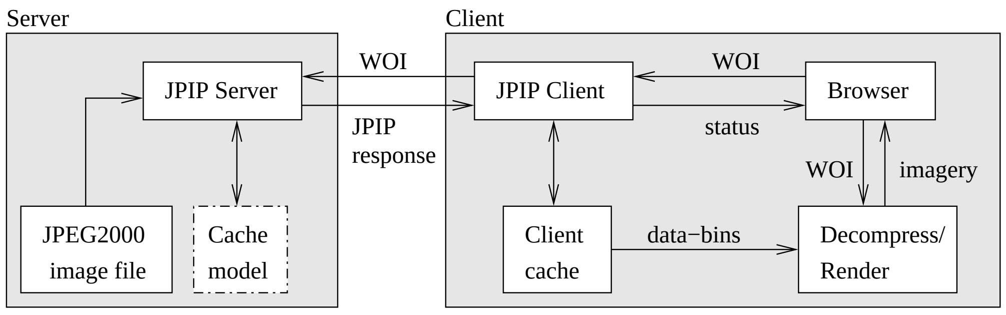

Fig. 2. JPIP structure.

have to know the length of these packets and their offset within the compressed image file. It is solved if we use the JPIP architecture, because the server has the images locally stored and can read them to extract the required packets, according to the requested WOI, and send them to the client. But, if we want to use other different protocol, we will have to index the image file. In this fashion, we could read firstly the index, process it to determine what packets are necessary, and read them independently.

Neither the Part 1 nor Part 2 of the JPEG 2000 standard define the formal way to index a compressed image. Viewing this lack, in the Part 9 of the standard, a part from the JPIP protocol, a collection of boxes for JP2-family files was defined to allow us to include indexes in image files. With these indexes it is possible to obtain the length and offset, within the image file, of the different parts of it, including the packets. One of the main purposes of these boxes was to ease the implementations of JPIP servers.

Furthermore, these kind of boxes can be used to implement applications for remote browsing, using other different protocol, like HTTP, that offers the possibility of access randomly to the image. Using the index boxes, the client would read firstly the index boxes stored in the JP2-family file, would analyze it, and finally would read the precise parts.

To use this solution with other protocols different from JPIP has two important deficiencies: i) there is a big overload of information required to be transmitted, and ii) are necessary a considerable number of round-trips for retrieve the index and image data.

On average, removing the initial read data required to find the main box and parsing the two initial sub-boxes, to read the index of the packets of a tile, in the simplest case, is necessary to read at less $(17 + 8n_{pc})n_c$ bytes, where n_c is the number of components and n_{pc} is the number of packets of every component. It is quite suboptimal.

Finally, the creation of the index boxes requires an external application that parses an already created image file, because the most of the current compressors, so free as commercial, do not support this indexing process.

C. Deshpande and Zeng proposal

The proposal of Deshpande and Zeng [6] is perhaps the solution more specifically oriented to Web systems. It is completely based in the HTTP/1.1 protocol [7]. Their solution consists in creating an external index file, associated to every JPEG 2000 image file. The external index file would be referenced, for example, in a Web page, and a client, when downloads it, can build in memory the appropriate image index for remote browsing those parts of interest. By byte-ranging, feature offered by HTTP/1.1, the client can request to a generic Web server the necessary parts.

This solution has the following desirable features: i) it is very flexible, being able to index almost all kind of images, from simple raw files to complex JP2 family files, ii) it needs a simple implementation for clients, and iii) has a fully integration with a Web system, being able to use easily a generic Web server and to be included in a common Web page.

The client needs with this solution to do only one round-trip for retrieving the index, instead of in the solution with indexed JP2-family files, where the client has to do many round-trips to only find the index. However, the client has to read the entire index before to request any part of the image. The initial wait before to show any reconstruction to the user is proportional to the connection speed and the size of the index file. Deshpande and Zeng propose several formats for the index file, from the simplest and smallest to the most complex and biggest. As much complex the index file is, much big is, but more efficiency and fast is the process of the client to calculate the interesting parts of the image. The simplest index file would contain just the main header, tile-part headers, packet headers (if any), and the length of all the tile-parts and packets.

The authors estimate that the average size of the index file would be around the 1% of the original code-stream. So, for example, for a raw JPEG 2000 image file, of 5MB, a client that wanted to remote browsing it through a connection with a speed of 4KB/s, it would need the client to wait around 13 seconds downloading the index file, before to be able to request any image region. It is unacceptable.

The use of external index files has in addition the same problem of indexed JP2-family files, that is, the storing overload in the server, although for the simplest index format, this overload is smaller.

For index JP2-family files, once it is found the appropriated index, we do not need to read it entirely

TABLE I
ADVANTAGES/DISADVANTAGES OF EXISTING TECHNIQUES.

Technique	Advantages	Disadvantages
JPIP	- Minimal additional information. - Transmission efficiency. - It is not needed an index. - Accepts any kind of JPEG2000 image file. - Server has local access to image files and can transcode them if it is necessary.	- Need for a specific server. - Few server implementations currently available. - Processing overload in the server. - Web proxy caching not exploited. - Not fully compatible with Web systems.
Indexed JP2-family files	- Possibility of using generic servers. - Index included within the image file.	- High transmission overload due to the index structure. - Too many round-trips. - No raw files support.
Deshpande and Zeng proposal	- Specifically designed for HTTP/1.1 protocol. - Possibility of indexing any kind of image file. - Requires a simple implementation.	- Requires an external index file. - It is necessary to retrieve the index file before to do any byte-range request.

before to request any image part, we can do it in parallel with the part requests (in the case of HTTP, a part would be called byte-range), reducing the initial delay. In this way, as soon as we obtain, for example, the offset and length of a packet, we can request it to the server. It is possible with the HTTP/1.1 protocol and its pipelining. The Deshpande and Zeng proposal does not allow this.

For concluding the evaluation of the existing techniques for remote browsing of JPEG2000 images, we present their main advantages and disadvantages in Table I.

IV. The proposed technique

The JPEG 2000 standard allows complex image files, with the possibility of including a wide variety of information, apart from the image data. But really, practically all of this other information is not necessary for applications for remote browsing at all. On the other hand, this additional information makes more complex and difficult to retrieve the image data. In almost all cases, a simple raw file (a J2C file with just a code-stream) can contain any kind of image destined for remote browsing applications. Limiting and fixing the image file structure we can simplify the image data retrieving process, making it more efficiency and faster.

The code-stream still permits a big number of image file configurations, so we would have to fix it a bit more. This limitation does not obstruct at all to use any image, but we will have to store the images in files with a specific structure.

In this paper we propose to use a limited J2C file format (hereinafter this kind of files will be called J2L files) and a client structure for retrieving the image data using the HTTP/1.1 as a technique to implement systems for remote browsing of JPEG 2000 images through Internet. A J2L is a standard J2C file that contains a JPEG 2000 code-stream with a fixed and known structure, that implies the following restrictions: i) the main header must contains TLM markers with the size of all the existing tile-parts, ii) all the tile-part headers must contain PLT markers with the size of all the associated packets, iii) the progression must be always the LRCP progression, that is, by quality, and iv) the tile-part partitioning must be by resolution.

A part from these restrictions, it is recommended (but it is not necessary) the following guides for a better transmission efficiency: i) the main header should be as little as possible, containing only the essential marker segments (SOC, SIZ, TLMs, QCD and COD), ii) the tile-part header should be as little as possible too (SOT, PLTs and SOD), iii) there should be precincts with a little size, and iv) there should be only one tile.

SOC, SIZ, COD, QCD, SOT, SOD and EOC are necessary marker segments for any code-stream. The TLM marker segment stores the size of every tile-part belonging to the code-stream. As all tile-parts are stored in a sequential way, reading the TLM data we will be able to access randomly to any tile-part. The PLT marker segment contains the length of all the packets of a tile-part, as they are stored. Every one of these lengths is coded to occupy the minimum bytes. In general, a length that is represented with L_B bits, is coded in a PLT with $\lceil L_B/7 \rceil$ bytes. As the packets are stored sequentially in every tile-part, and every tile-part contains all packets of a resolution level of a quality layer, we can construct an index reading the PLTs data. The PLT marker segment produces less overload than for indexed JP2

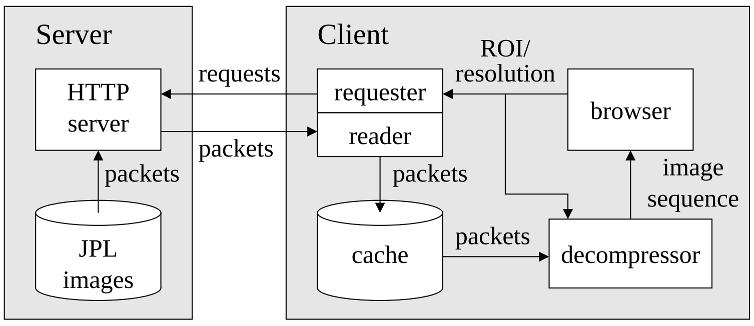

Fig. 3. Proposed system structure.

files, because in this last case, the stored information for every packet is bigger and fixed.

In the Figure 3 it can be observed the proposed system structure. In the server side it is necessary only a HTTP/1.1 server and a set of JPL image files. The client side is practically the same as in the JPIP solution, except for the module responsible for the communication with the server. In our proposal, this module is divided in two sub-modules which run concurrently: the requester sub-module and the reader sub-module. The requester sub-module receives the WOI request from the browser module and translates it into HTTP byte-range requests for the server. The reader sub-module is always listening to the socket for possible server responses.

For a WOI request, the requester sub-module calculates the necessary packets to reconstruct it, with simple geometrical operations. In JPL image files, the packets are grouped in one tile-part per resolution level and quality layer (R-L), so that the image index will be built as are required packets of a new R-L. When it is required a packet of a new R-L, the client has to read the header of the associated tile-part (using the TLM marker segments), where there are PLT marker segments. Processing the PLT information it is possible to build an index of the packets belonging to new R-L. In the PLT read processing, as soon as it is known the offset and length of a required packet, the appropriate request is made to the server.

V. Evaluation

For the evaluation we have used several true-color images (three components), with several sizes, from 1024x1024 to 5462x7087. The precinct sizes have been from 64x64 to 256x256. We have used a large number of quality layers and several resolutions levels. In order to evaluate the JPIP solution, it have been used the JPIP server and the JPIP browser of the Kakadu package, version 4.0, for Windows. The images have been coded in a JPL format. To evaluate the solution of indexed JP2-family files, we have been the Apache HTTP/1.1 server 1.3.4 for Windows and a self-implemented client for requesting the image data by byte-ranging. The images have been stored in indexed JP2 files. For the evaluation of the Deshpande and Zeng proposal, we have used simple index files and JPL images. The server used has been the same as for the previous evaluation. The client used has been a self-implemented one. Finally, in order to evaluate our proposal we have used the same HTTP/1.1 server, and JPL images.

In the Figure 4 it can be seen the graphic of the evaluation result, as average of all the tests done. The graphic shows the relation between the PSNR in dB of the reconstructed image and the received data in Kbytes. In all the tests it has been selected a different specific WOI, that is, a region within a resolution level. If we just limit the evaluation to the relation between PSNR and received data, it is true that the JPIP solution it the optimal. But, the proposal solution is very closed, above the others solutions. The difference it is caused in the first instance by the HTTP headers overload. Nevertheless, our proposal solves nearly all the mentioned disadvantages of the JPIP solution for Web applications.

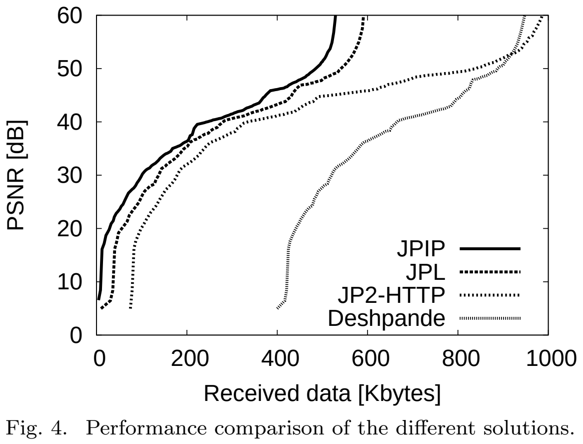

Fig. 4. Performance comparison of the different solutions.

VI. Conclusions

After evaluating the existing techniques for remote browsing of JPEG 2000 images on Web systems, it has been proposed a solution which tries to offer a simple and efficient implementation, and to avoid many of the presented disadvantages of the existing ones. Our proposal has the purpose of promoting and easing the implementation of powerful Web applications for remote browsing of images, exploiting the JPEG 2000 standard features, making use of the Internet infrastructure.

References

[1] ISO/IEC 15444, *Information Technology – JPEG2000 Image Coding System – Part 1: Core coding system*, 2000.

[2] D.S. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Transactions on image processing*, pp. 1158–1170, July 2000.

[3] ISO/IEC 15444, *Information Technology – JPEG2000 Image Coding System – Part 2: Extensions*, 2000.

[4] ISO/IEC 15444, *Information Technology – JPEG2000 Image Coding System – Part 9: Interactive tools, APIs and protocols*, 2003.

[5] D.S. Taubman and M.W. Marcellin, "JPEG2000: Standard for Interactive Imaging," *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1336–1356, August 2002.

[6] S. Deshpande and W. Zeng, "HTTP Streaming of JPEG2000 Images," in *Proceedings of the IEEE International Conference on Information Technology: Coding and Computing*, April 2001, pp. 15–19.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, http://www.ietf.org/rfc/rfc2616.txt, June 1999.

[8] "Netcraft Web server survey," http://news.netcraft.com/archives/web_server_survey.html.