Adaptation of Web service architecture in distributed embedded systems

Nikolay Kakanakov, Grisha Spasov

Abstract: *The paper discusses the possibility of adaptation of Web Services Architecture (WSA) in distributed embedded systems. It provides a brief description of this architecture and a tool for creating services in Java and C/C++. A three-tier client/server model of a distributed system for monitoring of temperature and humidity is presented and techniques for adaptation of this model to the WSA are proposed. The adaptation is made with Apache server and eSOAP toolkit. A comparison of efficiency and ease of integration between three-tier client/server model and services-oriented model is made.*

Key words: *Distributed Embedded Systems, Web Service Integration, Client/Server Model, Remote Monitoring and Control.*

INTRODUCTION

Computer science is currently built on a foundation that largely assumes the existence of a perfect infrastructure. That is why even a single failure can ruin the whole systems. Among these complex systems are Distributed Embedded Systems (or Networked Embedded Systems - NEST), composed of many different nodes in need to communicate with each other. The tasks for these systems are dynamically assignable, and require reliable and predictable performance even though the nodes are individually unreliable. This approach is now feasible because Moore's law continues to reduce cost of computation and memory, new technologies are present for reducing size of the controllers, and the adaptation of well-known business standards in embedded systems is successful. Coordination, control and programming of embedded systems is currently an unsolved problem [2, 6].

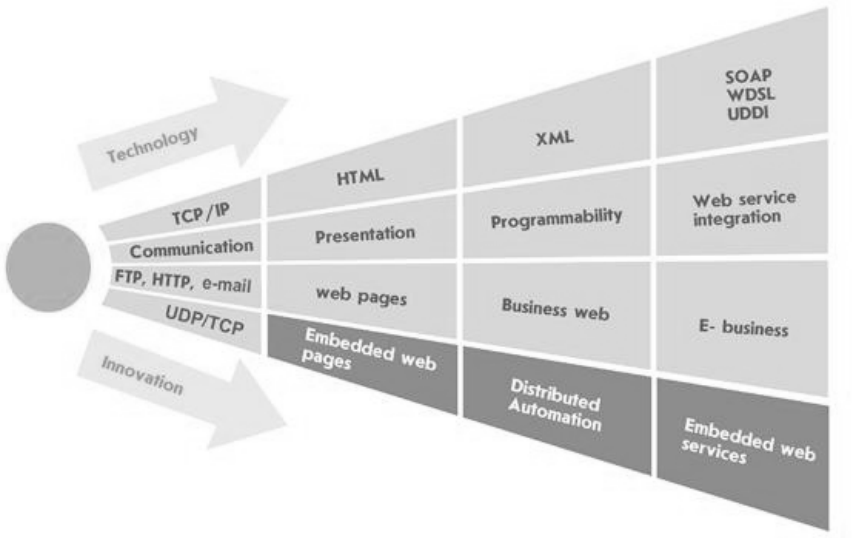

Figure 1. Technology and Innovation in Embedded Systems.

The technology and innovation in embedded systems now follows the line of progress of distributed business systems as shown on Figure 1. The Web Services Architecture (WSA) provides this reliability, scalability and flexibility needed for distributed embedded systems.

The experiments in this paper are part of National Science Fund of Bulgaria project – **"ВУ-906"**, 2005, entitled "Web Services and Data Integration in Distributed Automation and Information Systems in Internet Environment".

Introducing Web Services Architecture (WSA)

The WSA combines the best aspects of component-based development and the World Wide Web. According to [1] a service is: "a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."

Applications access web services via common Web protocols and data formats. The most common used protocol for transferring data in the Internet is HTTP (Hypertext Transfer Protocol) and it is the key transport protocol in the WSA. The universal schema in the Internet that codes the data is XML (Extensible Mark-up Language). The current structure of Internet is based on program-to-user interaction and the WSA is based on program-to-program interaction [7, 8].

The WSA is based on some key standards: XML for data representation; SOAP for accessing services; WSDL for describing services; UDDI – registering and discovery of services [1, 7].

SOAP (Simple Object Access Protocol) is XML-based, lightweight protocol for data exchange in a decentralized, distributed environment. SOAP request packets call methods available on the SOAP server, and the server sends a response packet, structured in a similar manner [1, 7].

WSDL (Web Services Description Language) is a XML grammar for specifying properties of a Web Service, such as what it does, where it is located, and how to invoke methods on a particular service. WSDL is an interface definition language similar to those of CORBA and DCOM [1, 7].

UDDI (Universal Description, Discovery and Integration) is a group of web-based registries that expose information about a business or other entity and its technical interfaces (or API's). By accessing any of the public UDDI Operator Sites, anyone can search for information about web services that are made available by or on behalf of a business. The information that a business can register includes several kinds of simple data that help others determine the answers to the questions "who, what, where and how" [1, 7, 8].

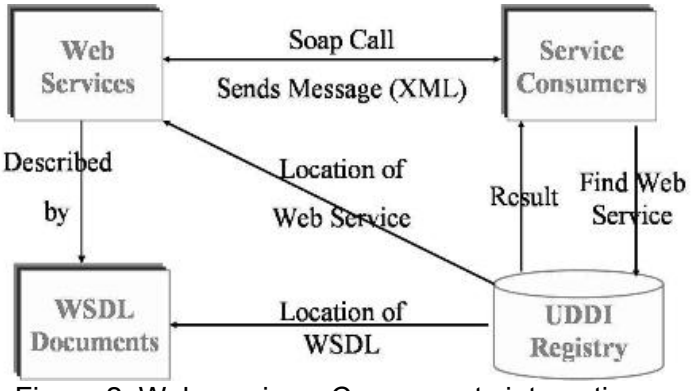

Figure 2. Web services. Components interaction.

The scenario of the interaction between the consumer and provider of the service is shown on Figure 2. The consumer contacts the UDDI registry and finds the service description (in WSDL). Using WSDL document consumer finds the location of the service and "binds" to it. Then, using the interface from WSDL document, consumer generates SOAP calls and receives SOAP responses [1, 7].

The complexity and verbosity of XML Web services protocols creates a whole new set of design tradeoffs and issues for developing Web services applications for embedded systems. Embedded systems rarely have enough memory and processing power to run a Web services. On the other hand, current Web services implementations do not

adequately apply to the embedded processing [4, 5].

For solving this problem there are several possible techniques. One is the generation of "light-weight" web services [5]. Another is to adapt WSA to the three-tier architecture as it is described in the current paper.

Apache and Web services

Apache is one of the most popular web servers in the Internet. Apache is open source, free of charge to use web server, which has modifications for most popular operating systems – MacOS, Windows, Linux, UNIX, etc. It is built on a modular basis and this makes it very scalable. You can add modules to support Java, Perl, and PHP, Python and even web services and XML [11].

There are several different modules for Apache to add web services capabilities to it. One possible toolkit to add web services capabilities to Apache web server is Axis. The Apache Axis is an implementation of the SOAP W3C protocol for Java and C/C++ [11].

Another module is available from EXOR International (www.exorinternational.com). This module works with eSOAP toolkit. The module includes a dynamic library for Apache that works as a filter. This filter accepts requests from clients and passes them to the user configured SOAP application. This module is used in realizing the system described in the current paper [10].

The Embedded SOAP Software Component, eSOAP, is a SOAP reference implementation that implements a Web Services interface for embedded devices. The software consists of a C++ library that implements a SOAP engine based on the SOAP 1.1 specification. The memory footprint of the eSOAP engine is about 150KB. The eSOAP core engine is written in 100% ANSI C++ and it is highly portable. It has been designed to use STL (Standard Template Library) when available or an internal container library when STL is not available. Exception handling has been avoided since this feature is usually not available on embedded platforms [10].

The toolkit also provides a compiler for generating C++ stub and proxy classes based on WSDL files and a Java library of interface classes for client development. The Java library includes a SOAP client interface that can be executed from with a web browser. In this way, developers can concentrate on the business logic of their SOAP applications and start coding quickly [10].

Three-tier client/server model with Java transaction server and Apache

A model of distributed system for monitoring of temperature and humidity based on three-tier client/server model is shown on Figure 3. On the data-tier there is a network of embedded controllers. Every controller measures temperature and humidity in a particular point. On the client-tier there is a PC with a connection to Internet and a web browser with Java Applet capabilities. On the middle tier there is a Linux-based application server. The application server consists of two parts. First part is Java program that polls the data from the controller network and organize this data for statistics. Second part is an Apache web server that provides measurement results and statistics in html documents [9].

The key benefits of this model include [9]:

- Security mechanism is concentrated in the middle tier. It is based on the security of the Apache server. There is no need to adapt memory and calculation consuming encryption algorithm in the embedded controllers.
- Data storage and organization of statistics is provided by the middle tier, which in most cases has much more memory than the embedded controllers.
- Middle tier can organize data and services from big number of controllers in the network and provide them as a single HTML document in response to the user.
- Middle tier can provide some reaction mechanism to the controller network, based on the measured data.

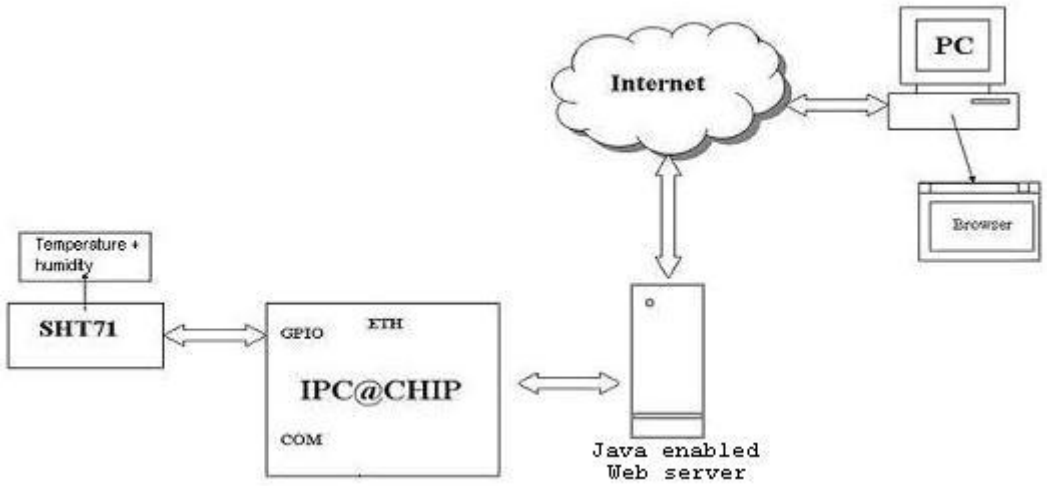

Figure 3. Three-tier model of distributed embedded system [9].

The described three-tier distributed system for monitoring of temperature and humidity is developed and currently works in a laboratory in Technical University – branch Plovdiv. It can be viewed on the following web address: http://temperature.tu-plovdiv.bg/.

Adaptation of WSA to the system for monitoring of temperature and humidity

To adapt an embedded system to the WSA it is needed to add XML parser and SOAP engine to it (assuming that there are network capabilities and Web server). These two programs are large (about 200KB) for a small system with just few hundreds of kilobytes of memory. Additionally encapsulation of SOAP envelope and parsing the XML messages is very time-consuming for embedded processors. There are some attempts for optimization of XML parsers and SOAP engines [4, 5].

The optimization techniques are not always the right decision. Sometimes the goal is re-using of the existing embedded hardware and software. This is possible by using server that connects the network of controllers to the outside Internet.

In the three-tier model the application logic and security mechanism are provided by the middle tier. In WSA application logic is broken into small units named services. In that case it is straightforward to add web services capabilities exactly to the middle tier [3].

To a particular web service the middle-tier can be considered as a proxy object – Figure 4. This object parses XML messages, encapsulates SOAP envelope and provides endpoint for the service. It can be stand-alone server or integrated into existing Web server [3].

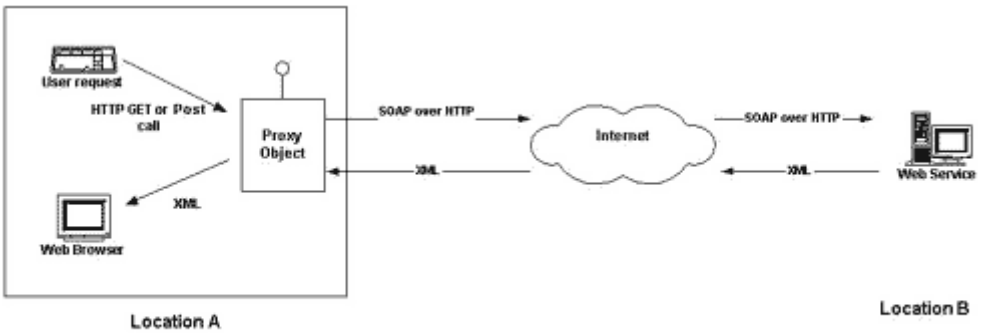

Figure 4. Providing web services through Proxy Object.

The architecture is still three-tier and the middle tier is a proxy for the embedded controllers to provide web services. The middle tier runs Apache web server which provides HTTP protocol for communication and provides direct way for the client to access

the services. This is made by HTML document with Applets integrated inside.

The structure of the proposed architecture is shown on Figure 5. The current paper focuses only on the middle tier and experiments are not made via Internet cloud.

The connection between middle tier and controllers is currently based on sockets but it can be based on RPC calls as well.

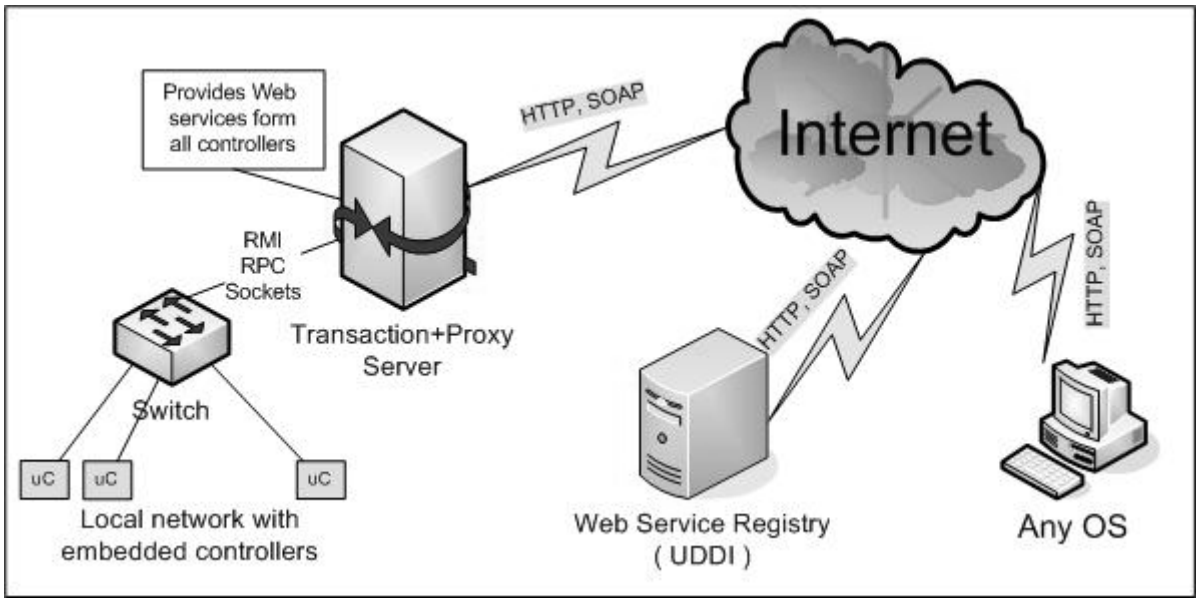

Figure 5. Adaptation of WSA in distributed embedded systems.

The brief observation of the network traffic shows no visible difference between the two systems. They both transfer packets with data part bellow 1KB in size. In that case there is no difference in packet size distribution and network load. There is a flow of small packets in short time intervals. The SOAP uses HTTP 1.1 which supports persistent connections, so in the sequence of SOAP requests/responses there is only one TCP connection establishment (three-way handshake).

The request-response times of Java-enabled system is about (9-11)ms and that of SOAP-enabled system is about (16-18)ms. These times are calculated for the case of a small LAN with just a few nodes, the Nagle algorithm and turned off delayed Acknowledge of TCP.

In both cases an Applet client is used. The "AppletTest.class" uses SOAP and the "my_test.class" uses Java-enabled server. Both Applets are almost the same size, but in the SOAP-enabled system there is one-time download of "esoap_core.jar" file which is about 58KB – Figure 6.

```
192.168.0.8 - - [02/May/2005:13:53:35 +0300] "GET /test/ HTTP/1.1" 200 922
192.168.0.8 - - [02/May/2005:13:53:40 +0300] "GET /test/my_test.class HTTP/1.1" 200 3419
192.168.0.8 - - [02/May/2005:13:53:40 +0300] "GET /test/esoap_core.jar HTTP/1.1" 200 59234
192.168.0.8 - - [02/May/2005:13:53:40 +0300] "GET /test/AppletTest.class HTTP/1.1" 200 4543
192.168.0.8 - - [02/May/2005:13:53:53 +0300] "POST /rpcrouter HTTP/1.0" 200 546
```

Figure 6. Extraction from web server access log.

The last row shows the actual request for the web service with the POST method. The destination "/rpcrouter" refers to the endpoint of the web service.

The key differences are repositioned in the middle tier. In both cases there is Apache installed. Java-enabled server uses about 50KB of RAM during work and this can increase when there are many simultaneous connections. The SOAP-enabled system needs a "mod_esoap.so" module to run with Apache. It is about 20KB in size. Service implementation module is about 300K ("esoapserver.so") in size and is placed in memory as a shared object. So, the SOAP realization uses more RAM.

CONCLUSIONS AND FUTURE WORK

The comparison of the two presented models has led to the following conclusions:

- The size of the Java-enabled server is smaller than that of the SOAP-enabled system. But the differences are not significant for a general purpose PC or server.
- Java-enabled system is applicable in specific implementations where small memory footprint and small processor overhead is needed.
- SOAP-enabled system with adding just a little overhead provides all the benefits of the WSA:
  - The system is no more just an application. It is a service which can be easily integrated in bigger systems.
  - It provides public interface in WSDL, so it can be used by clients written in different programming languages (VS.NET, Java, PHP, C/C++ etc.).
  - It is dynamically discoverable through service registry such as UDDI.

The future work of this project can go in different direction. One is the integration of some web service capabilities to the controllers. The other is deeper analysis of the SOAP-enabled three-tier model. This includes long-term analysis of network traffic and web server log; using of proxies and firewall; calculating processor overhead, etc.

REFERENCES

[1] Austin, D., A. Barbir, C. Ferris, S. Garg. Web services architecture requirements, http://www.w3c.org/TR/wsa-reqs.

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

[3] Channabasavaiah, K., K.Holley, M. Edward, Jr. Tuggle, Migrating to a service-oriented architecture, Part 1 (2003), http://www.ibm.com/developerworks/webservices/library/ws-migratesoa/index.html.

[4] Davis, D., M. Parashar. Latency performance of SOAP implementations. In 2nd IEEE International Symposium on Cluster Computing and the Grid, 2002, pp. 407-412, ISBN: 0-7695-1582-7.

[5] Engelen, R., 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.

[6] Estrin, D., G. Borriello, R. Colwell, J. Fiddler, M. Horowitz, W. Kaiser, N. Leveson, B. Liskov, P. Lucas, D. Maher, P. Mankiewich, R. Taylor, J. Waldo, Embedded, Everywhere. A Research Agenda for Networked Systems of Embedded Computers, NAP Washinton, D.C. 2001, ISBN 0-309-07568-8.

[7] Kreger H., "Web Services Conceptual Architecture (WSCA 1.0)", IBM Software Group, May 2001, www.redbooks.ibm.com.

[8] Stoilov T., K. Stoilova, Integration of Web Services in Internet, 18th International Conference "SAER-2004", 24-26 September, St. Konstantin resort, Varna, Bulgaria.

[9] Spasov G., N. Kakanakov, N. Lupanov, Three-tier distributed applications, Computer Science'2004, 6-8 Dec 2004, Technical University Sofia, Bulgaria, pp. 172-177.

[10] Embedded SOAP toolkit, http://www.embedding.net/eSOAP/.

[11] Web Services Project at Apache, http://www.apache.org/.

ABOUT THE AUTHORS

Nikolay Kakanakov, PhD student, Department of Computer Systems and Technologies, Technical University, branch Plovdiv, Phone: +359 32 659758, E-mail: kakanak@tu-plovdiv.bg.

Assoc.Prof. Grisha Spasov, PhD, Department of Computer Systems and Technologies, Technical University. Branch Plovdiv, Phone: +359 32 659576, E-mail: gvs@tu-plovdiv.bg.