World Applied Sciences Journal 29 (Computer Sciences, Engineering and Its Applications): 79-81, 2014
ISSN 1818-4952
© IDOSI Publications, 2014
DOI: 10.5829/idosi.wasj.2014.29.csea.2233

Security Requirements in Distributed E-Business Environment

R. Udayakumar, K.P. Kaliyamurthie Khanaa and A.V. Allin Geo

School of Computing Science, Bharath University, Chennai-73, India

Abstract: This work presents a detailed analysis of the security requirements for multi-device environment for Distributed Computing in an e-business, still missing in the current literature. The purpose of this work is twofold. First, to provide protocol architects and software engineers with a map of security requirements in ubiquitous computing, through the evaluation of existing protocols and architectures. Second, to highlight architectural issues, including technologies and trade offs; in the design and implementation of a secure Distributed message transfer Architecture for various electronic devices. In addition to this the Wireless devices are being increasingly used to Transfer chunks of the data in the recent years. Since intermittent connectivity and client mobility are inherent in such environment this paper examine the impact of these factors on wireless devices. This paper also suggests a new proposal, push – based transfer mechanism to handle the client devices with lower message overhead and better fidelity. This is the most prevalent technologies for developing mobile applications.

Key words: Security Requirements · Distributed Message transfer Architecture · Multi-device Architecture and E-business

INTRODUCTION

Dynamic messaging involves bringing together content and preferences to create a custom message for each recipient. The complexity of this process increases when the client side has multiple devices. Dynamic messages inevitably utilize far more content than static ones. All this content must be created, cataloged, formatted and then sent to the indented recipients. Recent trends in organizations and software systems have included the move to Web-based information, the growth of mobile computing technologies and the use of "software components" which give the reuse and integrate functionality for communicating information among them [1, 2]. Mobile systems are deployed on cell phones wearable computers, PDAs and laptops, allowing mobile work with networked, usually thin-client, interfaces via WAP/WML and similar technologies.

The use of such dynamic messaging technologies has become wide-spread, but many existing systems are too difficult to integrate with domain-specific software applications, only work for specific user interface hardware, or provide inappropriate, thick-client architectural solutions [3, 4]. The main objective of this project is to develop a simple dynamic messaging technique for multi device environment which can be used within an organization for broadcasting vital information in a secure way to intended recipients which includes PC, mobile, printer and FAX. Both client side and server side have multiple devices and communication between PC and mobile phone is established via Bluetooth which improves the efficiency and security.

Generalized Messaging Architecture: In the proposed messaging architecture the client device characteristics have been detected and appropriate user interface has been provided in order to maintain the dynamic multi-device environment [5, 6]. There is no direct communication between client and server. Moreover client does not need to send a request to the server. It dynamically obtains the response from the server [7]. If the client device is a mobile, blue-tooth connection is being used between the server and the client. Device independency is the major criteria in the proposed

Corresponding Author: R. Udayakumar, School of Computing Science, Bharath University, Chennai-73, India.

three-tier messaging architecture, which consists of data source tier, middle tier and the client tier [8]. The service provider object and the shared objects constitute the data source tier. The middle tier has the server side message conversion application, which receives the file name, its MIME (Multiple Internet Mail Extension) type and the data from the container object. This tier decides which device has to receive the message or output and dynamically creates a formatted object file for that device [9]. Java embedded Formatted Object Processor is used to convert the output into an appropriate format and to create the corresponding fo files. The device independence feature is implemented in this tier.

The client tier is configured to handle four different types of devices: PCs, Printer, Fax, blue-tooth enabled mobile phones [10]. Each client device receives the output through specific server automatically, which has been initiated by the server side application in the middle tier. The client programs running in the client PCs receive output through UDP broadcast server [11]. Similarly the blue-tooth client code running in mobile phone receives the data from blue-tooth server. The output is sent to printer and fax through specific Java interfaces [12]. Push registries are used to send SMS messages to mobile clients.

In order to support device independence, the content must be delivered in a format compatible with the device since different devices require different user interface technologies [13]. For example a mobile phone has smaller screen and PDAs have limited or no colors.

The list of client devices to which the content has to be sent is stored in an XML file and the dynamic selection of the device is based on the content type. i.e the client device is determined on the MIME type of the data [14]. For example if the MIME type is an audio or video, then the content should be sent only to the client devices such as PC and mobile phone otherwise to all the devices including Printer and Fax. Once the client device has been identified, the information about the device, i.e about the driver, the URL of the driver can be obtained from the device configuration file.

The server obtains the required information about the client device from the configuration file and then dynamically makes changes in the output to be transmitted. Formatted Object Processor (FOP) is used by the middle tier for formatting the layout of the output specific to different devices. The formatting information is stored as. fo file where a separate style sheet document is required for specifying the formatting information. When the file is a simple text file, fo file can be created by adding the text content in the body of the fo file. If it is an image file, then image can be included as an external-graphics <src> tag.

The. fo file is then processed and converted to text file or PDF file using FOP embedded in a Java application. The driver namely, org.apache.fop.apps.Driver should be instantiated to embed FOP in the Java application. Once this class is instantiated, methods are called to set the Renderer to use and the OutputStream is used to output the results of the rendering (where applicable).

Once the FO file is converted to a text file with required layout, the file can be sent to printer and other devices. The layout formatting is extremely useful for customizing the layout for mobile devices since they have smaller screen. For sending the file to a PC, the server side application in the middle tier invokes a broadcast server running on the same machine. The UDP broadcast server broadcasts the whole file to the registered group of clients in the form of UDP packets. For sending the file to printer, the server side application invokes a Java Bean which has a method to send the file to the printer directly which is connected to the server machine. Even though the overall printing process is controlled by the system, the application has to initially set up a

PrinterJob, which stores the print properties and controls the display of print dialogs that is used to initiate printing.

For sending the file to fax machine, the server side application calls a method in another Java Bean which uses a third party API known as RFAX for sending the fax to the required fax machines. This API makes use of Java Communication API for establishing the serial port connection with the fax modem connected to the server computer. The client tier will receive the formatted output from the middle tier.

Java communication API is used to dial the required fax number, to get the connection stream and to send the data. It is a Java extension that facilitates developing platform-independent communication applications for technologies such as Smart Cards, embedded systems, fax, modems, display terminals and robotic equipments. Many third party APIs are available to directly send fax from Java application using javax.comm API. RFAX is one such API and is used in this model to send the output to fax machine.

The procedure for sending the output to a fax machine using RFAX API is explained below:

- RFAX implements the FaxProducer interface, which provides the pages that must be sent. The format of the interface is as follows:
  public java.awt.Image getFaxPage(int page);

The method returns an image object that contains the data to be faxed. The maximum size of the image is 1728 x 2387. Smaller images can be scaled to get the maximum size but the scale of the image shall not be larger than the maximum size.

RFAX also provides some ready to use FaxProducers which take strings or HTML as input source.

- The second step is to create a faxModem object, set up the port and modem configurations and call the sendFax() method.

Some modem specific flow control commands are set using the FaxModem reference. Then the connection is opened and the fax data is sent to the specified fax number.

Effective communication across various services is essential as accessibility is an important consideration. Three different types of user interfaces such as Text (for printer and fax), HTML (for Web browser) and WML (for mobiles) have been developed to support multiple devices. The presentation-tier information such as the type of device, the appropriate driver to be used with that device and the driver's location (URL) are represented in an XML file and can be retrieved by the server as and when it is needed. This information has been plugged into the generalized messaging architecture thus enabling reliable and device independent communication.

REFERENCES

1. Ganesh, J., S. Padmabhuni and D. Moitra, 2004. Web services and Multi-Channel Integration: A Proposed Framework, IEEE International Conference on Web Services, pp: 70-77.
2. John Grundy, Xing Wang and John Hosking, 2002. Building Multi Device, Component- Based, Thin-client Groupware, Issues and Experiences, Proceedings of 3rd Australian User Interfaces Conference.
3. John Grundy and Biao Yang, 2003. An environment for developing adaptive, multi-device user interfaces, Proceedings of the Fourth Australasian user interface conference on User interfaces, pp: 47.
4. Srijeeb Roy, 2006. Push messages that automatically launch a java mobile application, http://www.javaworld.com/javaworld/jw-04-2006/jw-0417-push.html
5. Daniel A. Menasce, 2003. Web Server Software Architectures, IEEE Internet Computing, 7(5): 89-92.
6. Daniel A. Menasce, 2003. Workload characterization, IEEE Internet Computing, 7(5): 89-92.
7. Datacomm Research Company, 2000. Global CDMA Business Opportunities, August 2000. http:// www.mindbranch.com/ catalog/ print_product_page.jsp?code=R222-005.
8. Udayakumar, R., Kumaravel A. Rangarajan, 2013. Introducing an Efficient Programming Paradigm for Object-oriented Distributed Systems, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(5S): 4596-4603.
9. Udayakumar, R., V. Khanaa and K.P. Kaliyamurthie, 2013. Performance Analysis of Resilient FTTH Architecture with Protection Mechanism, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(6): 4737-4741.
10. Udayakumar, R., V. Khanaa and K.P. Kaliyamurthie, 2013. Optical Ring Architecture Performance Evaluation using ordinary receiver, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(6): 4742-4747.
11. Saravanan, T. and R. Udayakumar, 2013. Optimization of Machining Hybrid Metal matrix Composites using desirability analysis, Middle-East Journal of Scientific Research, ISSN:1990-9233, 15(12): 1691-1697.
12. Saravanan, T., V. Srinivasan and R. Udayakumar, 2013. Images segmentation via Gradient watershed hierarchies and Fast region merging, Middle-East Journal of Scientific Research, ISSN:1990-9233, 15(12): 1680-1683.
13. Thooyamani, K.P., V. Khanaa and R. Udayakumar, 2013. Detection of Material hardness using tactile sensor, Middle-East Journal of Scientific Research, ISSN:1990-9233 15(12): 1713-1718.
14. Thooyamani, K.P., V. Khanaa and R. Udayakumar, 2013. Blue tooth broad casting server, Middle-East Journal of Scientific Research, ISSN:1990-9233, 15(12): 1707-1712.