EVALUATION OF WEB SERVICES IMPLEMENTATION FOR ARM-BASED EMBEDDED SYSTEM

Mitko P. Shopov, Hristo Matev, Grisha V. Spasov

Department of Computer Systems and Technologies, Technical University of Sofia, branch Plovdiv, 61 "St. Petersburg" Blvd., Plovdiv 4000, Bulgaria, phone: +359 32 659758, www: http://net-lab.tu-plovdiv.bg/ , e-mail: {mshopov, gvs}@tu-plovdiv.bg, hristo.matev@gmail.com

The paper presents test-bed experiments for evaluation of Web services implementation for ARM-based embedded system running embedded Linux 2.6. The gSOAP Web services generation toolkit optimized for embedded devices is used. Security is also included in the experiments with the use of SSL/TLS and WS-Security. Two Web services are developed for the experiments: Echo and Temperature. They both are tested as a standalone application and as a CGI application running in the context of Apache Web server. The services are tested with gSOAP and .NET Web services clients. The presented work is in the scope of Multi-tiered architectures for building Distributed Automation Systems (DAS) and suggests one possible configuration of the automation tier.

Keywords: Distributed Embedded Systems, Web Services, gSOAP toolkit.

1. INTRODUCTION

One of the most promising trends from the recent years – the service oriented architecture (SOA) is now emerging in the domain of distributed embedded systems. One of the most important benefits of such a shift is the possibility to replace traditional vendor specific solutions with popular open standards for communication and to satisfy the arising need to connect distributed embedded devices within the network of enterprise systems.

Although the resources and capabilities of embedded systems are continuously increasing while their price continues to drop [1], the rate is not enough to fully apply the popular enterprise technologies and protocols directly on embedded systems. Thus, there is always a need for optimizations and implementation techniques to achieve improved performance with minimal requirements [3].

Security is another issue that must not be ignored especially when exposing the embedded systems directly on the Internet or inside the enterprise networks. Proven solutions like SSL/TLS (Secure Socket Layer/Transport Layer Security) for transport security or even recently issued standard for Web service security (WS-Security) require a lot of processing power and may disrupt other tasks of embedded device.

This paper presents a test-bed performance evaluation of web services implementation on ARM-based embedded system, built with the gSOAP toolkit [4]. Several scenarios are used in the evaluations: Apache CGI-based Web services – standard and HTTPS; standalone Web services – no security, SSL/TLS, and WS-Security.

2. BACKGROUND AND RELATED WORK

2.1. Service oriented architecture and Web services

Service oriented architecture (SOA) is a paradigm aimed at organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a framework to build an autonomous and interoperable systems. It is particularly significant for the domain of embedded systems, where the usage of a high-level service-based communications infrastructure opens entirely new perspectives. There are several ongoing projects [2, 6] aimed at proposing a variation of SOA that takes under consideration the specific characteristics of embedded systems [2, 6].

Web services are the most adopted technology for implementing SOA. They constitute a group of XML-based standards (WSDL – Web Services Description Language, SOAP – Simple Object Access Protocol, and UDDI – Universal Description Discovery and Integration), designed for the communication of loosely coupled, heterogeneous systems. Web Services allow application to application communication. The SOAP protocol is used for message based and remote procedure call (RPC) communications. It also defines the serialization of function's parameters in a completely system independent format and allow the exchange of every data that could be serialized. WSDL is used to describe the services in a standard way. Most existing toolkits have both tools to generate WSDL file from the sources and to generate the skeleton code from a given WSDL file.

2.2. Multi-tier architectures

Multi-tier architectures provide many benefits over traditional client/server architectures. A four tier model for building distributed automation systems is presented in [8]. It consists of client, presentation, service, and automation tiers. The automation tier is represented by embedded devices with a set of sensors and actuators. To increase integration and to further abstract the communication of automation tier with other tiers the functions of the embedded devices can be provided as a set of Web services [8].

2.3. Web service generation toolkits – gSOAP

There is a long list of Web services generation toolkits available for use. Some of the most popular are Apache Axis [11] (C/C++ and Java version), kSOAP (J2ME), bSOAP (Grid applications), eSOAP, .NET Compact Framework, gSOAP [4], and etc [5].

While AXIS is dominating in the desktop domain, gSOAP is one of the best choices for the domain of embedded systems and the reasons for that are lying in its design characteristics [3]:

- Performance enhancing strategies – uses XML predictive pull parsing technique for efficient XML serialization. Avoids the overhead of message exchanges through multiple protocol layer APIs;

- Static proxy generation – reduces memory requirements and run-time processing overhead;
- Scalability – achieved with the use of linear-time XML serialization algorithms;
- Limit memory usage – dynamically allocating data only when necessary;
- Support for pure "C" code – essential for many embedded systems kernels and system-oriented applications developed in "C".
- The RPC compiler used generates compact code with small memory footprint;
- Full support for the basic set of Web services protocols with provisions for building stand-alone (embedded) HTTP/HTTPS Web services.

The gSOAP toolkit is platform independent and includes a WSDL parser *wsdl2h* and skeleton generating compiler *soapcpp2*. The only platform dependent module is the run-time library *stdsoap2*. The development process begins with the creation of a service header file based on the WSDL file. Next the gSOAP compiler is used to populate needed code files. At run-time RPC calls are made on client side proxies [4].

2.4. Related work

In [7] the author introduces the idea of remote services for embedded systems. The work focuses on three aspects: distribution, minimum embedded device modification, and component-based development. Disadvantage of the proposed solutions is the use of a special Service Description Markup Language (SDML) that could be substituted by standardized SOAP and WSDL.

The authors of [10] in their work describe a test case scenario with gas chromatograph using both embedded web server with CGI application and embedded SOAP. The results show that applying SOAP services on embedded devices is not only feasible, but desired because it will improve interoperability.

Schall et. al. [9] in their work are measuring round-trip delays to estimate the performance of gSOAP and kSOAP implementations of the Google Web service API. The results show that the gSOAP developed Web services performs faster and with smaller deviations than the those developed with kSOAP.

3. EXPERIMENTAL RESULTS

3.1. Test-bed architecture and scenarios

The test-bed architecture consists of two PCs (Web services clients), embedded device (Web services server), and microcontroller with temperature sensor. One of the PCs (linux client) is running OS *Debian Linux 2.6.18-4-686* and the other one (windows client) – *Windows XP Servicepack 2*. They both are running on a machine with Intel(R) Pentium(R) 4 CPU 3.00GHz and 1GB of RAM. The Web services client on the linux client is developed with gSOAP toolkit and on the windows client – developed with *.NET Compact Framework*.

The embedded device is development board from Olimex [12] with ARM9-based 200MHz processor, 32 MB external SDRAM, 2GB USB flash (OS runs from it), Fast Ethernet port, running *Linux 2.6* OS. The Web services are developed with *gSOAP*

v.2.7.3 toolkit for *Debian Linux*. Packet capturer used in experiments is *Wireshark v.0.99.4*. Test-bed architecture is shown on figure 1.

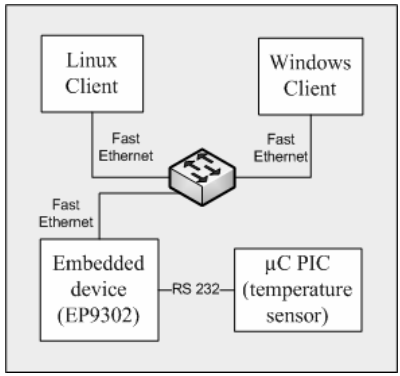

Figure 1: Test-bed architecture.

Two sample Web services are developed for the experiments: *Echo services* – accepts a string and reply with that string and *Temperature service* that obtain and return real-time temperature. Five different scenarios are used: Web services running as standalone applications – with no security provisions, with SSL/TLS, and with WS-Security; Web services running on embedded apache server as CGI applications both HTTP and HTTPS.

3.2. Results

A series of experiments are carried out 101 times for each scenario and the average delay and jitter are calculated. The deviation of the delay and the jitter from its mean value is determined (see Table.1 – the values are in seconds and rounded).

Table 1: Minimal, average, and deviation values for delays and jitter.

		Standalone (no security)		Standalone SSL/TLS		Standalone WS-Security		Apache (HTTP)		Apache (HTTPS)	
		Echo	Temp	Echo	Temp	Echo	Temp	Echo	Temp	Echo	Temp
delay	min	0,01	3,89	0,43	3,90	0,35	4,23	0,07	3,94	0,11	3,98
	average	0,01	3,89	0,44	3,91	0,37	4,27	0,14	3,97	0,22	4,03
	deviation	0,79	1,72	3,21	1,83	3,06	5,08	3,70	2,41	4,92	9,59
jitter	min	-0,01	-0,01	-0,02	-0,01	-0,02	-0,02	0,00	0,00	0,00	0,00
	max	0,00	0,02	0,02	0,02	0,03	0,03	0,10	0,20	0,09	0,31
	average	0,00	0,00	0,00	0,00	0,00	0,00	-0,18	-0,09	-0,17	-0,30
	deviation	1,08	2,57	4,25	2,92	4,13	5,58	5,32	3,68	7,24	14,55

The scenario with WS-Security gives around 90% of message overhead from the security header included in the SOAP message. The tests are made only for short exchanged values like measured temperature. For longer exchanged values the overhead should be lower because the security header should not increase significantly.

The delay is measured at the Packet level and is calculated as the time between the first SYN packet sent from the client and the last ACK received from the server. Application level measurements are also made, but for .NET clients only and are not included in the comparisons. The Figure 2 gives a notion of delay introduced by different scenarios.

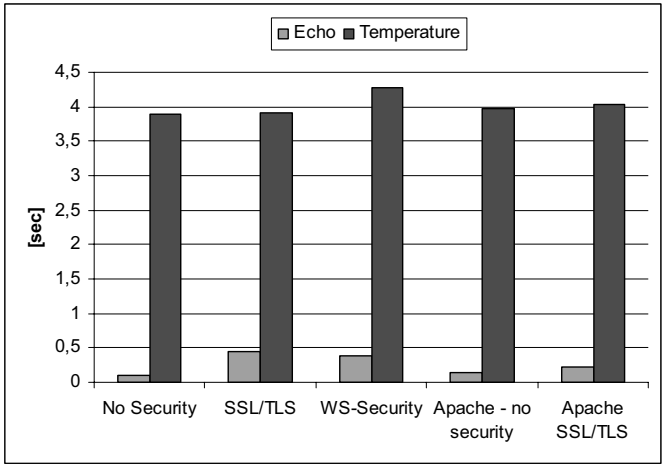

Figure 2: Average delays for two sample web services and all five scenarios.

The Apache-based SSL/TLS scenario outperform standalone scenario for the *Echo* service, but perform worse for the *Temperature* service. The reason for that could be in the way Apache handles serial port communication and that the standalone version does not include any special optimization. As could be expected the WS-Security performs worst. For the *Temperature* services it has the highest delay and for the *Echo* service is almost the same as for the standalone SSL/TLS.

4. CONCLUSIONS AND FUTURE WORK

The results from the experiments show that running Web services on embedded devices is feasible. Remark has to be made that the experiments included no optimization strategy and the Web service generation toolkit was used as it is available in the public domain. This gives possibility to further enhancements especially if commercial solutions are used.

Introducing state of the art Internet technologies in the embedded systems domain will increase interoperability between devices from different manufacturers and will allow much deeper integration within the existing enterprise systems. However, giving remote access to embedded devices must involve securing measures. The results show that including security increases the delay, jitter and their deviation values in times and so require additional optimizations.

The future work include thorough experiments involving different Web services generation toolkits, different embedded devices and new variants of Web services

that will endorse more precise results.

5. ACKNOWLEDGEMENT

The work in this paper is supported by National Science Fund of Bulgaria, projects – "BУ-966/2005" and "MU-MI-1602/2006".

6. REFERENCES

[1] Borriello, G., R. Want, "Embedded Computation Meets the World Wide Web", Communications of ACM, Vol. 43 №5, pp. 59-66, May 2000.

[2] Deugd S., R. Carroll, K. Kelly, B. Millett, and J. Ricker, "SODA: Service-Oriented Device Architecture", IEEE Pervasive Computing, vol. 5, no. 3, 2006, pp. 94-C3.

[3] Engelen, R. van, "Code Generation Techniques for Developing Light-weight XML Web Services for Embedded Devices", ACM SAC'04, March 14-17, 2004, Nicosia, Cyprus, pp. 854-861, ISBN:1-58113-812-1.

[4] Engelen, R. and K.Gallivan, "The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks", Proc of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), pages 128-135, May 21-24, 2002, Berlin, Germany.

[5] Govindaraju, M., A. Slominski, K. Chiu, P. Liu, R. Engelen, M. Lewis, "Toward characterizing the performance of SOAP toolkits", Proceedings. Fifth IEEE/ACM International Workshop on 8 Nov. 2004 Page(s):365 – 372.

[6] Jammes, F., H. Smit. "Service-Oriented Paradigms in Industrial Automation, Industrial Informatics", IEEE Transactions on Volume 1, Issue 1, Feb. 2005 pp. 62 – 70.

[7] Jazdi, N., Component-based and Distributed Web Application for Embedded Systems, International Conference on Intelligent Agents, Web Technology and Internet Commerce, 9-11 July 2001, Las Vegas, USA.

[8] Kakanakov, N., M. Shopov, I. Stankov, and G. Spasov, "Web Services and Data Integration in Distributed Embedded Systems in Internet Environment", International Review on Computers and Software (IRECOS), vol. 1, no. 3, November 2006, pp.194-201, ISSN: 1828-6003.

[9] Schall D., M. Aiello, S. Dustdar, "Web Services on Embedded Devices", International Journal of Web Information Systems (IJWIS), 2006.

[10] Topp, U., P. Müller, "Web based service for embedded devices", Lecture Notes in Computer Science, Volume 2593 / 2003, pp. 141 – 153, ISSN: 0302-9743.

[11] Apache Axis project's website – http://ws.apache.org/axis/

[12] http://www.olimex.com/dev/cs-e930x.html