Transmitting Video Images in XML Web Service

Francisco Prieto, Antonio J. Sierra, María Carrión García

Departamento de Ingeniería de Sistemas y Automática
Área de Ingeniería Telemática
Escuela Superior de Ingenieros
Avenida de los Descubrimientos, s/n
41092 Sevilla, España

Abstract. Transmitting video images across Internet implicates, a quality of image, by one side and, small time to refresh the reception without increase the bandwidth by the other side. We can manipulate encoding images to adequate to the specific request/response format for XML Web Services over HTTP. We consider transmission sequences images consecutively for display at screen one-by-one. The size of binary data transmitted across the network must be controlled. Measures are shown for a client/server model to evaluate the performance of three different encoding for an XML Web Service over HTTP.

1 Introduction

Last years a great evolution in the world of the telecommunications has been take place, mainly in the field of the personal communications. Mobile phones, laptops or PDAs have stopped being objects of luxury to take part in our daily life. Particularly in case of the mobile phones turns out to be a clear example of this evolution: connection to Internet, downloading of images, sounds or videogames are essential characteristics in any last generation device.

With the arrival of the third generation, videoconference through the mobile telephone has become a reality. Therefore it turns out interesting to study the different formats from existing image and video, as well as how to transmit them by the network in a fast and efficient way. There are several ways to send video images using web services. We are going to describe two of them. In the first case, the server sends a set of images, one for each request from the client. By this way the client seems to display a video, as it is showing the images consecutively. In the second case, the server sends a video signal. It can be done by sending a small file with the desired video, or by a real time protocol using streaming technology. The following image shows a series of possible configurations for capturing video images to transmit them by the network.

This article is organized as follows: First we present the most used images codification formats. Next we describe some video codification format. Finally we present the scenario where we realize tests for three codifications types, with the purpose of verifying which of them turns out to be more efficient for the transmission of images through the network.

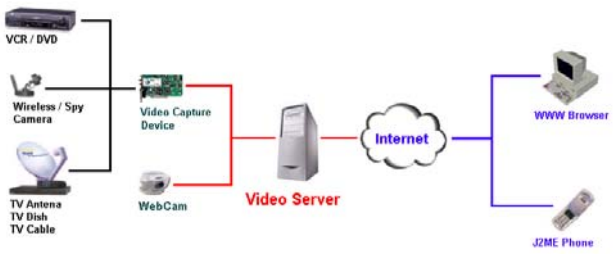

Figure 1. Client-server model for transmitting and receiving video signals

2 Transmission of a sequence of images

In this section we will see the different formats of image codification that can be used by the client and the server, evaluating the most indicated to make the transmission.

2.1 BMP format

An image in BMP format (Bitmap) is essentially a bit map, where the colour of each pixel in the image is defined one by one. Each pixel comes represented by 3 bytes, where each one of them indicates the proportion of colour in format RGB. This is the less used format for the interchange of images, since it has not got any type of compression. Therefore, it generates files with great size in comparison with the other formats.

2.2 GIF format

There are two variants, GIF87 and GIF89a. The second variant allows generates animated images, transparent backgrounds and interlaced format. GIF format allows transmit images efficiently through data networks. This type of files uses a kind of compression based on the algorithm LZW, which optimises the image storage without producing losses or distortion. Due to the characteristics of the algorithm, this compression works in an optimal way with images that have great areas and homogenous colours. It is less efficient in images with a great number of colours and different textures. In addition, the maximum number of colours an image GIF admits is 256.

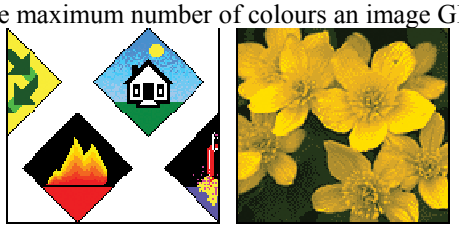

Figure 2: Comparison of images for a GIF compression. The picture on the left is better.

2.3 JPEG format

Unlike GIF images, JPEG admits a colour depth of 24 bits, having therefore a very superior palette. This is an ideal characteristic for storing photography, where a good quality is needed. In order to obtain the compression, it is used a mathematical technique called *transformation of the discrete cosine.* The compression of the image can be chosen, although simultaneously we will be determining the quality of image, because the process introduces loss of information.

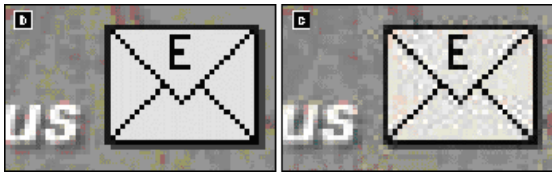

Figure 3: Quality loss in a compressed JPG image

2.4 PNG format

PNG format appears is a non-proprietary alternative to the GIF format, which is patented, like JPEG. In order to obtain compression it makes use of the LZ77 algorithm, also used in the creation of ZIP files. It is a lossless compression and it admits a colour palette up to 24 bits. In addition it supports up to 256 levels of transparency. On the other hand, the complete functionality of PNG format is not supported by all software.

Figure 4: Transparency effect in a PNG image

2.5 Evaluation of the image formats

Once we have analysed four of the most used formats for images, we evaluate them according to its quality-size relation. In order to transmit images with great quality and reduced data size, the most recommended format is JPEG. Between GIF and PNG formats we will show preference for the second, since it does not cause loss of information, admits a greater colour palette and has a better compression. Finally, we must indicate that format BMP is not recommendable for the transmission of images through Internet, since the size of the files is excessively great.

3 Transmission of a video sequence

There are two methods for the distribution of contents with audio and video on the Web. The first method uses a standard Web server to transmit data to a video receiver. The second one uses a streaming server.

3.1 Video transmission through a standard web server

This option requires that the video have a finite length. Once we have the codified file, we store it in the web server. In this way, a user who makes a request to the server will obtain the file. The protocol used in web servers is HTTP, which operates over TCP. This one assures the trustworthy transference of data, since it requests broadcasting of lost packets. On the other hand, it cannot assure that all packets will arrive at the client to be visualized on time. Because of that it is possible to experience some delay or losses of sequence.

3.2 Video transmission using a streaming server

This option allows the video transmission in real time using Internet. This technique is used to lighten the downloading and execution of audio and video in the Web, since it allows listening and visualizing the files while they are downloading. The signal is reproduced on the flight: the client makes the request and the server begins to send the file. The client receives the data and constructs a buffer where the information can be saved. Once the buffer has been filled with a small part of the data, the client simultaneously reproduces the video and continues with the downloading. If there is one moment where the speed reduces, the information stored in the buffer will be used. By this way, the video sequence will be played continuously.

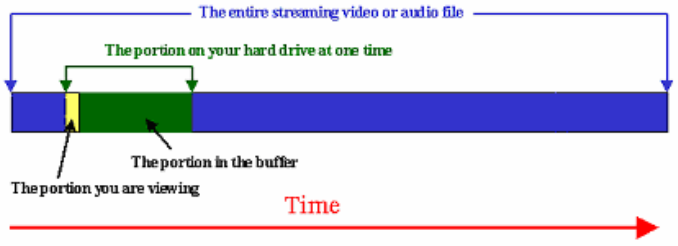

Figure 5: The principle of streaming

3.3 Formats for video codification

A video signal is a reproduction in sequential form of a set of images which, if they are visualized with a certain speed and continuity, gives the sensation of continuous movement. Many video formats are related to image formats. In a video signal there

are two types of redundancy. By one side there is a redundancy within the image, which is the typical one in a graphical file. This one can be reduced using compression techniques. On the other hand there is a redundancy between images, since the differences between two consecutive images are very small. We will be able to increase to the compression of the video storing only the differences between these images. Here are some of the most used formats for video codification: RGB, which is equivalent to BMP format in image. The data is captured with a 100% of quality and without compression. M-JPEG uses JPEG compression in each image, reducing therefore the redundancy of each image but without eliminating the one that exists between images. H263 uses more advanced codification techniques, being based on predictions of movement to reduce the redundancy between images. Finally there is MPEG, which uses an algorithm that not only compares an image with the previous ones but also with the later ones, increasing the compression.

3.4 Video transmission oriented to web services

The format of video codification is not the only important thing for transmitting information through the network. A robust data codification is also necessary to turn the bits of the video file in characters supported by the transfer data protocols. When we work with web services based on XML, the data needs to be processed correctly between equipment in different platforms. The best data codification with these characteristics is Base64, also used in MIME format for transmitting binary data over HTTP. Next we show the Base64 codification process. The binary data is grouped each 6 bits. If it were left some incomplete block it fills up with zeros. Next, each block is replaced for a character according to the relation established by the specifications. The map of characters used is formed by a set of 64 ASCII characters. At the end of the sequence a '=' character is added for each pair of zeros that we needed for filling.

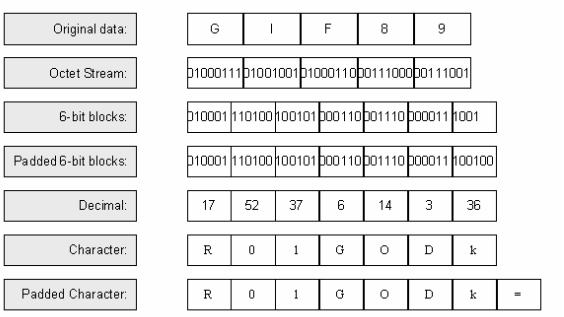

Figure 6: Fundament of Base64 codification.

4 Test scenario

We want to verify the effectiveness of the Base64 codification for video transmission. To do this, we will use a XML web server that sends an image in PNG format by each request that the client makes. We have used *Apache Tomcat* as web server, and the module *Axis* so that it has support for the SOAP protocol. This protocol, designed specifically for services Web, is based on XML. The client is a mobile device that we will simulate with the *J2ME Wireless Toolkit* program. Client and server establish a communication with SOAP protocol following the JSR-172 specification.

4.1 Tests

We realize tests for the following three codifications: Base64, commented previously; Base16, which consists of making groups of four bits and replacing them by ASCII characters. Finally we have used a codification based on an array of integers: each byte is codified like an integer, so a complete file can be stored in an array of integers.

For each one of these three codifications we made a series of tests. Each test consists of sending several images with the same size. The goal is to verify the time between transitions. Ten different file sizes have been used: from 1kb to 10kb, with increases of 1kb.

4.2 Obtained results

The obtained results show that the most effective codification is Base64, followed by Base16. The array of integers codification has turned out to be absolutely little efficient to transmit data using SOAP. The reason is the following: as we are sending an array, SOAP uses labels for each one of the elements of the matrix. Therefore, for each integer to send the beginning labels and the closing labels are also sent, which diminishes the yield remarkably. The other two formats, however, only send a string with the codified characters, because of that it is only needed an opening label and a closing label. The following graphics show the yield reached with the three codifications. The first image shows a comparison between the Base64 and Base16 codifications. In the X-axis we represented the different sizes of image used, measured in kilobytes, and in the Y-axis we showed the time that passes while we received an image of a certain size and the following image with the same size. As we can observe, Base64 is always more efficient than Base16. For files with 10 kb, we verified that the time between images with Base64 codification turns out to be half that the time measured between images codified with Base16.

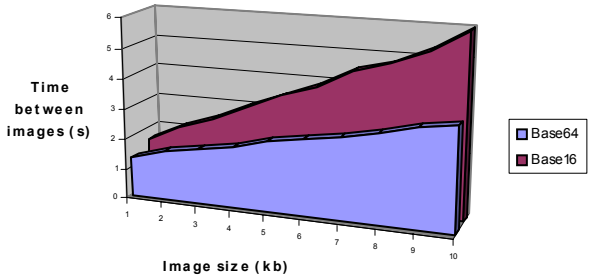

Figure 7: Comparison between Base64 and Base16 codifications

The following image shows a comparison between the three types of codification that have been tested. The codification with array of integers turns out to be absolutely little efficient, with a time between images thirty times greater than Base16 and sixty times greater than Base64 for files with 10 kb.

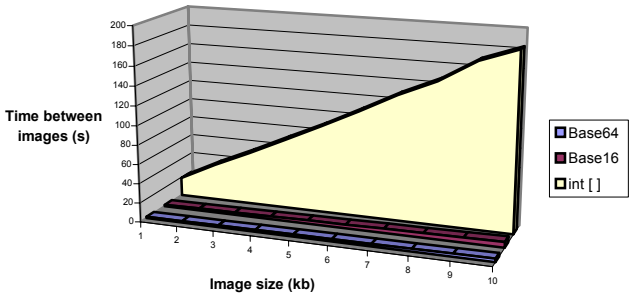

Figure 8: Comparison between Base64, Base16 and array of integers codifications

5 Conclusions

In this paper we describe the most used image and video formats at the present time, as well as some possible formats of codification to transmit binary data through the network. If we want to implement streaming for the shipment and reception of video images we need a data transfer protocol different than HTTP. The reason is that the flow to send has an indefinite size, since the length will come marked by the time that the user wishes to continue visualizing the images. There are protocols in real time like RTP to do that. Unfortunately, no mobile client implements nowadays this kind

of protocols. The only possible solution will be then the transmission of sequences of images, each one of them under the request-answer model. The ideal image format is JPEG because of its quality-size relation, but we find that MIDP 2.0 specification only supports PNG format. Finally, we would like to comment that the test showed that the most adapted data codification for the binary data transmission over the network turns out to be Base64, since it obtains a robust and portable codification increasing only to a 33% the size of the original binary file.

References

1. CauldWell, P.: Servicios web XML. Madrid: Anaya Multimedia (2002)
2. Rao, K. Ramamohan, Hwang, J.J.: Techniques and standards for image, video, and audio coding. Upper Saddle River (New Jersey): Prentice Hall (1996).
3. Seely, S., SOAP : Cross platform, Web service development, Using XML. Upper Saddle River, NJ : Prentice Hall (2002).
4. Sierra, A. J., Prieto, F., A comparative study of delay for transmission images for a mobile devices over HTTP. IADAT-micv2005 International Conference on Multimedia, Image Processing and Computer Vision, Madrid, pp.68-72, 2005. (ISBN: 84-933971-5-6), 2005.
5. Schulzinne, H.: "RTP: A Transport Protocol for Real-Time Applications", http://www.ietf.org/rfc/rfc1889.txt.
6. Java Specification Requests (JSR) 135, Mobile Media API, http://jcp.org/en/jsr/detail?id=135.
7. Fielding, R., et al., Hypertext Transfer Protocol -- HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt.
8. Java Specification Requests (JSR) 118, Mobile Information Device Profile 2.0, http://jcp.org/en/jsr/detail?id=118.
9. Portable Network Graphics (PNG) Specification (Second Edition). Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): functional specification. ISO/IEC 15948:2003 (E), http://www.w3.org/TR/PNG.
10. Java Specification Requests (JSR) 139, Connected, Limited Device Configuration, http://jcp.org/en/jsr/detail?id=139.