Multimedia Web Services Performance: Analysis and Quantification of Binary Data Compression

Amer Al-Canaan, Ahmed Khoumsi
Department of Electrical and Computer Engineering, Université de Sherbrooke, Québec, Canada
Email: {Amer.Al-Canaan, Ahmed.Khoumsi}@USherbrooke.ca

Abstract—Multimedia Web services constitute a considerable load on Internet bandwidth and on Web servers. With the increasing demand of multimedia Web services, the need to maintain a respectable QoS is a major concern for multimedia Web services’ providers. Binary data compression, lossy and lossless, increases performance gain and maintains QoS, at large and small-scale implementations. At the expense of increased CPU processing time, data compression provides a reliable solution to sustain QoS parameters such as response time, download time and server delay where bandwidth is limited.

This paper presents a detailed analysis, which is based upon realistic and representative data, in order to quantify the performance gain for multimedia Web services due to the application of binary image compression. To provide data for our analysis method, we have built an image-retrieval Web service that provides a set of images in both compressed and non-compressed form. The work in this paper promotes the utilisation of multimedia compression techniques and proposes the integration of multimedia compression techniques in current Web standards such as HTTP.

The analysis method and results can be extended and adapted to various service and network models, such as wireless LAN, Ad-Hoc, mobile networks, IP-telephony and distributed systems.

Index Terms—Web services, performance, QoS, image compression, response time, server delay, bandwidth.

I. INTRODUCTION

Web services are capable of conveying text and binary data. The exchange of multimedia including images, voice and video, is very common amongst Web users. This phenomenon adds more pressure on the resources of the Web, including available bandwidth. Thus, the best-effort Web can hardly cope with the increasing demand on multimedia services while sustaining acceptable quality of service (QoS).

To help accommodate new multimedia Web services, while improving the QoS, two natural solutions that can be considered include the following [1], [2]:

- Improve the medium [3], [4], by increasing the bandwidth. This may necessitate hardware upgrade and new infrastructure, which may or may not be feasible due to lack of investment and high cost. For example, when replacing a 56 – Kbit/s modem by a 512 – Kbit/s DSL (Digital Subscriber Line) or cable modem.
- Improve the media [3], [5], [6], by decreasing the message size through applying data compression, for example. This seems to be more effective, since it requires typically software upgrade and no change on the existing costly bandwidth and infrastructure.

Generally, for a multimedia Web service, payload is binary encoded and constitutes the major part of the sent data. Therefore we focus our attention on the payload itself, rather than on the header information, which can be text or XML-encoded1.

Binary data compression (whether lossy or lossless) improves QoS for both small and large-scale implementations where bandwidth is limited. The improvement of QoS is not only valuable to Web services that are based upon SOA2 and ROA3 [7], but also to other types of traffic on the Web including FTP (File Transfer Protocol) and IP-Telephony.

Although the utilisation of binary data compression reduces several QoS parameters such as response time, server delay and download time, the central processing unit (CPU) takes additional time during compression and decompression. The CPU contributes to the local factors that affect the parameters of the QoS. On the other hand, the response time is susceptible to different factors that make it uncertain. However, the extra processing time can be minimised by the use of optimised software code, more CPU and memory resources.

This paper presents a thorough and quantitative analysis of the performance obtained when applying binary compression to multimedia Web services. For the purpose of our analysis, we have developed an image-retrieval Web service, which provides various measurements and calculations of relevant QoS parameters. The open source DjVu [8] compression library is utilised in order to compress, decompress and display the image samples. The various measurements involve LAN (Local Area Network), WAN (Wide Area Network) and locally within one computer (localhost) configurations, with different hardware resources, bandwidth and platforms.

The analysis method and results can be extended, adapted and applied to improve several QoS [9] parameters in various types of networks, such as wireless

1XML, eXtensible Markup Language.
2Service Oriented Architecture, which is based upon the SOAP (Simple Object Access Protocol) protocol.
3Resource Oriented Architecture, which is based upon the RESTful (Representational State Transfer) style.
LAN (WLAN) [10], Ad Hoc\(^4\), mobile networks, IP-Telephony [11] and distributed systems.

The contribution of this paper include: 1) the methodology of the analysis including the detailed discussion of its results, which quantify accurately the performance improvement for the implemented multimedia Web service based upon realistic and representative image samples, 2) the proposition to standardise image compression tools and formats into HTTP upon which Web services are built.

This paper is divided into six distinct sections. Section II highlights some related works. Section III gives an overview of data and image compression. Section IV presents the development of an image-retrieval multimedia Web service and Web client application utilising an efficient library of image compression, i.e. DjVu. This Web service provides the input data for realising our analysis. Section V presents the performance analysis method, various service configurations and provides a discussion of the results. Finally, section VI consists of the conclusion and future work.

II. RELATED WORK

A comparison between two lossless compression techniques, namely, RLE and Huffman encoding [12], is presented in [13]. An implementation of a multimedia Web service is provided in order to demonstrate the compression ratio and the compression speed for both encoding techniques.

The performances of Web services, ASP (Active Server Pages) and RMI (Remote Method Invocation) [14]–[16] are compared showing that the data overhead of Web services’ messages is non-negligible, which lowers the throughput and increases the response time of Web servers. The throughput of Web servers is measured with and without payload compression. The performance of Web services can outperform that of RMI when document-oriented style of communication is utilised, rather than RPC (Remote Procedure Call). Generally, RMI is faster than SOAP Web services and thus some work has been done to determine when it is more adequate to utilise RMI rather than SOAP, according to predetermined QoS parameters such as availability and accessibility. Other factors that speed up SOAP applications include certain of XML-processing tools [17].

QoS contract composition is part of Web services orchestration and choreography. The work in [18] proposes a probabilistic approach to QoS soft contract composition that is based upon probability distributions rather than fixed figures. The objective is to produce more natural and less pessimistic QoS contracts that can be generated using Monte-Carlo simulations.

Reducing the message size through compression [3], [9], [19], [20] enhances the security and the QoS for mobile devices running XML-based Web services. The security is enhanced through XML data encryption. The utilisation of XML compression and then encryption reduces the transmission time as well as the CPU processing time. Compression reduces message size and thus leaves less data to be processed by the encryption algorithm. Obviously, this approach is beneficial when the payload itself is composed mainly of XML data. Binary XML data compression technique results in a trade-off between response time and throughput. Data compression reduces messages sizes and thus increases throughput. On the other hand, the response time increases due to compression and decompression overhead [3]. The security level, which includes authentication, encryption and encapsulation of data, has an impact on the performance of wireless networks. Different levels of security, up to ten levels, were tested on different configurations with several performance parameters such as mean response time and throughput.

Other proposed approaches to improving the performance of Web services at the transport layer include HTTP caching and UDP (User Datagram Protocol) message binding [21], [22]. UDP offers fewer headers, lower message overhead, no error correction and connectionless behaviour. Binding protocols such as HTTP (Hypertext Transfer Protocol), FTP and SMTP (Simple Mail Transfer Protocol) are compared with UDP in terms of performance. On the other hand, caching web responses can be beneficial for read-only Web services or when the responses do not vary in a relatively short time.

Our work can be distinguished from those mentioned above in several aspects. Firstly, we concentrate our research on lossy compression in multimedia Web services, where the payload data are in binary format (images, voice and video) rather than XML. Secondly, we study the influence of binary image compression on Web services at both of the client and server sides in order to analyse several relevant QoS parameters such as response time, download time and processing time. Thirdly, we cover up a wider range of data sizes, which is more realistic and common to multimedia Web services.

Compared to our work in [23], we provide more details about our on-line analysis method including the pseudo code. We provide also more analysis and discussion about the three scenarios (LOCAL, LAN and WAN) including the experimental results of the response time.

III. DATA AND IMAGE COMPRESSION

Payload sizes of multimedia Web services can be considerably large. When dealing with high-resolution images and high quality video, data sizes can be above 1 MB, which increase packet congestion probability, lower QoS and augment bandwidth overload. Two solutions come to rescue the situation, namely, increasing the bandwidth and utilising data compression.

The first solution may involve new hardware upgrade and new infrastructure, which may not be feasible without adding more investment cost. We believe that this solution is effective merely in the short run, since it would eventually lead to saturation and degradation of QoS. On the

\(^4\)Ad Hoc, refers to networks that do not require a router or a wireless access point to establish a connection between two or more computers.
other hand, our tests amongst many others reveal clearly a gap between the actual and the nominal bandwidth figures [1]. For example, for an Internet cable modem with nominal download rate of 512 Kbit/s, the actual download rate was about 75 Kbit/s.

The second solution is utilising data compression. Different data compression techniques can be utilised efficiently to help Web services keep up with the growing demand on multimedia Web contents and respect their QoS constraints. Several approaches propose compressing SOAP headers. This is less effective for multimedia Web services, since the length of SOAP headers is much smaller than the payload size.

Lossless data compression is inherent in HTTP 1.1 standard and is available for Web browsers and Web clients. The supported formats include those generated by gzip (LZ77), compress (LZW) and deflate (zlib) [12], [24]. This has considerable influence on Web services, which are based upon XML and text content. On the other hand, multimedia Web services require adequate compression tools (lossy and lossless) to be adopted and standardised. The DjVu library is a promising tool, due to its high image compression ratio, which can be utilised in conjunction with the existing compression standards in order to achieve full compression to textual content and binary payload.

The application of a high compression ratio utility is desired so as to maintain the available Web bandwidth and QoS for multimedia Web services as well as for other types of services such as IP-telephony. The choice of a data compression technique is based upon the following criteria [25]:

- Compression ratio.
- Speed of compression and decompression and,
- Compatibility and ease of conversion between different formats.

Usually, binary compression libraries utilise one or more algorithms of compression to achieve higher compression ratios. The open source DjVu library [8], [26] relies on the classification of every pixel of an image into one of the two categories: foreground or background. Foreground pixels represent text or drawings, while background pixels refer to pictures, paper textures and colours. The foreground is compressed using pattern-matching technique, while the wavelet technique [27] is utilised for the background. For example, JPEG images can be compressed and displayed efficiently as DjVu format, denoted djvu. However, other equivalent image compression tools can be utilised.

The djvu-image format cannot be converted directly to JPEG (Joint Photographic Experts Group) format. An intermediate step has to take place, i.e. converting from djvu to TIFF (Tagged Image File Format). Other tools and libraries can be utilised to convert between various image formats. For example, ImageMagick [28] is another utility, which is an open source project that is available for a variety of platforms including UNIX, OS X and Windows.

For demonstration purposes, the main characteristics of two tools of DejaVu, namely c44 and bzz, are compared to the characteristics of PPM [25], with three orders, namely PPM(4), PPM(6) and PPM(16) as represented on figure 1. The compression ratio (bits/Byte) for a given image is calculated as its compressed size in bits divided by its original size in bytes (the lower the better). The speed of compression and decompression are expressed as K B/s (the higher the better). The c44 image compression tool is based upon the wavelet technology (suited for only image compression), while the bzz and the PPM encoding utilities are general purpose since they can be used to compress text and binary image data.

According to our test results as shown on figure 1, PPM(6) offers the highest compression and decompression speeds, while c44 has the best compression ratio (3.2 bits/Byte). As the order of PPM is increased, the compression and decompression speeds become lower due to the increasing number of iterations of the PPM algorithm. Figure 1 shows that the decompression of the bzz tool is more sensitive to machine processing capabilities compared to compression (the test was held with two different machines: OS X having more resources, in terms of CPU and memory, and Linux with fewer resources).

![Figure 1. Comparison of some image compression tools.](image_url)

IV. IMAGE-RETRIEVAL MULTIMEDIA WEB SERVICE AND WEB CLIENT DEVELOPMENT

To help quantify the influence of data compression on multimedia Web services, we developed an image-retrieval Web service and a Web client application using NetBeans 6.5 IDE (Integrated Development Environment) and GlassFish server v2.1. A good tutorial for Web service implementation with NetBeans can be found in [29]. This section provides details about the image-retrieval Web service and the Web client we developed. The Web client issues requests, which are processed at the Web service, and collects relevant data for analysis.

A. Development environment

For comparison, each image is available in its original JPEG format as well as in djvu-compressed format (com-
pressed in advance using the c44 tool of the DjVu library because it offers a respectable compression ratio).

Our developed Web service involves several standard API's and tools [30], namely:

- **JAX-WS** (Java API for XML-Based Web Services): it is part of Java EE and it enables the creation of RESTful as well as SOAP Web services and is intended to replace JAX-RPC. The current version utilised in our implementation is 2.1.
- **JAXB** (Java Architecture for XML Binding): it is used to bind Java objects to XML schemas and vice versa. This makes it possible to process XML as Java objects and to represent Java objects as XML.
- **WSIT** (Web Services Interoperability Technology): an extension to JAX-WS intended to enhance security and interoperability with Microsoft’s Windows Communication Foundation.
- **MTOM** (Message Transmission Optimisation Mechanism): a mechanism to send binary data, as raw bytes, separately from the SOAP data. One way to include binary data inside SOAP is to utilise base-64 encoding (using 64 printable characters, thus each character is encoded using six bits). To encode three bytes (twenty-four bits) of binary data utilising base-64, it takes four bytes, since the 24 bits are taken six bits at a time to produce four characters. Obviously, this is inefficient since the size of binary data gets larger, due to the base-64 encoding, which is necessary in order to fit within a SOAP message.

In addition to the Web service itself, we developed a comprehensive Web client agent that measures various QoS parameters. The message flow diagram between the Web client and Web service is shown in figure 2. The Web server responds to client requests, which includes getting an image by name, querying its size, returning a list of all available image names, and calculating the time duration needed to process the request and prepare the required image bytes to be sent back to the client.

The Web client application has two roles:
1) Retrieving and displaying a set of 20 images on a specially designed GUI (Graphical User Interface, not shown here).
2) Measuring and saving several parameters for further analysis, such as the following (to be explained more in section V-A):
   - **ping time**, which is the time needed for a tiny packet to travel from one host $a$ to another host $b$ and back to its origin, i.e. host $a$.
   - **processing delay** time at the server ($S_{Delay}$), which is the time needed for a server to process a request. This time is between receiving a request at the server and sending a response.
   - **response time**, which is the time duration between sending a request at the client and receiving a response.
   - the actual **download time**, which is the time between receiving the first and last bits of data.
   - **image-rendering time**, which is the time between receiving the last byte of an image and displaying the image on the GUI.

**B. Network configuration**

Our Web service and Web client were developed for several platforms (see table I) that are located at two different domains:

- Domain 1, with nominal download speed of 512 $KB/s$ for WAN and 10 $MB/s$ for the LAN, which consists of Mac OS X and Linux operating systems.
- Domain 2, with nominal download speed of 10 $MB/s$ for WAN and 10 $MB/s$ for the LAN, which consists of Windows XP, Linux and Sun Solaris operating systems.

The implementations are built with the most recent Java versions for each platform, i.e. jdk1.5 on OS X (G5) and jdk1.6 elsewhere.

![Image Retrieval Web Service Diagram](image)

**Figure 2. Overview of the image-retrieval Web service.**

At the Web server side, three entities can be distinguished, namely, a Web application, a Web service and a Java bean. At the outer level, the Web application wraps both the Web server and the Java bean, while the Web service wraps the Java bean. The requests are delegated down to the Java bean, which acts roughly as a database. Since the internal structure of the Web server side components is invisible to the client, we refer to them by the abstract term **Web service.**

**TABLE I. SYSTEM AND PLATFORM CONFIGURATIONS.**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Platform</th>
<th>RAM, CPU@clock frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OS X 10.4, Linux 2.6</td>
<td>1.25 GB, G5@1.6 GHz, 386 MB, PIII@0.7 GHz</td>
</tr>
<tr>
<td>2</td>
<td>Windows XP, Linux 2.6, Solaris</td>
<td>512 MB, PIV@2.5 GHz, 2 GB, Intel D@2.8 GHz, 512 MB, UltraSPARC@0.4 GHz</td>
</tr>
</tbody>
</table>

Although we demonstrate our analysis method on the wired-network, we believe that the same approach can be extended to cover the wireless networks [10] with the following generalised model:

- Bit error rate. Although, the wireless network has much higher bit error rate due to the radio link, the wired network has a very low bit error rate.
• Link loss probability. This factor is higher for wireless networks. On the other hand, wired networks can suffer from link loss due to different causes such as weather conditions (corrosion of cables), hardware failure of routers and power failure.
• Mobility and roaming. Mobility for wired networks is limited compared to the wireless network.

V. PERFORMANCE ANALYSIS

In this section, we analyse the performance of the multimedia Web service we have developed. The Web service provides ten images in JPEG format and in djvu format by utilising the c44 tool [8], which provides a respectable compression ratio as shown on figure 1 (a total of twenty images). The selected images sample span actual sizes from $1 \times 1$ pixel up to $3822 \times 4819$ pixels, corresponding to 655 bytes up to 4617.087 KB. The ten JPEG images were selected arbitrarily from various sources, and thus they are compressed differently.

The analysis is realised for the following scenarios:

1) LOCAL, where the client and server code share the same resources of a single machine (local host). This configuration enables rapid verification and preliminary testing before deploying the actual implementation on the Web, since the delay times and response times are lower. This configuration provides guidelines in order to validate the calculated results.

2) LAN, where the client and server are on two different machines within the same local area network (LAN). Two different LANs were available to our tests: fast LAN in domain 2 with 10 Mbit/s download speed and slower LAN in domain 1 with nominal download speed of 0.512 Mbit/s. This configuration is valuable for comparison purposes as well as for verifying the analytical model. This configuration has also several advantages over the LOCAL version, since it separates the client from the server, thus it gives a better overview of the actual behaviour of the Web service once deployed on the Web. It is particularly advantageous when using devices and nodes with limited memory, CPU resources and bandwidth.

3) WAN (Web), where the client and server are located on the Web or on different domains. This configuration offers 70 Kbit/s for download and 20 Kbit/s for upload, which represent many legacy systems and data networks including wireless, mobile, sensor and residential networks.

The analysis is divided into two phases: online and offline. During the online phase, the Web service and the Web client application interact with each other in order to retrieve images, measure and save performance data in real time. During the offline phase, the data obtained during the online phase are utilised to calculate different performance parameters that can be plotted and submitted for further analysis.

A. Online phase of analysis

Once the Web service is deployed and running, the Web client application offers two modes of operation: manual and automatic. In the manual mode, the client has to specify the names of the images to be retrieved one after another. On the other hand, the automatic mode retrieves all of the twenty images and processes them successively in a cycle of twenty iterations through the GUI. The number of desired cycles $n$ is to be set before starting operation in the automatic mode. The Web service can provide a number $m$ of images, up to twenty. To ensure the fidelity of data, we chose $n$ to be five so that every performance parameter for each image is measured five times at five dispersed time intervals. The following pseudo-code V.1 resumes the operations done by the Web client application:

```
Algorithm V.1: WEB CLIENT FUNCTIONS(n, m)

begin

where $\{n, m\} \subset \mathbb{R}_{\geq 0}$

$n = 5$ , $m$ is the number of images

for $j := 1$ to $n$ step 1 do

for $k := 1$ to $m$ step 1 do

Query the $k^{th}$ image

if image is djvu – compressed then

Decompress and save image

end if

Display image on the GUI

Calculate performance parameters

Display results

Wait a few seconds

end for

end for

end
```

As can be depicted from the above algorithm, the time needed to complete all iterations would be considerable for the WAN configuration, in particular, due to the low bandwidth. The LOCAL configuration becomes handy during deployment and validation of the calculated QoS parameters, since it provides results quickly compared to the LAN and WAN configurations.

The Web client measures, records and calculates several relevant parameters as shown on figure 3, which constitute the input to the offline phase of analysis. It is worth noting that several other parameters are derived from those mentioned below, thus they are not mentioned for the sake of brevity:

• The ping time (round-trip time), which is the time needed for a tiny packet (sent by the client) to reach the server and to return back to the client. This time is denoted as $t_3 - t_1$ on figure 3.
• The processing delay time (S_Delay) at the server ($t_5 - t_4$), which is the time duration between receiving a request and sending a response.
• The response time ($t_6 - t_3$), which is the time
duration between sending a request at the client and receiving a response.

- The download time ($t_7 - t_6$), which is the time between receiving (at the client side) the first and last bits of a response message.

- The image rendering time ($t_8 - t_7$), which is the time between receiving the last byte of an image and displaying the whole image on the monitor (GUI).

- The image compression ratio, which is expressed using the relative per cent error calculation. It describes the relation between an image’s non-compressed (original) size $os$ and its compressed size $cs$ as follows: 
  $$\%\text{image compression ratio} = 100\% \frac{os - cs}{os}. $$
  A positive image compression ratio, thus a desired value, means that the compressed size is less than the non-compressed size.

- The image-retrieval speed, which is the image size divided by the download time.

- The actual download speed, which is calculated by dividing the received data size (or response message size) by the download time.

### B. Offline phase of analysis

As shown on figure 5, the percentage compression ratio (expressed as relative per cent difference) and the image size are not correlated. It is also noticed from the same figure that the compression ratio is higher than 40% for all of the images except for the one with the largest size. Each image has different content, thus the compression ratio varies accordingly [27]. Images with high variations in colour and details have the least compression ratios, whereas images with large blocks of uniform colour have more redundant pixels and thus high compression ratio is attained. The highest compression ratio is 87.2% for the image with 655 bytes, since it has only one pixel and one colour. The second highest compression ratio is 81.0% as shown on figure 4(a).

Dealing with the effective compression ratio instead of individual ratios simplifies the analysis without loss of precision. Thus we deal with the image samples as a single large image.

The overall average image compression ratio can be obtained by dividing the area of the histogram on figure 5 by the range of image sizes. Let $n$ and $i$ be the number of images (i.e. $n = 10$) and the index of the current image, respectively, where $\{n, i\} \subset \mathbb{R}_{\geq 0}$. For each image, where $i \in [1, 10]$, we denote its size by $\Delta x_i$ and the corresponding compression ratio by $y_i$. The total area of all columns $Area_{total} = \sum_{i=1}^{n} (\Delta x_i \cdot y_i)$ and the total width of all columns $Width_{total} = \sum_{i=1}^{n} \Delta x_i$. The global compression ratio (i.e. for all samples) is given by:

$$\%\text{Compression} = \frac{Area_{total}}{Width_{total}} \times 100\% \quad (1)$$

By substituting the expressions of $Area_{total}$ and $Width_{total}$ in equation 1 we obtain the global average compression ratio for our sample images as follows:

$$\%\text{Compression} = \frac{100\%}{n} \sum_{i=1}^{n} \Delta x_i \sum_{i=1}^{n} (\Delta x_i \cdot y_i) \quad (2)$$

As shown on figure 5, the number of images is 10 and thus the global average image compression ratio is 30.21%.

The compression ratio influences the following parameters: response time, download time and rendering time. Our practical results show that the ping_time does not change much for a given scenario and thus it can be considered constant. Moreover, with the assumption that multimedia Web services deal with large data sizes, the $S_{Delay}$ (as shown on figures 6 and 7) is the dominant factor in the response time calculations. It is worth noting that the LOCAL configurations are for theoretical comparisons or proof of concept that indicates the coherence of the obtained results for the LAN and WAN configurations.

The overall download time of the whole image samples, i.e. 20 images, for the WAN configuration (as shown on...
figure 6) is the highest (392.46 s) since it has the least bandwidth of all scenarios. However, the S_Delay for the LOCAL configuration has the smallest value (0.20 s), which is natural since the packets stay at the local host machine. The S_Delay for the LAN configuration is higher than that of the WAN configuration. This is justified by the higher CPU and memory resources available for the WAN configuration, for example both server and client run with more than 1.25 GB of memory.

The percentage decrease in S_Delay and in response time on figure 8 due to compression for the three scenarios (LOCAL, LAN and WAN), as shown on figure 8. The decreases in S_Delay (calculated using the per cent error calculation) for the LAN and LOCAL scenarios are close to each other (44.71% and 49.57%), while the decrease in S_Delay for the WAN scenario is 25.80%. The decrease in S_Delay for the LOCAL configuration is greater than that for the LAN scenario, since the resources are shared between the Web service and the Web client at the local host machine and thus the performance is affected.

As expected, the S_Delay decreases due to compression for the three scenarios (LOCAL, LAN and WAN), as shown on figure 8. The decreases in S_Delay (calculated using the per cent error calculation) for the LAN and LOCAL scenarios are close to each other (44.71% and 49.57%), while the decrease in S_Delay for the WAN scenario is 25.80%. The decrease in S_Delay for the LOCAL configuration is greater than that for the LAN scenario, since the resources are shared between the Web service and the Web client at the local host machine and thus the performance is affected.

For WAN configuration, the S_Delay time is negligible, which is only 0.36% of the download time as can be noted from figure 7. On the other hand, the S_Delay is non negligible for both LAN and LOCAL configurations, 30.18% and 30.10%, respectively. This can be explained by the fact that the bandwidth available locally (for LAN and LOCAL configurations) is larger than that of WAN, which means that data transfer is higher and thus download time is lower. It is noticed that the S_Delay for the WAN configuration is less than that for the LOCAL configuration since it depends on the hardware and CPU resources available within each configuration as shown on table 1.

The analysis results shown on figure 9 indicate that the download time is decreased by 30.52% for the WAN (Web) configuration. This decrease in download time is very close (within 1.03%) to the image compression ratio calculated previously (30.21%) in equation 2. Image compression reduces image size, thus fewer bytes are sent and less time is needed to download them at the client.
However, the download time decrease for LAN and LOCAL was not close enough to the theoretical value (30.21%), since some collected values were very small (sub-millisecond) and could be rounded off easily. The download time is much reduced in the LOCAL scenario where the Web client application and Web services codes are situated side by side and benefit from high-speed internal communication channels.

Figures 10, 11 and 12 show the various response times analysed for several configurations (client → server) for LOCAL, LAN and WAN scenarios. For example, Win → Mac (WAN) configuration means that the client is running on Windows XP and the server is running on Mac OS X and both of them are situated in two different networks and two domains on the Web (as explained in section IV-A). In figures 10, 11 and 12, the response time is not linearly correlated to the image sizes, and thus the behaviour of multimedia Web services that deal with large payloads, for instance images, cannot be simply predicted by utilising analysis models that are based upon small-sized packets. Figures 10, 11 and 12 show that each curve tends to have three distinct regions: below 60 KB, within 60 – 300 KB and above 300 KB, where the slope changes accordingly.

The non-linear behaviour of the response time is attributed to several factors, which can be summarised as follows:

- The variation in network propagation delay and traffic congestion.
- The variation in CPU load. This factor is kept minimum as we utilised the Web server and Web client application in single user mode with minimum processes running. Running the complete set of iterations in a sequential manner and at the same period of the day can help keep the variations in network configurations to a minimum.
- The need for virtual memory for large payload sizes. The processing of large images takes considerable amount of RAM (Random Access Memory). The client application needs additional memory for the processing of received images and for the corresponding calculations. This in turn increases the response time.

Figure 10 shows that the response time is higher for the Mac → Mac (LOCAL) configuration, compared to the Win → Win (LOCAL) configuration, since the Windows machines have higher CPU and memory resources available, as shown in table I.
Figure 11 shows that the Linux → Win (LAN) curve is below that of Win → Win (LAN) and Win → SUN (LAN), i.e. response time is lower for the Linux → Win (LAN) since the Linux machines have more effective processing resources and memory available (see table I). It is noticed that best linearity or quasi-linearity is noticed for Linux and OS X (Linux → Win and Mac → Win), while for Windows it was non-consistent, which is related to the particularities of the Windows operating system.

Figure 12 shows that the response time values for the Win → Mac (WAN) configuration are about 20 ms and are almost constant in the range of image sizes below 60 KB, which is explained by the fact that the ping time becomes dominant in the calculation of the response time. The other factor in the calculation of the response time is the S_Delay, which is negligible for the image sizes below 60 KB.

We utilised different tools in order to analyse the internal composition of packets, which include the built-in HTTP monitor of GlassFish (accessible from within NetBeans), the service tester Web interface in NetBeans and Wireshark.

The last three rows on table II indicate that the header information for response messages is well above 10%, which correspond to image sizes of 12,096 KB (240 × 180 pixels, compressed image), 655 bytes (1 × 1 pixel, non-compressed image) and 84 bytes (1 × 1 pixel, compressed image). This can be explained as follows: SOAP wraps the actual data into an XML envelope (SOAP overhead), augments the header information and thus the total size of sent information is increased. When the message size is relatively small, the ratio (header size)/(total message size) becomes relatively high. We consider that the SOAP overhead is small when the ratio is below 10%. When the envelope information becomes considerable, i.e. above 10%, compared to the response message size, the efficiency of SOAP becomes low.

The data in table II and the Web service tester interface is utilised to analyse the request messages in details. The request size is almost constant, approximately 700 bytes, for all images, since the only difference between requests is the image name. The Web service takes two arguments: the image name, for example, tinyImage.jpg and the name of the method (getImage) to process the request. The total request size is 698 bytes of which the SOAP part is 273 bytes (encoded as XML with UTF-8) and the length of both arguments is 22 characters (bytes). The effective information in the SOAP message is only 22 out of 273 bytes, which constitutes 22/273 = 8.06% of its size. Alternatively, to send 22 bytes, the SOAP overhead was 273 – 22 = 251 bytes, which constitutes 251/698 = 36% of the request size. On the other hand, if we consider that the payload is the SOAP part (273 bytes) then the efficiency of HTTP (transport protocol) and the underlying TCP/IP protocols becomes 273/698 = 39%.

6Wireshark is a general packet sniffing tool that is available from http://www.wireshark.org.
Compressing payload in multimedia Web services greatly improves their QoS, enhances security and saves bandwidth. Local resources such as CPU and memory can be easily afforded and tuned in order to improve the QoS, while Internet resources such as bandwidth and remote processing are shared and vulnerable to congestion and delay, leading to QoS degradation. This emphasises the role of data and multimedia compression for Web services and other Internet applications.

We developed an analysis for the performance of multimedia Web services utilising binary data compression for three scenarios: LOCAL, LAN and WAN. The LOCAL configurations provides guidelines for the measured and calculated parameters as well as a rapid test and implementation on a single machine. In order to demonstrate our analysis method we implemented an image-retrieval Web service and a Web client application that measures and calculates various QoS performance parameters in real time. Our analysis is based upon realistic data model, where the payload is mainly multimedia data and the SOAP header information is negligible except for the smallest images in the samoke. The analysis spans a wide range of image sizes from $1 \times 1$ pixel up to $3822 \times 4819$ pixels.

We have also demonstrated some further analysis including payload and data overhead, which are influenced by the choice of SOAP as a messaging protocol.

An important contribution of our approach is that it promotes the utilisation of data and multimedia compression techniques in Web services implementations. The analytical approach to Web service QoS quantification is based upon realistic data and representative data samples. We propose also including multimedia compression techniques such as DjVu in current Web standards such as HTTP. The proposed approach can be extended and adapted to accommodate the specific criteria imposed by LAN/WAN, wireless LAN, Ad-Hoc, mobile networks, IP-Telephony and distributed systems.

A. Future work

We plan to extend our approach to include more QoS parameters, more multimedia types such as video and voice. Also, we would like to extend our analysis to include RESTful Web services.

REFERENCES


Amer Al-Canaan was born in Jordan. He received the B.Eng. degree in electronics engineering from Yarmouk University, Irbid, Jordan, in 1997. He has worked at the Jordan Telecom in 1999-2002 as a telecommunications engineer. He received the M.S degree in electrical engineering from the University of Sherbrooke, Canada, in 2006. He is currently a Ph.D. candidate as well as a lecturer at the University of Sherbrooke in Canada, where he is researching Web services, VoIP and QoS.

Ahmed Khoumsi received the Engineer degree in Aeronautics and Automation from the engineer school SUP AERO (Toulouse, France), in 1984. From 1984 to 1988, he achieved his research activities in the LAAS, a CNRS research center, in Toulouse. In 1988, he received the Ph.D. degree in Robotics and Automation from the University Paul Sabatier in Toulouse. From 1989 to 1992, he was an Assistant Professor in Robotics and Computer Engineering at the engineer school ENSEM (Casablanca, Morocco). From 1993 to 1996, he was a Postdoctoral Fellow in the Communication Protocols group of the University of Montreal. From 1996 to June 2000 he was Assistant Professor, from June 2000 to June 2006 he was Associate Professor, and since June 2006 he is Full Professor, all in the Department of Electrical and Computer Engineering, at the University of Sherbrooke, in Canada. His present research activities include: modeling, testing, supervisory control, diagnosis and prognosis of distributed and real-time discrete event systems; design and creation of telecommunications services; wireless sensor networks; and Web services. He has been a one-year Visiting Professor at IRISA (Rennes, France) in 2002-2003 and at King Saud University (Riyadh, Kingdom of Saudi Arabia) in 2010-2011.