DISTRIBUTED MEASUREMENT SYSTEM BASED ON JAVA AND WEB TECHNOLOGIES

M. Shopov, G. Spasov

Technical University of Sofia, branch Plovdiv, Faculty of Electronics and Automation, 61 St. Petersburg Blvd., Plovdiv, {mshopov, gvs}@tu-plovdiv.bg

Abstract. The paper discusses the design and implementation of a system for distributed measurement. An adaptation of some of the well-proven architecture models and their applicability in distributed measurement is presented. The system's implementation presented is based on open and standardized approaches – Java programming language, standard communication interfaces, and Web technologies. The measurement component in the system is represented by a network of web-enabled microcontrollers with temperature and humidity sensors.

Keywords: Distributed measurement, Java technologies, Three-tier model, Web technologies.

INTRODUCTION

Recent years, with the progress in computer networks, more and more people and organizations have access to the global network - Internet and the services it offers. Besides, the advances in information systems have emerged new technologies like e-learning, e-business, e-government in different areas of society (medicine, industry, education, etc.). These new technologies are now being transferred toward the field of distributed measurement and automation [9].

A trend from the recent years is to migrate away from proprietary hardware and software platforms for distributed measurement systems (DMS) in favor of open and standardized approaches. High-level programming languages, object-oriented platforms, Internet technology, and standardized communication interfaces, all influence the development of today's DMS [5, 6]. Additionally, rapidly advancing hardware, provide the market with a plenty of new embedded devices with integrated TCP/IP stack, embedded web server and continuously increasing processing power [1]. This gives the designers of distributed measurement systems the ability to put into practice some of the well-proven architecture models from distributed desktop systems.

This paper describes the design and implementation of a system for distributed measurement that uses popular client-server architectures, Java and Internet technologies. The design is based on three-tier architecture model [2], with the focus put on the middle tier. The implementation of the system is based on Java Server Pages (JSP) technology [11].

MODELS FOR DISTRIBUTED MEASUREMENT

As documented in [9] there are three major models of systems for distributed measurement and automation. These models are derived from some of the well-proven architecture models for business systems. However, they have some specific characteristics that reflect the limited resources available and the increased demand for data actuality. These models are [9]:

- *Client/Server systems, based on custom communication protocol.* Although cheap for manufacturing, these systems are based on proprietary technologies. This limits their freely distribution and makes them hard to extend and difficult to integrate in complex systems. An improvement of such systems is to use Applets or Active-X controls at the client side, thus unifying the user interface.
- *CGI based distributed embedded systems.* Such systems have additional requirement to microcontrollers – Network interface, TCP/IP stack and embedded Web server. This is no more a problem, since embedded systems that fulfill these requirements are widely available on the market today [1].
- *Three-tier Client/Server architecture.* Such systems are derived from the popular three-tier client server architecture – user interface on the front-end tier, business logic on the middle tier, and database on the back-end tier. Here, the back-end tier is replaced with controllers' network.

Fortino, et al. in [3] have proposed patterns for distributed measurements. A Java applet is used at client interface and a Java-based server at the middle tier. Communication with the measurement devices is based on sockets and more abstract RMI mechanism. Shortcomings of this approach are the need for Java-enabled browser that limits the number of supported clients and additional communication channel opened (additional port) that can be blocked by firewalls or NAT services deployed along the path.

In [7] the authors have presented the use of Common Gateway Interface (CGI) for web-based distributed embedded systems. It uses the CGI integrated in the real-time operation systems (RTOS) of the embedded device. Thus, on every client's request CGI process returns the measured temperature and humidity. In [8] the system is expanded to three-tier model with addition of Java-enabled web server.

The authors of [5] in their research have proposed another system for distributed measurements. It uses Java language for better abstraction and code reusability, three-tier architecture and TCP/IP for flexibility and extensibility. Again, a Java applet is used as user interface.

In a closely related work, Jazdi in [4] has proposed a model for adapting Web technologies in industrial automation. In this model, the functions of embedded devices are presented as services on the middle-tier server, called "Remote Service Server". Embedded devices form the data source. The paper suggests the use of component-based design and standard interfaces.

Multi-tier architectures provides many benefits over traditional client/server architectures [2]:

- Installing and deploying the user interface is virtually instantaneous - only the Web interface in the middle tier needs to be updated.
- Using "thin" client interface, it is easier to deploy, maintain, and modify applications - no matter where the client is located.
- Because the application itself is server-based, users always access the most up-to-date version.

These benefits explain the growing popularity of the multi-tier architecture, and Web-based clients.

DISTRIBUTED MEASUREMENT SYSTEM BASED ON JAVA AND WEB TECHNOLOGIES

The design of a modern, web-based distributed measurement systems have to be carried out in accordance with well-proven architecture models, allowing the system to benefit from various available technologies, thus giving it added flexibility and scalability. It has to be highly abstract, easily extensible and user-friendly. These characteristics have motivated the design of the system for distributed measurement, based on three-tier architecture, Java programming language and Web technologies.

The system architecture is shown on figure 1. From the view point of the three-tier architecture, it consists of standard Web browsers located at the client tier that provides an interface to other applications or operators; Web/application server located in the application tier that realize presentation and application functions; and data producer components – controller networks and database servers – located in data tier.

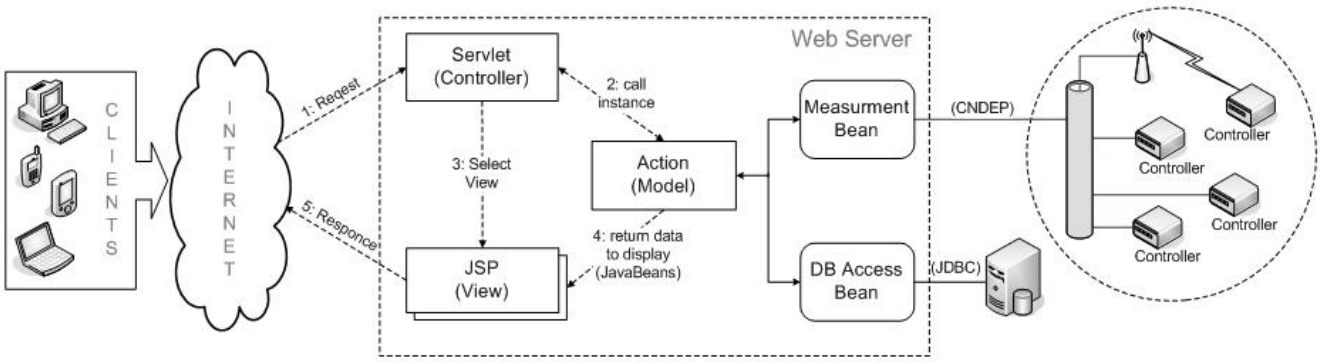

Figure 1. Architecture of a Web-based, three-tier system for distributed measurement.

A client of the system can be any device with a standard internet browser (e.g. PC, Laptop, PDA, and Cellphone – figure 1). Because the interaction with the system is based on exchange of standard HTTP request/response pairs, there is no need for extra plug-ins, thus the client is kept as thin as possible.

Most of the functionality of the system is concentrated at the application tier. It consists of a web/application server (figure 1) that has two key tasks – presentation and application. The popular Model-View-Controller architecture (MVC) [2] is used. The model is a non-visual object that contains all data and behavior and is concerned with business policies and data extraction. In our case, the model functions are extracting measurement data from controllers' network and interactions with databases. Two different objects (Measurement Bean and DB Access Bean – figure 1) are responsible for the two tasks.

The view represents the display of the model in the user interface. In our case, the view is an HTML/WML page rendered with information from the model. It is only responsible for displaying of information; any changes to the information are handled by the controller. The controller takes user input, manipulates the model, and causes the view to update appropriately. In this way user interface is a combination of the view and the controller.

The data tier plays the role of data producer component. It incorporates two types of data sources – controllers' network and database. The controllers' network consists of web-enabled microcontrollers with attached sensors. Web-enabled microcontrollers have an Ethernet adapter, embedded TCP/IP stack, and embedded web server. For extracting of sensors' data various TCP/IP application layer protocols can be used (e.g. CNDEP – Controller Network Extracting Protocol [10]). The controllers' network is a producer of real-time measurement data. On the other hand, a database is used for collecting of logging and statistical information.

Next section describes a sample implementation of a distributed system for measurement of temperature and humidity. This system is implemented in the Virtual laboratory of computer networks and distributed systems [12] and is accessible at http://net-lab.tu-plovdiv.bg/temperature/.

IMPLEMENTATION OF A DISTRIBUTED SYSTEM FOR TEMPERATURE AND HUMIDITY MEASUREMENTS

An application of the architecture from figure 1 has been developed for measurement of temperature and humidity. Most of the implementation issues concern the middle tier. Its implementation is based on Sun Microsystems's Java Server Page (JSP) technology [11].

The JSP technology uses the MVC architecture. Controller functions are handled by a servlet (figure 1). It processes all HTTP requests and determines the appropriate object from the model and appropriate view. The servlet also offers authentication and validation services.

The model consists of two components – Measurement Bean and DB Access Bean (figure 1). The Measurement Bean component is a permanent object for the application. It is instantiated once at the first request received. Its functions are to

contact the remote controllers that measure temperature and humidity and to collect these values in internal variables. This is done periodically in intervals of 10 seconds. This interval is chosen because decrease of the interval will not gain more informativeness to the users. On the other hand, with the decrease of the interval a disruption of the controllers' performance is observed. Traditional sockets and CNDEP protocol [10] are used for connection with the controllers.

The view consists of various JSP pages – for observing of measurement values and for statistical data, for HTML and WML clients and etc. The formation of a HTTP response with temperature and humidity data from the Measurement Java Bean component is shown on figure 2 and the view of the response in the client's browser is shown on Figure 3.

```
<jsp:useBean scope="application" id="Measurment"
             class="temperature.MeasurmentBean"/>
...
<table>
  ...
  <TR align=\"center\">
    <TD><jsp:getProperty name="Measurment" property="tempTini"/></TD>
    <TD><jsp:getProperty name="Measurment" property="humTini" /></TD>
  </TR>
  ...
  <TR align=\"center\">
    <TD><jsp:getProperty name="Measurment" property="tempIpc"/></TD>
    <TD><jsp:getProperty name="Measurment" property="humIpc"/></TD>
  </TR>
  ...
```

DS TINI	
Temperature	Humidity
21.98	40.12
Status	OK
Location	lab 2104

IPC@Chip	
Temperature	Humidity
20.42	37.46
Status	OK
Location	lab 2104

Sat Apr 29 17:36:43 EEST 2006

Figure 2. Formation of the HTML response. **Figure 3. User Interface**

A JSP page may contains all standard elements of an HTML page (HTML tags, content text, etc.) among with some Java code incorporated as action tags and scriplets. Figure 2 exposes the use of the *jsp* tag library for extracting dynamic data from a Java bean. The *jsp:useBean* directive is used to initialize the Measurement bean. The scope *application* means that the bean will be shared for every Servlet and JSP in the web application, thus the same bean will be used every time. The directives *jsp:getProperty* are used for extracting of temperature and humidity values from the Measurement bean.

For the Web server an Apache Tomcat 5.5.14 servlet container from Apache Software Foundation is used. It is run on a 335MHz Pentium II machine, with 256 MB operational memory, and OS Debian Linux 2.4.27-2-386. The system is tested and it proves to operate correctly with PC, PDA, and Cellphone clients.

CONCLUSIONS AND FUTURE WORK

The presented system shows the possibility for adaptation of open and standardized approaches, well-proven architectures, high level abstraction, and standard system protocols in distributed measurements. The use of off-the-shelf solutions, like internet browsers and web servers, brings various advantages and gives the system added flexibility and scalability.

Web servers have a build-in support to high loads, and authentication mechanisms. They are well tested and supported and various development tools exist. The standard user interface used – web browser, allows access to the system from various clients.

Some possibilities for future work include adaptation of web service architecture and employing rich client applications. Web services can be adopted by the controllers offering services like measurement of temperature and humidity. The web server will use these services basing communication on SOAP/HTTP. Rich client applications can be employed for clients having enough resources.

ACKNOWLEDGEMENTS

The presented work is supported by National Science Fund of Bulgaria project – "**ВУ-966/2005**", entitled "Web Services and Data Integration in Distributed Automation and Information Systems in Internet Environment", under the contract "**ВУ-МИ-108/2005**".

REFERENCES

[1] Borriello, G., R. Want. Embedded computation meets the world wide web. Communications of ACM, Vol. 43, May 2000, pp. 59-66.

[2] Folwer. M, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford. *Patterns of Enterprise Application Architectures*, Addison Wesley, 2002, ISBN 0-321-12742-0.

[3] Fortino, G., D. Grimaldi, L. Nigro. Distributed measurement patterns based on java and web tools. IEEE Autotestcon Proceedings, Sep. 1997, pp. 624-628.

[4] Jazdi, N.. Component-based and distributed web application for embedded systems. Proceedings of International Conference on Intelligent Agents, Web Technology and Internet Commerce – IAWTIC´2001, Las Vegas, USA, 2001.

[5] Pianegiani, F., D. Macii, P. Carbone. An open distributed control and measurement system based on abstract client-server architecture. IEEE Transactions on Instrumentation and Measurement, Vol 52, Issue 3, Jun 2003, ISSN:0018-9456, pp 686-692.

[6] Schneeman, R. D.. Implementing a standards-based distributed measurement and control application on the internet. http://ieee1451.nist.gov/framework.pdf, 1999.

[7] Spasov, G., N. Kakanakov. CGI-based applications for distributed embedded systems for monitoring temperature and humidity. Proceedings on the International Conference – CompSysTech'04, 17-18 June 2004, Rousse, Bulgaria, pp. I.6-1 – I.6-6.

[8] Spasov G., N. Kakanakov, N. Lupanov. Three-tier distributed applications. Proceedings on the conference Computer Science, 6-8 Dec 2004, pp. 172-177.

[9] Kakanakov, N.. Web based models for distributed automation. Journal of Automatics and Informatics, 2006 (in press). (in Bulgarian)

[10] http://net-lab.tu-plovdiv.bg/CNDEP/ – Controller Network Data Extracting Protocol.

[11] http://java.sun.com/products/jsp/ – Sun Microsystems JavaServer Pages.

[12] http://net-lab.tu-plovdiv.bg/ – Virtual laboratory of computer networks and distributed systems.