Performance Evaluation of RESTful Web Services for Mobile Devices

Hatem Hamad, Motaz Saad, and Ramzi Abed
Computer Engineering Department, Islamic University of Gaza, Palestine

Abstract: *Smart Mobile devices and web services are becoming very popular. Mobile devices are physically constrained devices; low processor speed, limited memory, limited battery, and slow intermit wireless connection. This implies to take in consideration these factors when implementing web services for mobile devices. In this paper, we evaluate the RESTful web service for mobile devices against conventional SOAP web services. The experimental results show that RESTful web services outperform conventional SOAP web services. A recommendation to use RESTful web services on mobile devices has been concluded from experimental result.*

Keywords: *Mobile Computing, Mobile Web Services, Web Services Performance, RESTful.*

Received April 23, 2009; Accepted June 7, 2009

1. Introduction

Like its predecessors, such as the Common Request Broker Architecture (CORBA), Remote Method Invocation (RMI) and Distributed Component Object Model (DCOM), Web Services [24] is a set of standards and a programming methods for sharing data between different software applications, moreover Web services is a standardized way to distribute services on the Internet.

Web Services achieves its goal in a technology-neutral manner; it provides well-defined interfaces for distributed functionalities, which are independent of the hardware platform, the operating system, and the programming language. So distributed functionalities, or services, which may be running on different hardware platforms, may be running in different operating systems, or may be written in different programming languages, can communicate through web Service interfaces.

Interoperability of Web Services mainly stems from its Extensible Markup Language (XML) based open standards. The Simple Object Access Protocol (SOAP) [8] is defined in XML. Since it is text-based and self-describing, SOAP messages can convey information between services in heterogeneous computing environments without worrying about conversion problems, there are many other Web Service specifications. Two of them, which are based on XML, are Web Service Description Language (WSDL) [4] and Universal Description, Discovery and Integration (UDDI) [23]. WSDL defines a standard method of describing a Web Service and its capability, and UDDI defines XML-based-rules for publishing Web Service information. Messages are exchanged through the SOAP protocol. SOAP works by exchanging information using GET/POST over HTTP. This allows the data to be exchanged regardless of where the client is in the network.

Just as Web Services technology has become an industry standard for connecting remote and heterogeneous resources, mobile devices have become a vital part of people's everyday life. People use mobile devices anytime and anywhere, they may use their mobiles to check Email, access the Internet, or run other web applications.

Web Services technology recognizes mobile computing as an area to which it should expand. Through integration, Web Services enable pervasive accessibility by allowing for user mobility as it overcomes the physical location constraints of conventional computing. However, mobile computing also requires a technology that connects mobile systems to a conventional distributed computing environment. Web Services may be the perfect candidate for such connection, since a strong interoperable capability is the key requirement of the technology. This will be important for its success when we consider the fact that the mobile computing environment is much heterogeneous in terms of hardware platform, operating system, and programming language. Thus, the integration of mobile computing with Web Services technology will give many advantages to both sides. Mobile devices getting computationally capable, so mobile devices enabled with web services can be equal participant of web services architectures (can be web service client or web service provider).

However, despite the fact that the condition of mobile computing has so much improved in recent years [14], applying current Web Service

communication models to mobile computing may result in unacceptable performance overheads. This potential problem comes from two factors. First, the encoding and decoding of verbose XML-based SOAP messages consumes resources. Therefore Web Service participants, particularly mobile clients, may suffer from poor performance. Second, the performance and quality gap between wireless and wired communication will not close quickly. It is caused by the mobile environment's constraints like limited processor speed, limited battery lifetime, and slow unreliable and intermit connection.

Mobile web services is an open research area [2, 3, 5, 13, 22, 27]. Several messaging optimization approaches have been introduced to the literature [20, 26, 15, 16, 17, 18, 19, 25] to address web service performance overhead for mobile devices. As described previously, applying current Web Service communication models to mobile computing may result in unacceptable performance overheads. The typical web application that requires the transmission of four to five times more bytes if implemented as a Web service compared to the same service implemented as a traditional dynamic program (e.g. Active Server Page application) [30] (more details in state-of-art section).

To the best of our knowledge, the performance of Representational state transfer (RESTful) web services [6] has not been evaluated on mobile devices. In this paper, we evaluate the performance of RESTful web services compared to the performance of conventional SOAP web services for mobile devices. Representational state transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. It is a style of web services use. It attempts to emulate HTTP and similar protocols by constraining the interface to a set of well-known, standard (generic) operations (e.g., GET, POST, PUT, DELETE). Here, the focus is on interacting with stateful resources, rather than messages or operations. RESTful offers a perfectly good solution for the majority of implementations, with greater flexibility and lower overhead.

The rest of the paper is organized as follows: in section 2, we review state-of-arts. Section 3 illustrates RESTful web services. In section 4, we present the implementation environment. Performance benchmarks are presented in section 5. Finally, section 6 concludes the paper status.

2. State-Of-Art

Web Services in a mobile computing environment face performance-degradation problems similar to those of the conventional distributed computing environment. So, a primary research issue in the area of mobile Web Services is the attempt to provide an efficient message processing scheme while preserving XML's interoperability.

XML overhead investigation has been performed [22]. The investigation evaluated the overhead of a regular web application compared to a web service that serves the same business function. The typical web application that requires the transmission of four to five times more bytes if implemented as a Web service compared to the same service implemented as a traditional dynamic program (e.g. Active Server Page application). Figure 1 shows the overhead of ASP and web service.

Works on solving this problem can be categorized as either individual message optimization or as message stream optimization [14]. An individual message optimization approach produces a simplified, efficient, and self-contained message, which is a different format (or representation) to XML. The messages in the different representation can be converted to and from the XML format, which is called roundtripping. For example, Fast Infoset (FI) from Sun Microsystems [17, 19] and XBIS [20, 25] fall into this category. On the other hand, the message stream approach optimizes a whole sequence of related messages, which we define as a stream. This approach includes a certain form of negotiation to define stream characteristics, and optimized message representation in the stream. Examples of this category include Fast schema from Sun Microsystems [26, 18] and Handheld Flexible Representation (HHFR) architecture [15, 16]. Table 1 summarize the categorize XML optimization efforts.

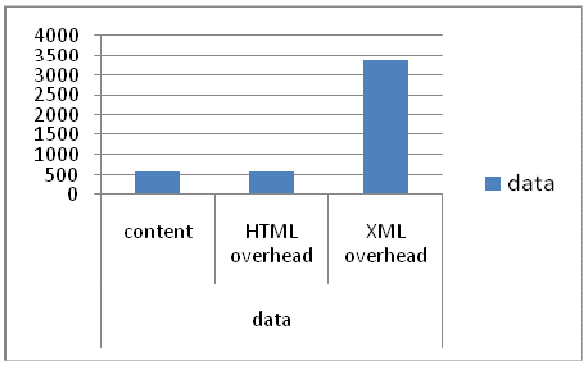

Figure 1. Overhead of server page and Web service [23]

Table 1. Categorized XML optimization efforts.

Individual Message Approach (Self-contained message)	Steam of Message Approach (Non Self-Contained Message)
Fast Infoset of Sun Microsystems	ExtremeFastWS
XML Schema-Based Compression	Fast Web Service of Sun Microsystems
XML Infoset Encoding (XBIS)	Handheld Flexible Representation (HHRF)

Another message optimization method is to compress XML – especially when the CPU overhead required for compression is less than the network

latency [11, 22]. The object model for Axis2 [1], called the Axis Object Model (AXIOM) has an interesting approach for processing headers. Another message optimization approach is to attaching binary data to SOAP message. Examples of this include Message Transmission Optimization Mechanism (MTOM) [9], XML-binary Optimized Packaging (XOP) [10] and Direct Internet Message Encapsulation (DIME) [12].

3. RESTful Web Service

REST [6] is a software application architecture modeled after the way data is represented, accessed, and modified on the web. In the REST architecture, data and functionality are considered resources, and these resources are accessed using Uniform Resource Identifiers (URIs), typically links on the web. The resources are acted upon by using a set of simple, well-defined operations. The REST architecture is fundamentally client-server architecture, and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture, clients and servers exchange representations of resources using a standardized interface and protocol. These principles encourage REST applications to be simple, lightweight, and have high performance.

RESTful web services [6] are web applications built upon the REST architecture. They expose resources (data and functionality) through web URIs, and use the four main HTTP methods to create, retrieve, update, and delete resources. RESTful web services typically map the four main HTTP methods to the so-called CRUD actions: create, retrieve, update, and delete. Table 2 shows a mapping of HTTP methods to these CRUD actions.

Table 2: HTTP Methods and their Corresponding CRUD Action

HTTP Method	CRUD Action
GET	Retrieve a resource.
POST	Create a resource.
PUT	Update a resource.
DELETE	Delete a resource.

3.1. RESTful Web Services and Other Styles of Web Services

REST web services [6] share many characteristics with other styles of web services like remote procedure call (RPC) and document-based web services that use SOAP as the underlying protocol, but also differ in several important ways. RPC and document-based web services, like REST web services, are designed to be lightweight, accessible via URIs, and typically use HTTP as the underlying protocol. REST and SOAP-based web services are also platform and programming language independent, and in both architectures clients and servers are loosely coupled. That is, clients and servers interact with a limited set of assumptions about each other.

REST web services were developed largely as an alternative to some of the perceived drawbacks of SOAP-based web services. The SOAP protocol was designed as a way to make remote procedure calls via HTTP, using XML as the underlying data format, and using standard XML types. Eventually the RPC aspects of SOAP web services were augmented with a document-based architecture, where clients and servers exchange XML documents to enact some change in the client or server applications. As the use of SOAP web services evolved, the architecture was expanded to deal with more complicated application functionality, like security and message reliability. As a result, developing SOAP web services and clients has become more complicated.

REST web services aim to be simple, and this is accomplished by limiting the types of operations one can perform on a resource. REST founders claimed that it [6]:

- Provides improved response times and server loading characteristics due to support for caching.
- Improves server scalability by reducing the need to maintain communication state.
- Requires less client-side software to be written than other approaches, because a single browser can access any application and any resource.
- Depends less on vendor software than mechanisms which layer additional messaging frameworks on top of HTTP.
- Provides equivalent functionality when compared to alternative approaches to communication.
- Does not require a separate resource discovery mechanism, due to the use of hyperlinks in content.
- Provides better long-term compatibility and evolvability characteristics than RPC. This is due to:
- The capability of document types such as HTML to evolve without breaking backwards- or forwards-compatibility.
- The ability of resources to add support for new content types as they are defined without dropping or reducing support for older content types (MIME types).

4. Implementation Details and Benchmarking Environment

To evaluate the performance of RESTful web services against conventional SOAP web services, we implement a RESTful web service and a conventional web service and develop a web service client on a mobile device for each class of web services. Next, we shall illustrate the service implementation, the client implementation and emulator configuration, and the benchmarking environment.

4.1. Service Implementation

We implement RESTful and conventional SOAP web service and host them on the Glassfish application server. Glassfish [7] is a web service framework developed at Sun Microsystems. The service provider, Glassfish Web Service container runs on IBM compatible PC with 3.2 GHz processor and 1 GB RAM, where Windows XP professional with Service Pack 2 operates. And mobile applications (service client) implemented using J2ME and runs on Sun Mobile Emulator (Sun Java™ Wireless toolkit 2.5.2 for CLDC [21]) which was configured to emulate a VM speed of 512 bytecodes/millisecond, and a network throughput of 9600 bits/second. Emulators profile is MIDP 2.1 and its configuration is CDLC 1.1. The time stamps are measured on mobile side (a session initiator) using System.currentTimeMillis() of MIDP 2.1 - CLDC 1.1 that returns 10 milliseconds precision time stamps. Figure 2 depicts the Emulated experiment environment. Sun Mobile Emulator is depicted in the appendix A.

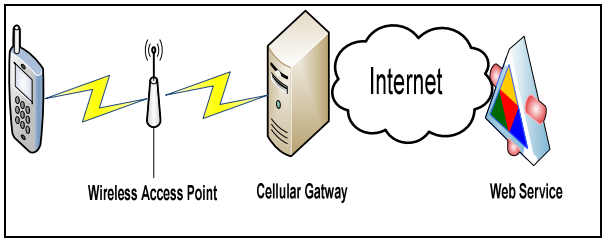

Figure 2. Emulated Experiment Environment for Performance Evaluation

4.2 Benchmark Configurations

We implement two benchmarks using two different data types as parameters to the web service: float data type, and string data type. We measure a total session time and a message size of service call. Benchmarking web services are listed below:

4.2.1. String Array Concatenation:

The first benchmark web service is a *string array concatenation service* that produces a single concatenated string of all string in a message (a pure-text data domain). We measure the response time of the service call. It includes the communication set-up latency, the transmission overhead, and the concatenation operation time. The benchmark focuses on the performance effect on runtime system by changing a number of array elements (size of array) in a message.

4.2.2. Floating Number Array Addition

The second service we benchmark is floating numbers addition service that returns a summation of all float numbers of an array in a message. The float numbers are representing a float data domain. It is remarkable that conventional Web Services message processing includes a float-to-text conversion that consumes many processing cycles. In addition to service response time, the SOAP application contains an OS level float to-text conversion overhead. Like string concatenation service benchmarking, we change the size of the array to observe the performance state change in the system.

5. Experimental Results

Table 3 shows the benchmarking results of the string concatenation and float numbers addition web services which are depicted in Figures 3, and 4. Figure 3 shows the messages size in bytes for string concatenation and float addition services. The message size of RESTful web service is smaller than messages of Conventional SOAP web service. The figures show higher advantage of using RESTful web service. Figure 4 shows the messages response time in milliseconds for string concatenation and float addition services. The response time of RESTful web service is smaller than messages of Conventional SOAP web service. The Figures show higher advantage of using RESTful web service. The gap is very large between the response time of RESTful and the conventional SOAP web service.

Less Message size and response time means less processing and transmission time which leads to lower power consumption, and faster web service. This satisfies the physical constraints of mobile devices and achieves the quality of service goal. These results support that RESTful web service is recommended for mobile devices. Therefore, REST offers a perfectly good solution for the majority of implementations with greater flexibility and lower overhead

Table 3. Service Response Time (Milliseconds) and Message Size (Bytes) of String Concatenation and float addition service.

Number of array elements	Message Size (byte)				Time (Milliseconds)			
	SOAP/HTTP		REST (HTTP)		SOAP/HTTP		REST (HTTP)	
	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition
2	351	357	39	32	781	781	359	359
3	371	383	48	36	828	781	344	407
4	395	409	63	35	828	922	359	375
5	418	435	76	39	969	1016	360	359
6	443	461	93	43	875	953	359	359
7	465	487	104	47	875	875	469	360
8	493	513	127	51	984	875	437	344

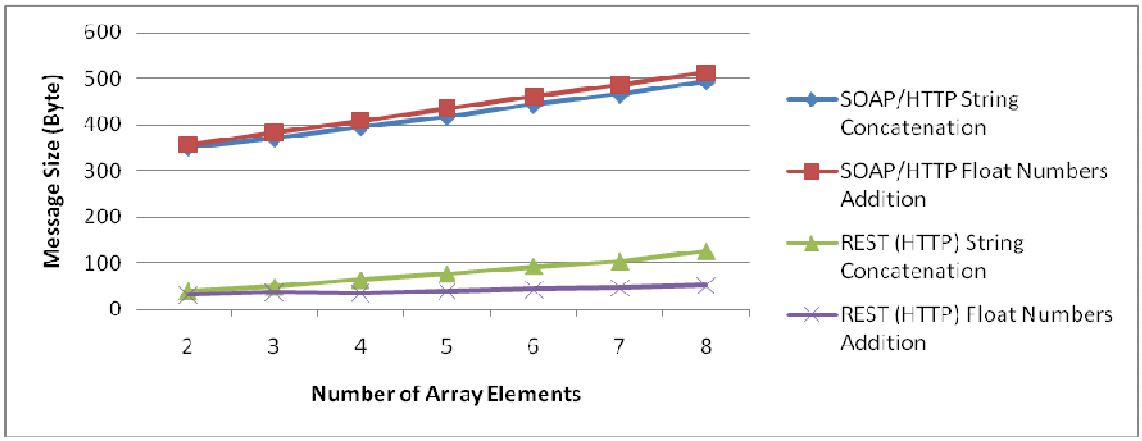

Figure 3. RESTful vs. SOAP message sizes of string concatenation and float addition service.

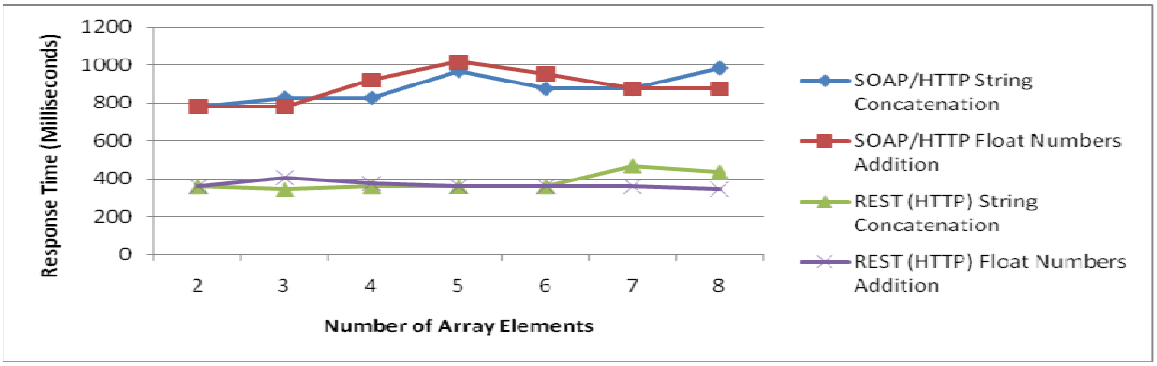

Figure 4. RESTful vs. SOAP Response time of string concatenation and float addition service.

6. Conclusion

We have evaluated a RESTful web service for mobile devices, where we developed RESTful and conventional SOAP benchmarking web service. Benchmarking includes string concatenation and float number addition web services. The performance evaluation results show the advantages of using RESTful web services over conventional web services for mobile devices. Advantages include less message sizes and response time. Results of performance comparison between conventional SOAP and RESTful show the obvious high performance RESTful over SOAP. Therefore, RESTful offers a perfectly good solution for the majority of implementations, with higher flexibility and lower overhead.

Acknowledgments

We would like to thank Dr. Rebhi S. Baraka for his comments.

References

[1] Apache AXIS2, *http://ws.apache.org/axis2*.

[2] Berger, S., McFaddin, S., Narayanaswami C., Raghunath M., "Web services on mobile devices-implementation and experience," *Proc. Of 5th IEEE Workshop on Mobile Computing Systems and Applications*, 2003.

[3] Cheng S., Liu J., Kao J., Chen C., "A New Framework for Mobile Web Services," *Proc. of the 2002 Symposium on Applications and the Internet* (SAINT.02w).

[4] Chinnici R., Moreau J., Weerawarana S., "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," *W3C Recommendation*, June 2007

[5] Chu H, You C, Teng C, "Challenges: wireless Web services," *Proc. Of 10th International Conference On Parallel and Distributed Systems*, 2004 (ICPADS'04).

[6] Fielding R., "Architectural Styles and the Design of Network-based Software Architectures," PhD Dissertation, University of California, Irvine, California, USA, 2000.

[7] GlassFish - *Open Source Application Server https://glassfish.dev.java.net*

[8] Gudgin M., Hadley M., Mendelsohn N., Moreau J., and Nielsen H., "SOAP Version 1.2 Part 1: Messaging Framework," *W3C Recommendation*, June 2003.

[9] Gudgin M., Mendelsohn N., Nottingham M, and Ruellan H, "SOAP Message Transmission Optimization Mechanism (MTOM)," *W3C Recommendation*, 2005.

[10] Gudgin M., Mendelsohn N., Nottingham M, and Ruellan H, "XML-binary Optimized Packaging (XOP)," *W3C Recommendation*, 2005.

[11] Mani A. and Nagarajan A. "Understanding quality of service for Web services," *http://www-106.ibm.com/developerworks/library/ws-quality.html*, 2002.
[12] Nielsen H., Sanders H., Butek R., and Nash S., "Direct Internet Message Encapsulation," Internet-Draft, June 2002 expires December 2002, *http://www.ietf.org/internet-drafts/draft-nielsen-dime-02.txt*.
[13] Oh S. "HHFR: A new architecture for Mobile Web Services Principles and Implementations," *Technical paper*, 2005
[14] Oh S. "Web Service Architecture For Mobile Computing," *PhD Dissertation*, University of Indiana, Irvine, USA, 2006.
[15] Oh S., Bulut H., Uyar A., Wu W., and Fox G.C., "Optimized Communication using the SOAP Infoset for Mobile Multimedia Collaboration," *In Proceedings of The Fifth International Symposium on Collaborative Technologies and Systems (CTS2005)*, St. Louis, Missouri, USA, 2005.
[16] Oh S., Fox G., "Optimizing Web Service Messaging Performance in Mobile Computing," *Community Grids Laboratory Technical Paper*, 2006.
[17] Sandoz P., Pericas-Geertsen S., "Fast Infoset @ Java.net," *In Proc. of XTech 2005, Amsterdam*, Netherands, 2005.
[18] Sandoz P., Pericas-Geertsen S., Kawaguchi K., Hadley M, and Pelegri-Llopart E., "Fast Web Services," *Java developer's Journal Technical Article*, 2003.
[19] Sandoz P., triglia A., and Pericas-Geertsen S., "Fast Infoset," *Java developer's Journal Technical Article*, 2004.
[20] Sosnoski D., "Improve XML Transport performance Part 1 and 2," *IBM developersWork Article, June 2004. http://www-128.ibm.com/developerworks/xml/library/x-trans1.html*.
[21] Sun Java Wireless Toolkit for CLDC: *http://java.sun.com/products/sjwtoolkit*.
[22] Tian M, Voigt T, Naumowicz , Ritter H, Schiller J., "Performance Considerations for Mobile Web Services," *Elsevier Computer Communications Journal*, Volume 27, Issue 11 , Pages 1097-1105, 2004.
[23] UDDI OASIS Standard: http://uddi.xml.org.
[24] W3C Web Services Activity *http://www.w3.org/2002/ws/*.
[25] XBIS XML Information Set Encoding, *http://xbis.sourceforge.net/*.
[26] XMLBeans, *Apache XML Project, http://xmlbeans.apache.org/*.
[27] Zahreddine W, Mahmoud Q, "An agent-based approach to composite mobile Web services," *Proc. of the 19th International Conference on Advanced Information Networking and Applications (AINA'05),* 2005.

Appendix A: Sun Emulator Configurations

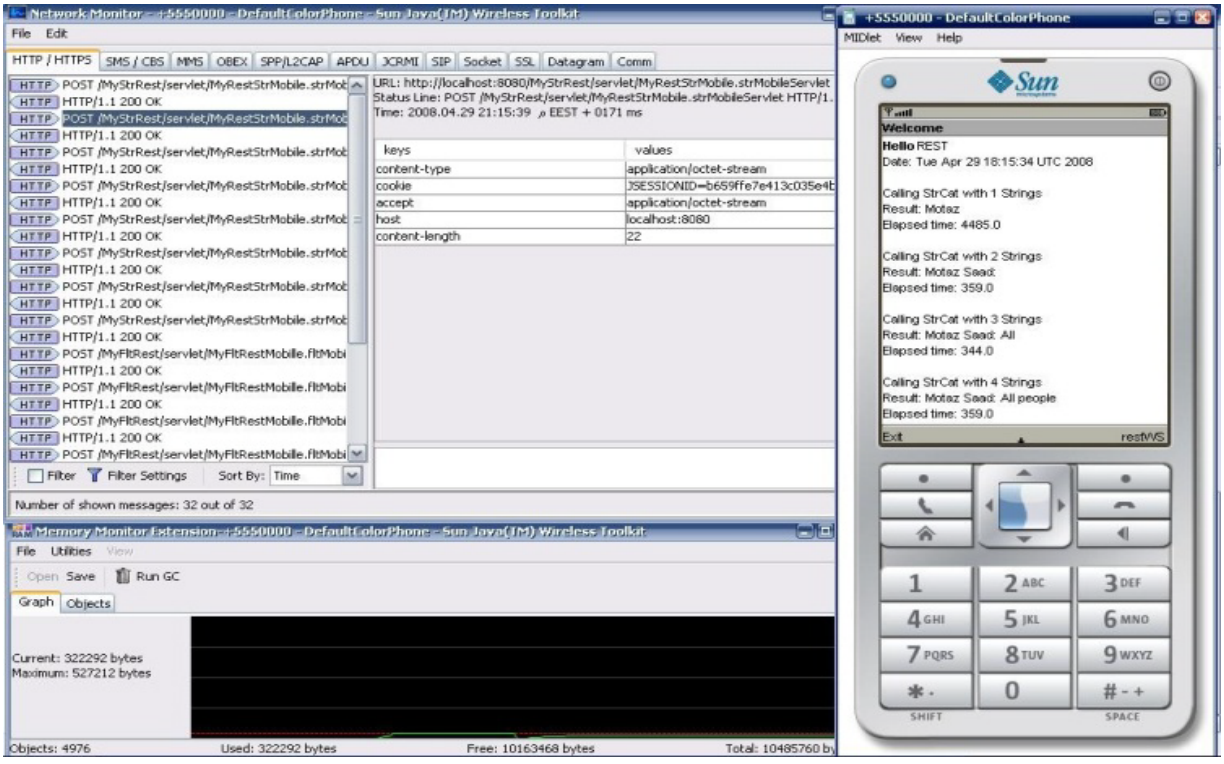

Figure A.1: Sum Mobile Emulator (Sun Java Wireless toolkit)

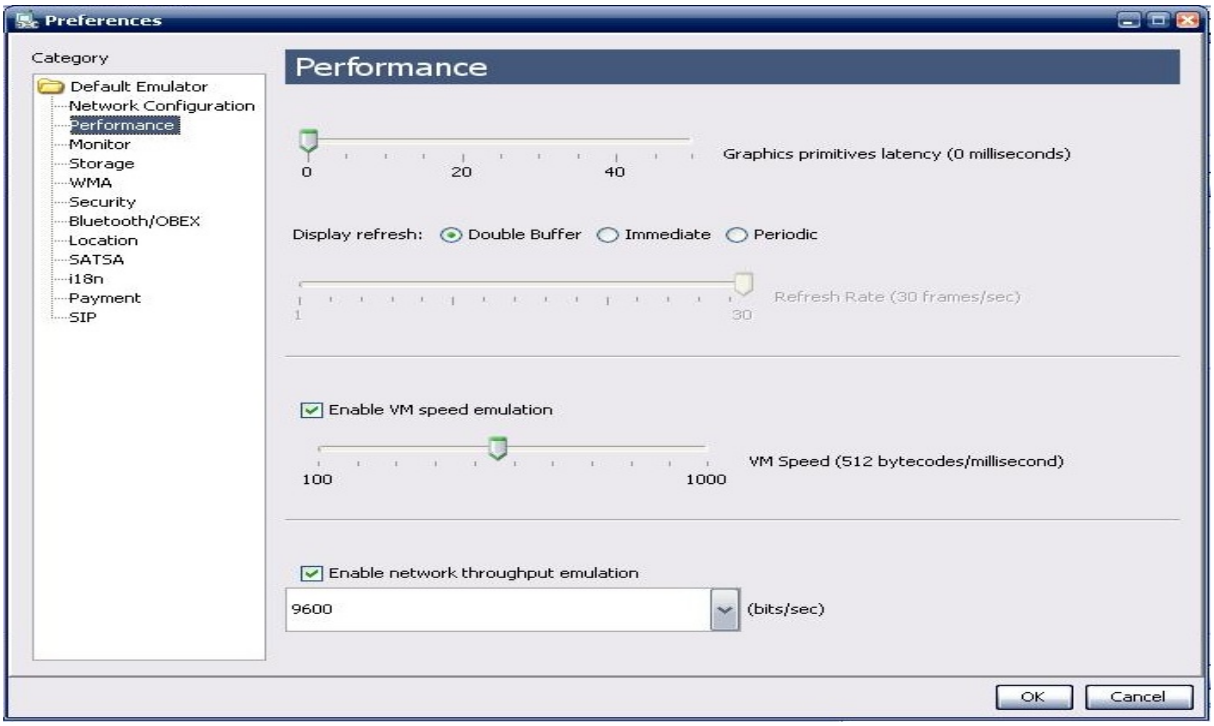

Figure A.2: Sun Java Wireless toolkit preferences

Hatem Hammad was born in Rafah, Palestine on August 31, 1961; he received the PhD in informatics from Technical University Darmstadt, Germany in 1995.from 1995 till now he is being a lecturer in computer engineering Dept. Islamic University of Gaza. His research interests include Web technologies and mobile computing.

Motaz Saad has received his BSc in Information Technolology in 2006 from the Islamic university of Gaza, Palestine, currently working as teaching assistant at the same university since 2006. He is MSc student in computer engineering at the same university. His area of research is Data Mining , Web Services, Natural Language Processing, and Open Source Software.

Ramzi Abed was born in Jeddah, Saudi Arabia, on July, 1976, he received the B. Sc. Degree in Math and Computer Science from the Islamic University of Gaza, Palestine in 1999. He is an M. Sc. Student in Computer Engineering at the Islamic University of Gaza. From 1999 till now he is being a teaching assistant in Information Technology Faculty, the Islamic University of Gaza. His research interests include Information Security, Network Security, and Web Services Technologies and Security. Abed gets an MCP certificate.