Enhancing Performance of Web Services in Mobile Applications by SOAP Compression

Ivan Ivanovski, Saso Gramatikov, Dimitar Trajanov

Abstract — **since being proposed in May 2000, the SOAP Internet protocol has by now revolutionized application development by connecting graphic user interface desktop applications to powerful Internet servers using the standards of the internet: HTTP and XML. However the fact that it uses a text-based XML encoding unlike the traditional binary protocols, results in higher bandwidth consumption, bigger storage requirement and increased processing overhead. When using mobile devices, one additional question about the battery performance raises. This paper presents the results of implementing efficient compression and its positive impact to the overall performance of the mobile devices that use wireless communication with web services, in terms of significantly reduced bandwidth, quite improved response time, and extended battery life of the mobile devices.**

Keywords – **web service, optimization, SOAP compression, battery life**

I. INTRODUCTION

THE DEVELOPMENT of wireless networks brought a new impulse to the portable devices as well. The new possibilities in the field of portability envisioned a new way in design and implementation of wireless communication protocols between mobile devices and one central system. Ever since, the transportation of data acquired from and sent to these devices has been challenging in aspects of security, stability, performance, response time, battery life, etc. In this paper an accent is put over the optimization of usage of web services through wireless medium by compressing the soap messages that travel through the network.

In general the web services allow the applications to use them as black boxes, passing some parameters to them and expecting results without any need for knowing what the web service actually does inside its code. The application “knows” which parameters to send and what results to expect, and this “knowledge” comes from the web reference within the application itself – a proxy created according to the web service description language file which is part of the web service itself. The self describing attribute of the web service is what makes it so powerful and gives the opportunity to be used as an outsourcing service for application. Since the processing happens on some server where the web service is hosted, using web services keeps application itself small in bytes, and also keeps the local processing light. In its root, a web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML, so that its definition can be discovered by other software systems. These systems can then interact with the web service in ways prescribed by its definition, using XML based messages. These messages are transported using standard internet protocols. HTTP is the standard network protocol for internet available web services, although platform or vendor specific protocols are also plausible. One level higher than the transport protocols lays Simple Object Access Protocol or SOAP – a simple, lightweight XML mechanism for creating structured data packages which are being exchanged between network applications. Four basic elements are fundamental components of SOAP: (1) an envelope that defines a framework for describing message structure; (2) a set of encoding rules for expressing instances of application-defined data types; (3) a convention for representing remote procedure calls (RPCs) and responses; and (4) a set of rules for using SOAP with HTTP. SOAP is quite simple and thin top-up over the existing network protocols that are widely implemented. It is easily extensible, understandable and it is based on XML. However, since the messages that pass here are self-describable, text-based, there is one setback – redundancy in data. This paper offers a solution how that redundancy can be eliminated with a cost-effective method that has a great impact on more aspects of the overall performance of mobile applications that use web services over wireless.

In Section II a related work is discussed. Several different proposals for optimizing web services are compared and analyzed. In Section III the optimization method proposed in this paper is discussed. Furthermore, in Section IV the actual implementation of the optimization technique is explained and clarified. In Section V the performance analysis and the improvement of the quality of service is presented. Conclusion is given in Section VI.

II. RELATED WORK

Different aspects of optimizing the performances of portable devices have already been approached. An approach has been made straight to the root of the problem, the text-based representation. A binary

Ivan Ivanovski, B.Sc., Faculty of Electrical Engineering and Information Technologies in Skopje, Macedonia (tel: + 389 71 386501; e-mail: i.ivanovski@gmail.com)

Saso Gramatikov, B.Sc., Faculty of Electrical Engineering and Information Technologies in Skopje, Macedonia (tel: + 389 2 3099 153; e-mail: saso.gramatikov@feit.ukim.edu.mk)

Dimitar Trajanov, Ph.D., Faculty of Electrical Engineering and Information Technologies in Skopje, Macedonia (tel: + 389 2 3099 153; e-mail: mite@feit.ukim.edu.mk)

representation has been proposed instead [2, 8]. Using a binary rather than text-based encoding format raises problems of multiple binary representation of XML [5]. The interoperability with the existing standards also rises as an issue. Many experiments with binary representations of XML lead to conclusion that although the compression is higher, the text compression methods are still competitive in many scenarios [8]. A proposal in direction of solving the verbosity and bandwidth issues suggests efficient compression [6]. This would keep the basic structure of data as text-based XML, and there are a numerous compression techniques [7] that are adequate for compressing XML. Suggestions are diverse so much that a proposal for data transfer over FTP has also been raised [9], instead of using standard HTTP protocol. However, additional requirements such as larger disk space for data storage appear when using FTP, so compression technique is again needed. Adaptive SOAP (or A-SOAP) [10] goes one step further and focuses, besides on compressing messages, also on accelerating message composition and reducing parsing overheads. Its endpoints do not exchange dictionaries in advance, rather as the communication progresses. So far, Microsoft has already discussed the compression of web services payload [11] and the positive impact of SOAP optimization in terms of reduced costs and more reliable transfer. The proposed mechanism of extending SoapExtensionAttribute and SoapExtension, the two classes of System.Web.Services.Protocols namespace, is the same one applied in the testbed that is presented later in this paper. Although all of these compression techniques produce good compression ratios, not many effectiveness studies have been conducted. This paper looks more into these optimization techniques and the impact they have on the performance.

One other aspect which is a key attribute to the overall performance of the portable devices is certainly the battery life. Extending the battery life has not only been a challenge on the hardware level, but also there have been several proposals on battery life improvement, one of which seems like ingenious one [3]. The solution proposed by the HP team consists of rapid switch of the state of the processor in standby mode and activating it back when a button is pressed or the screen is touched. In the meantime, the display maintains the last screen so the user has an illusion that the device is working fully active all the time. However, this solution would only work when using the device in a passive mode (watching/analyzing data at the screen, reading texts, etc.). In a case when continuous connectivity and processing is needed, the text compression seems like one of the most reasonable approaches.

III. TEXT COMPRESSION

Compressing text is a procedure that can reduce the size of a text file up to 80% which means that the compressed text can be stored using 80% less space than uncompressed text. It also means that the content needs less time to be transferred over a network, which translates into higher performance for client-server applications that communicate with text, like the Web services themselves. As introduced in Section I, the Web Services exchange data with client applications (or other web services, for instance), in a pure text-based format, i.e. XML structured SOAP messages. Since these data is highly redundant in its reserved xml characters and tags, the compression of such text could be looked at as a quite beneficial.

Most of the project designers try to find a way to lower the size of the data which is passed between the clients and the servers. Few experienced engineers are using advanced techniques to improve performance of the data transfer through a network, but the overhead accompanying this data transfer remains a bottleneck in many distributed systems. There are two possible solutions to this issue. One is to increase the bandwidth, but not always is such a solution possible or practical. Increasing bandwidth can also significantly increase the overall costs of the solution. The other solution is to decrease the amount of data that is travelling through the network. And “decrease” in this term means nothing more than compressing the text. This second approach has a trade-off itself, too, but not as costly as increasing bandwidth. It requires extra processor time at the client and at the server, respectively, so that the messages could be compressed at one side and then decompressed at the other, and vice versa.

IV. IMPLEMENTATION USING SOAP EXTENSIONS

Fig. 1 shows the cycle of exchanging messages using SOAP. At client side, the web reference in form of a client proxy holds the necessary interface for communicating with the real web service across the internet.

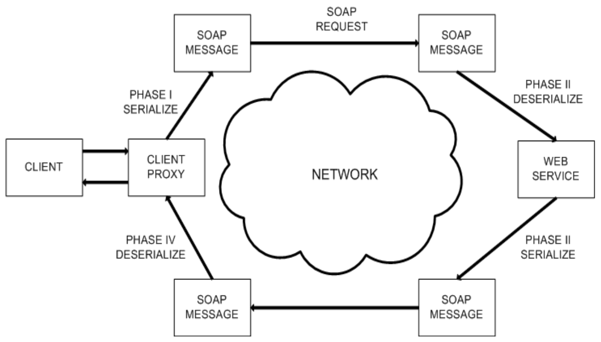

Fig. 1. XML Web Services Architecture

The proxy holds the knowledge of how each data that needs to be sent over the network to the web service as an input parameter should be serialized as text. The data is then serialized in a text-based SOAP message and at the other end, right before it is accepted at the web service side, it is being de-serialized. The web service then processes the request and returns the result which before passing across the network is serialized and then, before reaching the proxy de-serialized back to its original form (the original object), again with the knowledge contained in the proxy. This means that the data is transformed into text-based message right after serialization and it keeps format until the de-serialization at the other end. Using SOAP extensions, it is possible to intervene in this process and compress the messages after the serialization is complete. After travelling across the network as compressed text (TCP does not need to read the data, it only cares for accurate transfer of bytes), they need to be put back in understandable format before de-serializing.

As Fig. 2 shows, some processor time both at the client side and the server side needs to be traded off in order to compress all outgoing serialized data and decompress all incoming data, and then pass it to be de-serialized.

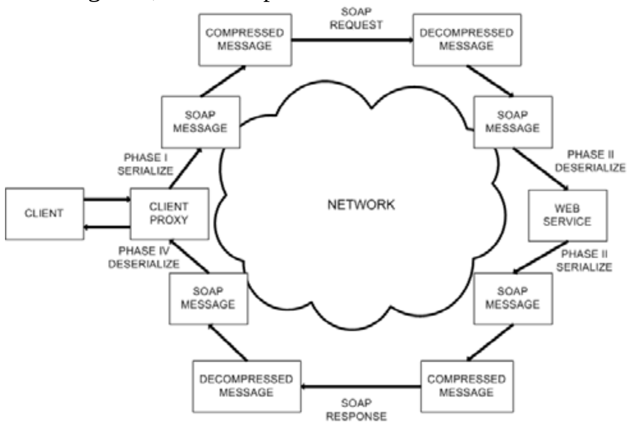

Fig. 2. Implementation of Compression and Decompression after serialization and before deserialization

In the first situation, when we have a usual no-compression-involved transmission, the total time needed for a client request to be processed would be given in (1):

$$T_{Total} = 2 \cdot (t_{ser} + t_{tr} + t_{deser}) + t_{servproc} \quad (1)$$

Where t_{ser} is time needed to serialize the request in xml format, i.e. prepare it to be sent via SOAP protocol, t_{tr} is the time needed to actually transfer the serialized request, t_{deser} is the time needed to de-serialize the xml text into objects understandable for the applications and $t_{servproc}$ is the time needed for processing the request and producing the results at the server side. The same circle of serialization, transport and de-serialization happens in the opposite direction as well and therefore the quotient before the sum of t_{ser}, t_{tr} and t_{deser} is 2.

In the situation when optimization is used, additional time for compression / decompression at both server and client side is needed. So the total time needed for processing the request would be given in (2):

$$T_{Total} = 2 \cdot (t_{ser} + t_{tr} + t_{deser}) + t_{servproc} + t_{com/dec} \quad (2)$$

One additional time appears in the latter formula, $t_{com/dec}$ stands for the average time needed for compressing and decompressing the text. More accurately, $t_{com/dec}$ is a sum of the time needed to compress the serialized text at the client side, the time needed to decompress this data into serialized text at the server side, right before de-serialization, the time needed to compress again the serialized text, but this time at the server side and finally, the time needed to decompress the compressed data into serialized text, before de-serialization at the client side.

V. PERFORMANCE ANALYSIS

In order to get the true impression about the impact the text compression has over the passing SOAP messages, a proper testbed has been set. The testbed, as seen in Fig. 3, consists of a remote mobile device – HTC P3300 with a SIM card included (with GPRS activated), and a web server across a public Internet network. From software components, the mobile device has a portable application installed, and the web server hosts a web service published on the Internet, interconnected with a database. The mobile application is implemented using Compact Framework .NET 2.0 and references the web service. The web service is implemented using .NET Framework 2.0 and the database is hosted at SQL Server 2005.

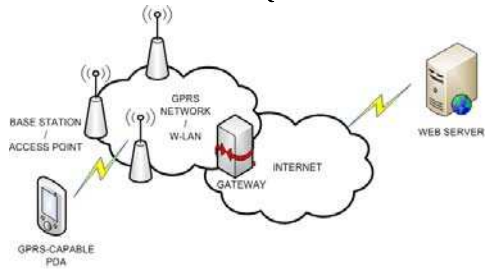

Fig. 3. The testbed

The test scenario was conducted as follows: the mobile application issues requests over wireless network to the publicly hosted web service. This web service then completes the processing, communicating with a database in the meantime, and returns back the results, again over the wireless network. All the communication takes place conforming to the XML web service and SOAP messaging standards discussed before. Two variations of this scenario were actually conducted, one without optimization included, and another test with the necessary soap extensions implemented. What is meant under wireless network are actually two variations. First a 801.11 WiFi wireless network was used as a transport medium, and then a GPRS network was used. Several experiments have been conducted. In each of them different requests in terms of different data loads have been issued first over WiFi and then over GPRS network. The same requests were then issued, but this time using optimization.

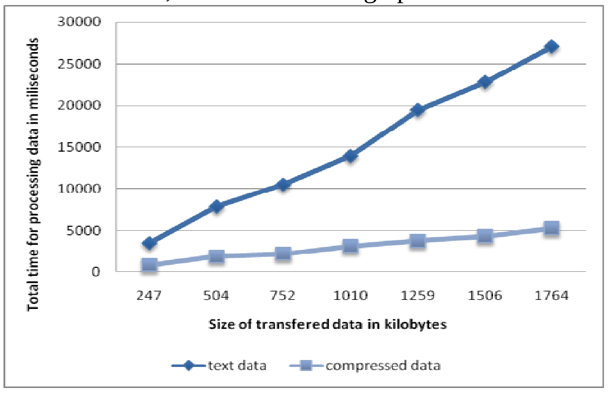

Fig. 4. Time needed to process client requests via Wi-Fi

The results of the experiment conducted over a Wi-Fi network are given in Fig. 4. As we can see from the chart, there is a significant reduction of time needed for the complete cycle of processing one client request to be completed. For instance, if there is a request for 2.500 rows from the database, the amount of data that would have to be carried back to the client is 1259kB. For a Wi-Fi environment, a normal uncompressed transfer would require 19.5 seconds. When using compressed data, this time drops to just 3.8 seconds. Several requests were processed, starting from 500 data rows, 1.000, 1.500, and so on, up to 3.500 data rows requested. With each increase of 500 data rows more being requested, i.e. around 250kB increase in traffic, the total time needed to process a request would rise for around 4-5 seconds. For the initial 500 rows requested, the T_{total} equals 3.5 seconds, but for 3.500 data rows requested (i.e. 1764kB data) the total time rises up to over 27 seconds. However, when using the compression and decompression, the total time significantly drops, for the first case of 500 data rows it is less than a second, and for the highly loaded case of 3500 data rows requested, the total time remains under 6s which is in fact an improvement of more than 80%.

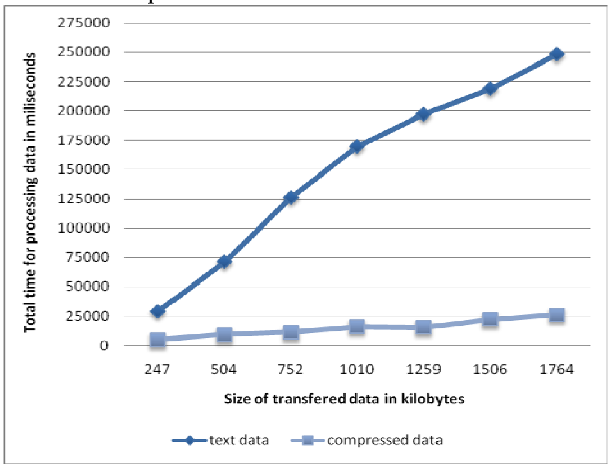

Fig. 5. Time needed to process client requests via GPRS

If we look at the next chart in Fig. 5, the results are more than promising. Due to the slower communication, the uncompressed transfer here takes quite more time. For the least loaded request of just 500 data records, it takes less than 30 seconds, but 30 seconds could mean a must-not-happen eternity in client real-time application. The highest loaded request of 3500 data records took staggering 248 seconds to be processed. The implemented compression/decompression decreased T_{total} for about 90% of its initial value without optimization, lowering it down to acceptable 22 seconds.

This significant reduction of the time needed for a complete client request processing is mostly due to the reduction of data by compressing the text by using the Zip algorithm with size reduction ratio of 5:1. The trade-off time needed for server side compression/decompression is less than 0.1s, and on the client side (HTC device) it can go slightly over 1s for large data. So, $t_{com/dec}$ is much lower than the time of data transfer t_{tr}, therefore the T_{total} mostly depends on the size of the data to be transfered, so when this size is reduced 5 times, the T_{total} drops in similar ratio.

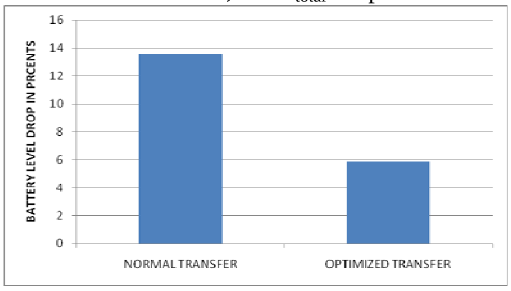

Fig. 6. Battery level drop (%), during normal and optimized transfer

We are not interested just in the time performance. Since the transponder/receiver of the device works for a significantly less time, especially when GPRS is used, the battery of the device is saved. Two tests were conducted – each consisting of 15 consecutive requests issued over the wireless medium from the mobile application towards the server. In the first test – without optimization, the battery level drops from level for more than 13%. When optimization is used in the second test, the battery level drops for just less than 6%, which is a great improvement. Graphical representation is given in Fig. 6. So the overall energy saving is more than double. Presenting results in percents is important, because the values measured could vary from device to device.

VI. CONCLUSION

The solution presented in this paper provides the opportunity to transfer data in a secure, reliable way which has proved to improve the performances of mobile devices using wireless transfer to communicate with a server. The improvement is not only demonstrated in shorter time needed for the transfer to be completed, but also in the extended battery life, which for mobile devices is of great importance. The SOAP extensions responsible for compression and decompression of SOAP messages are easily applicable to every web-service-based application and the effects presented in this paper are common for every similar environment.

The future work would mainly focus on creating a new lower-level layer that would deal with compression and decompression of the data traffic. In that way, instead of additional programming, a completely new transport layer extending the existing TCP protocol would be responsible for the traffic optimization.

REFERENCES

[1] Cai, M., Ghanderizadeh, S., Schmidt, R., et al. “A Comparison of Alternative Encoding Mechanisms for Web Services”, Proceedings of the DEXA 2002

[2] Alex Ng, Paul Greenfield, Shiping Chen, “A Study of the Impact of Compression and Binary Encoding on SOAP Performance”, Department of Computing, Macquarie University – North Ride, Australia, AWSA 2005

[3] Lawrence S. Brakmo, Deborah A. Wallach, Marc A. Viredaz, “µSleep: A technique for Reducing Energy Consumption in Handheld Devices”, HP Laboratories, Palo Alto, January 2004

[4] Chiu, K., Govindaraju, M., and Bramley, R., “Investigating the Limits of SOAP Performance for Scientific Computing”, Proceedings of 11th. IEEE International Symposium on High Performance Distributed Computing HPDC-11 2002 (HPDC'02)

[5] Pal, S., Marsh, J., and Layman, A.: “A Case against Standardizing Binary Representation of XML”, Proceedings of the Workshop on Binary Interchange of XML Information Item Sets. 2003

[6] Girardot, M. and Sundaresan, N.: “Millau: an encoding format for efficient representation and exchange of XML over the Web”, Proceedings of IX IWWWC, Amsterdam, May 2000

[7] Liefke, H. and Suciu, D. XMill: “An Efficient Compressor for XML Data”, Proceedings of the ACM SIGMOD International Conference on Management of Data. Dallas, USA, June 2000

[8] Software Technology Group, MSI, Växjö University, Växjö, Sweden: “The Effects of XML Compression on SOAP Performance”, World Wide Web – Springer Netherland, July 2007

[9] Tanakorn Wichaiwong, Chuleerat Jaruskulchai, “A Simple Approach to Optimize Web Services’ Performance”, Proceedings of IAIT 2007

[10] Marcel-Catalin Rosu “A-SOAP: Adaptive SOAP Message Processing and Compression”, Proceedings of IEEE International Conference on Web Services 2007, (ICWS 2007) pp.200-207

[11] Andy Wigley, Daniel Mothand, Peter Foot, “Microsoft Mobile Development Handbook”, Microsoft Press 2007