WEB SERVICES DEPLOYMENT METHODOLOGIES FOR WEAK PROCESSING DEVICES – AN ANALYSIS

DENIS TRČEK
Department of Communication Systems – E6
»Jožef Stefan« Institute
Jamova 39, 1000 Ljubljana
SLOVENIA

Abstract: Grids started as an approach to scientific computing and they are now gaining importance also in commercial sector thanks to web services. These are a fine way to implement services oriented architectures that enable organizations to expose their profiled and specialized applications so others can use them in different application scenarios. However, exposing services that used to be of internal nature, immediately brings security to the front line. In addition, it is a fact that many proprietary solutions already exist and that it is infeasible to completely rewrite them to meet web services specifications. Moreover, some of these services can not be exposed at all due to computational constraints. Thus this paper presents an analysis of possibilities to overcome the above problems for a wider penetration of web services into business environments.

Keywords: decentralized systems, wireless applications, grids, web services, weak processing devices.

1. Introduction

Web services (WS) present a new kind of implementation of an old paradigm about distributed computing. Older approaches include various technologies [1, 2]:

- classical client – server applications that the whole internet is based upon;
- Common Object Request Broker Architecture or CORBA, which is again a client – server paradigm, but tailored down to objects level;
- Java Remote Method Invocation or RMI, which is a competing technology to CORBA.

Another computing concept that is important in relation to WS is the concept of grids. According to [2], *grids enable the virtualization of distributed computing and data resources such as processing, network bandwidth and storage capacity to create a single system image, granting users and applications seamless access to vast IT capabilities*. Put another way, grids enable users to see computing infrastructure as a unified instance of computing resources, just as a plug in wall enables users to see the electricity system as a single power plant. This enables organizations to spread the workload to achieve optimizations, i.e. balancing, then to achieve redundancy for handling failures of certain sub-systems, better exploitation of resources and tighter collaboration, even with geographically dispersed resources. Shortly, all resources that form a grid function as a single large computer. And they can be, in principle, deployed as if they were indeed a single large computer.

2. Grid and WS Standards Overview

A variety of standards exist in the grid area. One of the first organizations to start the grid paradigm was GLOBUS that produced specifications for Open Grid Services Architecture or OGSA [4]. While GLOBUS can be seen as an initiative of academic origin, there exists another grid initiative that is market oriented. This is Enterprise Grid Alliance (EGA) with specifications that are focused on deployment in commercial environments [5]. These two models can be seen as complementary.

Grids can be seen as a concept, with one concrete implementation behind this concept being WS. To fulfill this role, WS must provide the ability to access and manipulate states, i.e. the ability to find and interact with a stateful resource in a standardized manner (the WS-Resource framework provides the necessary definitions [6]).

Further, all these processes have to be coordinated appropriately to complete a high level task. Such coordination is achieved mostly by orchestration. In orchestration, one process (controller) takes control over other needed web services and coordinates their execution. The web services involved are not "aware" of the fact that they are part of some higher scenario – but the controller is certainly "aware" of this. And for this purpose, the controller needs explicit definitions of operations and their order. To support orchestration, a special language has been introduced, called Business Process Execution Language or BPEL4WS [7].

The bottom line is that WS are defined with three basic specifications: Simple Object Access Protocol, or SOAP [8], Web Services Description Language or WSDL [9], and Universal Description, Discovery and Integration or UDDI [10]. SOAP is a messaging protocol for exchange of information between service requester and service provider, WSDL is an XML based description of WS services and inputs and outputs. Finally, UDDI is a mechanism that supports registration of a service and finding of this service by interested parties.

3. WS Implementation Environments

The wide variety of computing systems in current networks can be structured into the following categories (according to available computing resources from most powerful to the weakest ones):

- mainframes,
- desktop and laptop computers,
- palmtop computers,
- mobile phones,
- smart-cards,
- RFID devices.

With regard to mainframe computers and implementation of WS, no special discussion is needed. Implementation of WS is a straightforward task. With regard to desktop and laptop computers - their resources are comparable and belong to the same orders of magnitude. For both kinds of systems, implementation of WS is a straightforward task. With regard to palmtops – their physical dimensions are already severely constrained, which is reflected in their computing potential. The situation with mobile phones is even tougher. With regard to mobile phones and smart cards one might quibble that these two categories are artificial and that smart cards belong to the same range of available computing power. This is not the case – smart-card controllers are indeed included in mobile phones, but mobile phones, in addition, possess additional flash, etc. With regard to RFIDs there is no doubt – these are the weakest computing devices. In the table below typical quantified key characteristics of contemporary network devices are given.

	desk-top	palm-top	mobile phone	smart-card	RFID circuit
processor speed	3 GHz	0.3 GHz	0.3 GHz	7.5 MHz	20 kHz
RAM	2 GB	64 MB	18 MB	4 KB	few K gates
permanent storage	320 GB	128 MB	50 MB	96 KB	2 KB
network	1 Gbps	54 Mbps	54 Mbps	115.4 kbps	100 kbps
autonomy	full	few hours	few hours	none	none

Table 1: Categories of networking devices with their typical system resources

In the above table, permanent storage includes hard disks and FLASH memory without expansion cards. Further, smart-cards are assumed to be processor cards and together with RFIDs, they are assumed not to be battery powered. And finally, RFIDs do not have processors as such – the speed is just the speed of the clock that controls the RFIDs' gates operations.

4. Available Approaches to WS Implementation

In order to implement web services for the above variety of computing devices, the following methodological approaches can be taken:

- Full-blown WS implementation, where a service is built from scratch according to complete WS specifications.
- Lightweight WS implementation, where only certain (core parts) parts are implemented or where existing protocols are optimized into more compact forms or implementations.
- Wrapped WS implementation, where core service is left as-is, but a front-end is developed. Front-end takes care of WS operations on behalf of the service, and provides it in a WS manner to the outer world.
- No direct WS support, meaning that the service cannot be made available as such, or that it has to be completely rewritten.

The problem with the second option, of course, is that in the majority of cases this approach clashes with existing standards. However, it can also lead to their future improvement. Because we wish to conform to existing standards within this paper, wrapped WS implementation will be analyzed.

For our analysis we will assume one very basic WS that provides just the device's own unique identification. Further, we will focus on the most extreme cases, which are smart-cards and RFIDs. An RFID system is given in Fig. 1.

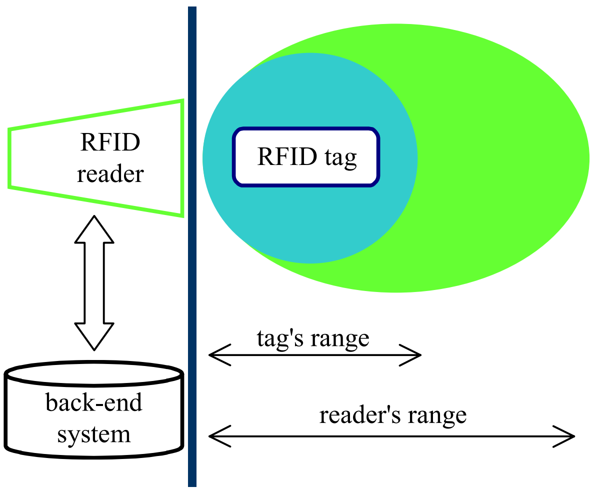

Figure 1: RFID system

For the purpose of our analysis we built an identification WS on a desktop computer. Our testing environment was built by using the following packages (the last three packages were used in case of certain compatibility problems):

- Tomcat 4.x that is a HTTPD engine with servlets support (also called servlet container).
- Java XML Pack Specification Interface Classes 02 / update 1, which includes various APIs:
  - JAXP for XML processing (validation of documents);
  - JAXM for XML based messaging for Java applications to send and receive a document using a pure Java API (it implements Simple Object Access Protocol v 1.1 with attachments);
  - JAX-RPC for XML-based RPC messaging that allows Java application to invoke a Java based WS (this can be seen as Java RMIs over Web Services);
  - JAXR for XML Registries that provides a uniform and standard Java API for accessing metadata registries like ebXML and UDDI;
  - JAXB for XML Binding that allows Java developers to create and edit XML using familiar Java objects.
- SOAP toolkit (apache.org implementation version 2.3), which provides a client library to invoke SOAP services available elsewhere and server-side tools to implement SOAP accessible services. It supports HTTP almost exclusively – other options like SMTP have limited support.
- Apache Xerces 2.9 namespaces-aware XML parser for validation of XML documents.
- JAXP 1.3 JAXP, Java API for generation, parsing and manipulation of XML documents.

The above versions of solutions already belong to the older implementations (versions), but they are still solid options for production use. The reason why they were chosen was that they require significantly fewer resources than the current, latest versions. The PC was a typical system with Intel P4 3GHz and 512 MB RAM.

The resources requirements for deployment of the above implementations - as we found out by our experiment - were as follows:

- The total permanent (hard disc) storage requirements were 73.5 MB (web-services packages); of these 39.8 MB were for Tomcat, servlets supporting HTTP daemon, 35.6 MB for Java RE, and 2 KB for the ID class file together with deployment descriptor file. The total hard disc usage was approx. 150 MB.
- RAM usage was up to 10.46 MB for Tomcat. SOAP (rpcrouter) and ID servlet used up to 7.6MB of RAM. These were all Java processes. They depended on 37 system modules that used approx. 8 MB of RAM. The total usage of RAM was over 26 MB.
- Use of CPU resources - execution of our testing ID web service increased CPU operations up to 52%. It should be emphasized that this percentage serves only for rough orientation, because real CPU load is a matter of duration of processing (process priority). So, in principle, if one assumes the same conditions except that a processor is a few times slower (lower clock frequency), the processing would just take a few times as much time as in this case.
- With regard to network utilization, this turned out to be the least problematic segment. The whole outbound and inbound SOAP message required approx. 2*800 B and was transferred with an average rate of 1500 bps.

From the above measurements (and taking into account facts from Table 1), it is evident that full-blown WS can be currently implemented on mainframes, desktops, palm tops, and conditionally in certain mobile phones. But for the rest of devices (smart-cards and RFIDs) the above analysis implies that wrapping by back-end systems is required. The concrete place can, in principle, be even a reader. Final remark - although it is a matter of a few years to have an implementation of a web server on a smart-card, full-blown WS are another bunch of years away. This is also proved by current implementation of Java 2ME Wireless Toolkit.

It is interesting to mention one lightweight option, which is called PocketSOAP [11]. As the name implies it is intended for palmtops with MS Windows OS. These binaries (provided as DLLs) require 0.7 MB and include HTTPD and SOAP. This implementation eases implementation for palmtops, and further supports our findings. For mobile phones, which currently constitute the border where implementations of WS are feasible, wireless markup language and other related specifications from Open Mobile Alliance can present a significant gain [12].

5. Conclusions

Services oriented architectures that are based on WS are a promising approach for distributed computing in business environments. But current implementations of WS are limited to environments with sufficient resources in terms of computing power, available RAM, permanent memory, and available bandwidth. However, mobile and various handheld devices, together with RFID systems, are becoming integral parts of global networks. Moreover, the number of such devices in global networks is increasing at a much faster pace than that of traditional devices. And implementing WS in such cases is not a straightforward task.

We have presented an analysis of typical properties of main types of network devices: mainframes, desktops, laptops, palmtops, mobile phones, smart cards and RFIDs. We also developed a basic WS solution, which provided only identification of a device or service. We made basic measurements about the use of resources of this application.

On this basis we found that even such basic WS is too demanding in terms of required assets and resources for RFIDs, smart cards, and even the majority of mobile phones. Thus we have proposed bypass solutions, which are lightweighting of existing protocols and use of wrappers. As the first approach is a matter of a long-term standardization process, we believe that the other architectural approach, which is wrapped-WS, is already feasible today. Thus further efforts about deployment of these weakest devices in WS scenarios should be concentrated on wrapped-WS approaches.

References:

[1] Nagappan R., Skoczylas R., Sriganesh P.R., *Developing Java Web Services*, John Wiley & Sons, Indianapolis, 2003.

[2] Chase N., *XML Primer Plus, SAMS Publishing*, Indianapolis 2002.

[3] IBM, What is grid computing?, http://www-1.ibm.com/grid/about_grid/what_is.shtml.

[4] H. Kishimoto H., Treadwell J., *Defining the Grid: A Roadmap for OGSA™ Standards*, Open Grid Services Architecture WG Draft GGF-OGSA-Roadmap-024, September 2005, http://forge.gridforum.org/sf/go/doc13521?nav=1.

[5] Enterprise Grid Alliance, *Reference Model and Use Cases, parts 1 and 2*, Reference Model WG standard v 1.5, March 2006.

[6] Maguire T., Snelling D., Banks T., *Web Services Service Group*, WS ServiceGroup Committee Specification wsrf-ws_service_group-1.2-spec-cs-01 V 1.2, 2006.

[7] Andrews T., Andrews T., Curbera F., Dholakia H, Goland Y., Klein J., Leymann F., Liu K., Roller D., Smith D., Thatte S., Trickovic I, Weerawarana S, *Business Process Execution Language for Web Services version 1.1*, 2003, http://dev2dev.bea.com/technologies/webservices/BPEL4WS.jsp.

[8] Mitra N. (ed.), *SOAP Version 1.2, Primer*, W3C Recom. REC-soap12-part0-20030624, 2003, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

[9] Christensen E. et al., *Web Services Description Language (WSDL) v 1.1*, W3C Rec. NOTE-wsdl-20010315, 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[10] Clement L. et al. (eds), *UDDI Version 3.0.2*, UDDI TC Spec 20041019, http://www.oasis-open.org/specs/index.php#uddiv3.0.2.

[11] Fell S., *PocektSOAP*, Program documentation, http://www.pocketsoap.com/.

[12] Open Mobile Alliance, *WAP Forum Releases*, 2006, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html.