Performance of mobile Web Service Access using the Wireless Application Protocol (WAP)

Guido Gehlen, Ralf G. R. Bergs
Aachen University
Communication Networks
Kopernikusstr. 16, 52074 Aachen
guge@comnets.rwth-aachen.de

Abstract

Accessing XML Web Services from mobile clients is a flexible and promising concept to build complex mobile business applications. Due to the computing constraints of mobile clients and bandwidth limitations of mobile communication systems performance considerations are essential.

Usually Web Services are accessible through standard internet protocols like HTTP, but within mobile communication networks it is quite inefficient. The Wireless Application Protocol (WAP) suite provides protocols optimized for packet oriented communication over mobile links. The interconnection to existing HTTP Web Services is achieved by the use of a WAP gateway.

This article presents a realization to access Web Services from J2ME mobile devices using WAP. For this, a Java Wireless Session Protocol (WSP) implementation has been developed. The performance of this WAP based access compared to HTTP is analyzed in terms of criteria as latency, data transfer volume, memory footprint and CPU power requirements.

1. Introduction

The Internet enables quick access to information and a variety of services in the form of Web Pages. These Web Pages offer one graphic user interface accessible through a web browser. Recently, it has been observed that Web Pages evolve to Web Services, which are user interface independent and enable applications to integrate Internet-specific services.

Mobile devices with their hardware limitations are generally not suitable to use Internet Services via Web Pages. The separation of user interface and service logic offered by Web Services are a new chance to bring internet services to mobile devices. Applications running on mobile devices, providing access to Web Services, can thereby be adapted to the specific device capabilities.

With the integration of Web Service technologies in mobile devices one has to consider the special restrictions of these devices and the mobile communication system. Mobile devices suffer from several limitations, such as slow Central Processing Units (CPUs), small memories, primitive operating systems and small displays. Mobile communication systems, especially GPRS and UMTS, imply limited bandwidth and high latencies.

This paper aims at presenting an alternative solution to access Web Services from Mobile Clients using WAP, as well as a performance analysis of this solution compared to access via Hypertext Transport Protocol (HTTP) from Java-capable mobile phones.

In section 2, the Web Service technologies are explained, and the concept of accessing Web Services from mobile Java phones is introduced. Section 3 illustrates the WAP access implementation for Java phones, which is used for the performance analysis presented in section 4.

2. Mobile Web Services

Predominantly, the technologies used by Web Services [2] are based on the Extensible Markup Language (XML) [15, 8], like the Simple Object Access Protocol (SOAP) [4, 11], the Web Service Description Language (WSDL) [7] and the Universal Discovery Description and Integration (UDDI) [22]. The dependencies of these technologies are depicted in figure 1 in Unified Modeling Language (UML) notation.

WSDL [7], a specialization of XML, describes the Web Service interface, its operations, data formats, transport bindings and endpoints (e.g. a URL).

Web Services can be published by service providers and located by service requestors by means of the UDDI [9]. It is a directory model, the so-called yellow pages of Web Services, whose entries consist of three parts. The "white pages" contain information about the Web Service provider, the "yellow pages" include industrial categories based on standard taxonomies and the "green pages" describe the interface of the service, e.g. by integrating the WSDL file.

The Web Service instances are communicating via a protocol called SOAP [11]. It is based upon messages encoded in XML, transmitted by arbitrary transport protocols, usually HTTP, and allows clients to perform Remote Procedure Calls (RPCs), among other things.

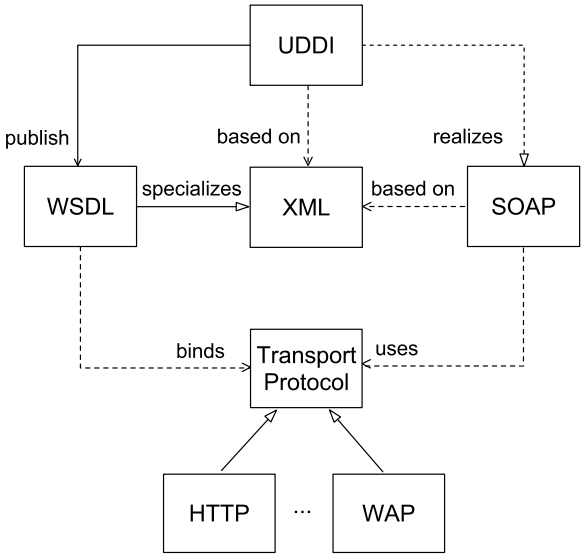

Figure 1. Web Services Technologies and their dependencies in UML notation

To Access Web Services from mobile clients a SOAP implementation and a transport implementation is needed. For this performance analysis solely Java implementations are used, since most of the mobile devices support a Java Virtual Machine (JVM). Currently, two SOAP client implementations for Java2 Micro Edition (J2ME)-capable devices are known, Wingfoot SOAP and kSOAP. Here, Wingfoot SOAP is used, since it enables the integration of further transport protocols in addition to the standard HTTP implementation.

For mobile access the protocol suite HTTP using Transport Control Protocol (TCP) is not an ideal solution due to a great protocol overhead. WAP [24] finds a remedy by reducing header information and protocol overhead. Since it is impossible to access the phone's native WAP implementation from inside the JVM, a Java implementation of the WAP has been developed.

Paying tribute to the limited resources available on mobile clients, the connectionless WSP [23] has been implemented. The advantage of this is the avoidance of the TCP three way handshake, header volume reduction and optionally reduction of the payload data volume using the binary XML encoding WBXML. Wireless Binary XML (WBXML) [25] was developed mainly for low bandwidth networks and restricted Central Processing Unit (CPU) power, and seems appropriate for mobile XML messaging. One further advantage is that mobile clients transmitting SOAP messages via WAP can access all existing Web Services using the usual HTTP binding, since the WAP gateway converts WAP into HTTP and vice versa.

3. WAP Transport Implementation

The WAP implementation has been split in several parts: On the lowest level the software package has the ability to construct request Protocol Data Units (PDUs) and to determine the octet stream for encoded PDUs, and vice versa to reconstruct reply PDUs from an octet stream received via the network. Optionally, the payload can be encoded in WBXML, if it is supported by the WAP Gateway.

One level above is the ability to send request PDUs and to receive reply PDUs to and from the network using User Datagram Protocol (UDP) packets. These PDUs have a field called Transaction ID (TID) used to associate replies with corresponding requests. Thus, this field can be used to detect lost packets.

On the highest level is the implementation of a SOAP transport to be used to convey SOAP data from the mobile application to the web service and vice versa.

3.1. WBXML Encoding

The binary format WBXML was designed to allow compact transmission with no loss of functionality or semantic information, enabling more effective use of XML data on narrowband communication channels.

A binary XML document is composed of a sequence of elements. Each element may have zero or more attributes and may contain embedded content. This structure is very general and does not have explicit knowledge of XML element structure or semantics. This generality allows user agents and other consumers

of the binary format to skip elements and data that are not understood.

All WBXML documents contain a specification version in their initial byte. The version number is followed by a representation of the XML document public identifier. This is used to identify the well-known document type contained within the WBXML entity. Moreover, the binary XML format contains a representation of the XML document character encoding.

Immediately after the charset a binary XML document must include a string table. Minimally, the string table consists of the encoded string table length in bytes, not including the length field (e.g., a string table containing a two-byte string is encoded with a length of two). If the length is non-zero, one or more strings follow.

Various tokens encode references to the contents of the string table. These references are encoded as scalar byte offsets from the first byte of the first string in the string table.

Finally, the body of the binary XML document follows. The body consists of an element optionally headed and/or followed by processing instructions. An element itself consists of a start tag, optionally one or more attributes terminated by an `END` token, and optional content which can, among others, be an element again or a string.

To illustrate the data reduction possible by using WBXML encoding, a SOAP call to the BabelFish translation Web Service of 508 bytes length is encoded in WBXML. The resulting encoded message has a length of 326 bytes, i.e. the original length has been reduced by 35%.

4. Performance Analysis

The performance of mobile applications is a significant aspect, since mobile devices have only limited CPU power and main memory, as well as the mobile communication network offers only a limited bandwidth. The application speed, i.e. how quickly the application responds to user actions, is a crucial factor for the acceptance of these applications and for a commercial use.

In this context the performance criteria latency, data transfer volume, memory footprint and CPU power requirements have been considered. Latency is caused by different points, as creation of objects, network setup, transmission of data, parsing of request and response data objects and response time of the server.

To measure these performance criteria a reference Web Service is used. It offers three methods for benchmarking, plus two methods which are used prior to the actual benchmarking to set up size and contents of the payload. This approach was chosen to reduce the processing time of the benchmarking methods on the web server to an absolute minimum, so as not to falsify the measurements.

The three benchmark methods are the following:

`nop()`	This method does nothing, so it can be used to measure the latency incurred solely for the invocation of a SOAP message.
`getString()`	This method transfers a `String` of pre-determined size.
`getIntArray()`	This method transfers an `int[]` with a pre-determined number of elements.

To be able to run the benchmarking measurements from a J2ME emulator or from a real device, a MIDlet[1] that instantiates and invokes the benchmarking methods has been created.

4.1. Data Transfer Volume

The total volume of protocol data – payload plus protocol overhead, such as headers – transferred for a given payload varies depending on which transport protocol is chosen.

In HTTP, the header fields are not encoded, but transmitted "in cleartext". In contrast, the WSP defines compact binary encodings for the well-known headers, thus the protocol overhead can effectively be reduced.

The protocol overhead, which basically is the size of the headers added to the payload, can be reduced by more than a third by simply choosing WSP instead of HTTP to transport the SOAP messages.

Further reduction, about 35%, of data volume could be obtained by encoding the XML data using WBXML as described in section 3.1, but the reduction factor highly varies depending on the content transmitted.

4.2. Memory Footprint

Fig. 3 shows the components that make up the MIDlet. The SOAP WSP transport class (`comnets.soap.transport.WSPTransport`) and the two packages `comnets.wap.wsp.encoding` and `comnets.wap.wsp.transport` are only needed in the case of WSP being used for communications, thus, the

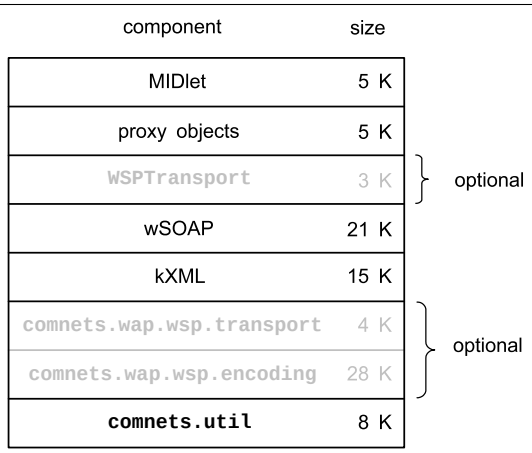

Figure 2. Sizes of MIDlet components (WSP support being optional)

size of the MIDlet using HTTP is 54 K and the size of the MIDlet using WSP 89 K.

4.3. CPU Power

To be able to *roughly* assess the computing power of the mobile phone used for some of the performance tests (a Siemens S55), a benchmarking MIDlet named "TaylorBench"[2], version 1.1, was used. The default options were kept when running the MIDlet.

Table 1 on p. 103 shows the results obtained from running TaylorBench.

Test performed	PC	S55
5000 1000-byte array copies	70 ms	7,407 ms
5000 VM tests	40 ms	-[a]
5000 random ints	40 ms	-[a]
5000 1000-byte controls	30 ms	464 ms

a Device failed to execute test

Table 1. Results obtained from running "TaylorBench" on a PC using the WTK Emulator and on a Siemens S55 mobile phone

Considering these figures, it can be expected that the PC runs MIDlets about 10–100 times faster than the mobile phone.

1 MIDlet - An application that conforms to the Mobile Information Device Profile

2 `http://www.poqit.com/midp/bench/`

4.4. Latency

For all latency measurements the total latency t_{inv} to invoke one reference method is divided into t_{SOAP} and t_{call}. The time t_{SOAP} represents the time used for parsing SOAP messasges, t_{call} is the roundtrip time for the message transmission.

First, the latency performance of the MIDlet running on a Siemens S55 mobile phone is compared to the case where the MIDlet runs on a mobile phone Emulator.

In both scenarios, HTTP was used as a SOAP transport protocol. Also, the Web Service ran on a remote machine in both cases. Finally, in the case of the MIDlet running on the emulator, the client machine was connected to the Internet with an ADSL line, adding about 85 ms per packet round-trip. Thus, the latencies that were incurred by the characteristics of the physical connection (ADSL in one case, GPRS in the other case), are roughly comparable.

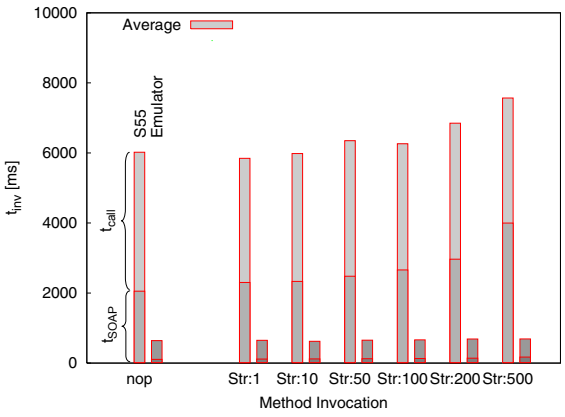

Figure 3. Benchmark Results (t_{inv}), `nop()` and `getString()` method invocations right: MIDlet on S55, using HTTP over GPRS left: S55 Emulator, using HTTP over ADSL

Fig. 4 and 5 show the benchmark results of the two series. The results of the MIDlet running on the S55 are shown on the left columns, the emulator results are shown on the right columns.

What was to be expected by the results of running TaylorBench (section 4.3), now has been confirmed by running the SOAP client MIDlet. The S55 mobile phone is at least a magnitude slower than the PC, resulting in latency values that are about ten times as large for the S55 as for the emulator.

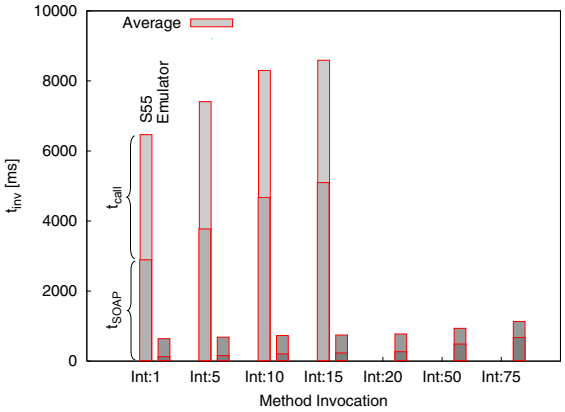

Figure 4. Benchmark Results (t_{inv}), `getIntArray()` method invocations right: MIDlet on S55, using HTTP over GPRS left: S55 Emulator, using HTTP over ADSL

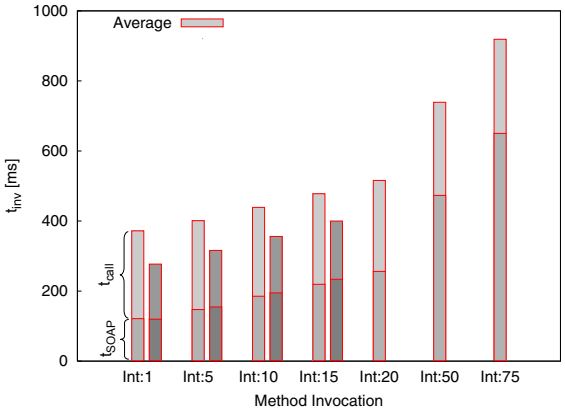

Figure 6. Benchmark Results (t_{inv}), `getIntArray()` method invocations, MIDlet running on Emulator, using WSP (right) and HTTP (left) over ADSL

Second, the latency differences of SOAP calls using WAP and HTTP are presented. In both cases, client and service were running on the local host. The only difference was the SOAP transport protocol used: HTTP in the first case, connectionless WSP in the second case.

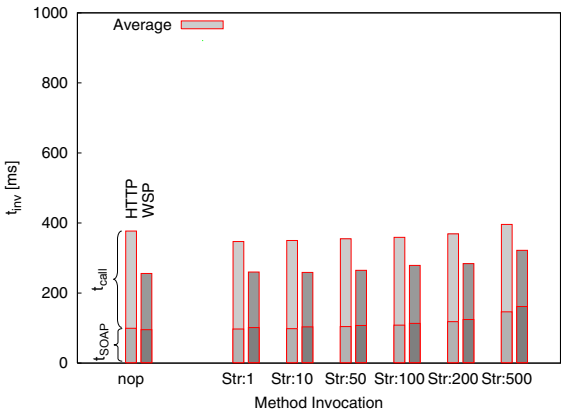

Figure 5. Benchmark Results (t_{inv}), `nop()` and `getString()` method invocations, MIDlet running on Emulator, using WSP (right) and HTTP (left) over ADSL

From this follows that the samples for $t_{\mathrm{call,\,HTTP}}$ (about 250 ms average) are considerably larger than $t_{\mathrm{call,\,WSP}}$ (about 160 ms average). Thus, time for transmission of the SOAP messages (request and response) can be reduced by about a third by choosing connectionless WSP instead of HTTP.

5. Conclusions

Applications on mobile devices accessing Web Services via the Simple Object Access Protocol (SOAP) are even possible on current mobile phones. But the performance of these applications is bad, since a simple SOAP method invocation from a Siemens S55 mobile phone over HTTP and GPRS takes about 6 to 8 seconds.

This work shows that the performance in term of latency can be increased using a WAP binding to SOAP. With WAP it is possible to access all existing HTTP Web Services. The SOAP messages are reduced by a third using the WBXML encoding. The connectionless WSP avoids TCP's three way handshake and reduces the protocol data overhead by more than a third compared to the HTTP overhead.

A Java WSP implementation has been developed since it was not possible to access the phone's native WAP implementation, due to Java's sandbox model. It is expected that performance considerably increases if this native implementation can be used.

References

[1] V. Bansal and A. Dalton. A performance analysis of web services on wireless pdas. Published on the internet. Available at URL http://www.cs.duke.edu/~vkb/advnw/project/, May 2002.

[2] A. Brown and H. Haas. Web services glossary. Published on the internet. Available at URL http://www.w3.org/TR/2002/WD-ws-gloss-20021114/, Nov. 2002. 2

[3] L. Clement. Uddi overview presentation. Published on the internet. Available at URL http://www.uddi.org/pubs/UDDI_Overview_Presentation.ppt, Sept. 2000.

[4] T. Clements. Overview of soap. Published on the internet. Available at URL http://developer.java.sun.com/developer/technicalArticles/xml/webservices/, Jan. 2002. 2

[5] B. Day. J2me faq. Published on the internet. Available at URL http://www.jguru.com/faq/J2ME, July 2003.

[6] Enhydra.org. kxml javadoc. Published on the internet. Available at URL http://kxml.enhydra.org/software/documentation/apidocs/, Feb. 2002.

[7] R. C. et al. Web services description language (wsdl) version 1.2. Published on the internet. Available at URL http://www.w3.org/TR/wsdl12, June 2003. 2

[8] T. B. et al. Extensible markup language (xml) 1.0 (second edition). Published on the internet. Available at URL http://www.w3.org/TR/REC-xml, Oct. 2000. 2

[9] T. B. et al. Uddi version 3.0. Published on the internet. Available at URL http://uddi.org/pubs/uddi_v3.htm, July 2002. 2

[10] J. Knudsen. Parsing xml in j2me - xml in a midp environment. Published on the internet. Available at URL http://wireless.java.sun.com/midp/articles/parsingxml/, Mar. 2002.

[11] N. Mitra. Soap version 1.2 part 0: Primer. Published on the internet. Available at URL http://www.w3.org/TR/soap12-part0/, June 2003. 2

[12] B. Morgan. Deliver mobile services using xml and soap. Published on the internet. Available at URL http://www.wirelessdevnet.com/columns/feb2001/editor14.html, Feb. 2001.

[13] V. N. Padmanabhan and J. C. Mogul. Improving http latency. Published on the internet. Available at http://archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/mogul/HTTPLatency.html, Oct. 1994.

[14] S. Seely. Documenting your web service. Published on the internet. Available at URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service07182001.asp, July 2001.

[15] S. Seely. An xml overview towards understanding soap. Published on the internet. Available at URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/xmloverchap2.asp, Nov. 2001. 2

[16] A. Skonnard. Understanding xml namespaces. Published on the internet. Available at URL http://msdn.microsoft.com/library/default.asp?url=/msdnmag/issues/01/07/xml/TOC.asp, July 2001.

[17] D. Srinivas. Soap faq. Published on the internet. Available at URL http://www.jguru.com/faq/SOAP, Jan. 2002.

[18] I. Sun Microsystems. Midp style guide. Published on the internet. Available at URL http://java.sun.com/j2me/docs/alt-html/midp-style-guide7/index.html, Aug. 2002.

[19] Mobile information device profile (jsr-37), java 2 platform, micro edition, version 1.0. Published on the internet. Available at URL http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html, Sept. 2000.

[20] Personaljava application environment specification version 1.2a. Published on the internet. Available at URL http://java.sun.com/products/personaljava, 2000.

[21] C. C. Tapang. Web services description language (wsdl) explained. Published on the internet. Available at URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wsdlexplained.asp, July 2001.

[22] uddi.org. Uddi data structure reference v1.0. Published on the internet. Available at http://uddi.org/pubs/DataStructure-V1.00-Published-20020628.pdf, Jun 2002. 2

[23] WAP Forum. *Wireless Application Protocol. Wireless Session Protocol Specification*, July 2001. 2

[24] Wireless application protocol architecture specification. wap-100-waparch. Published on the internet. Available at URL http://www.wapforum.org, Apr. 1998. 2

[25] WAPForum. Binary xml content format specification. version 1.3, wap-192-wbxml-20010725-a. Published on the internet. Available at URL http://www.wapforum.org, July 2001. 2

[26] Wireless markup language specification. Published on the internet. Available at URL http://www.wapforum.org, Feb. 2000.

[27] R. Wolter. Xml web services basics. Published on the internet. Available at URL http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/webservbasics.asp, Dec. 2001.