A RESPONSIBILITY-DRIVEN ARCHITECTURE FOR MOBILE ENTERPRISE APPLICATIONS

Qusay H. Mahmoud
Department of Computing and Information Science, University of Guelph, Guelph, Ontario, Canada N1G 2W1

Keywords: Mobile enterprise, mobile architecture, responsibility-driven, WAP, J2ME, mobile services, mobile agents

Abstract: This paper deals with wireless applications that get downloaded, over the air, on handheld wireless devices (or mobile devices) and get executed there. Once running, they may need to interact with applications residing on remote wired servers. The motivation for this work is provided in part by the characteristics of the wireless computing environment. There are several implications of these characteristics that require a software architecture that reduces the load on the wireless link and supports disconnected operations. We present a responsibility-driven architecture that enables mobile thin-clients to interact with enterprise servers. We extend this architecture with mobile agents to reduce the load on the wireless link and support disconnected operations. The architecture is simple and easy to implement, but also effective, scalable, and capable of supporting multiple devices.

1 INTRODUCTION

The nature of the wireless Internet will be different from simply accessing the Internet wirelessly. Users with mobile devices, being mostly mobile, have different needs, motivations and capabilities from wired users. Mobile enterprise applications are mobile/wireless computing applications and services that allow the user to access and use existing enterprise applications from mobile devices. The explosive growth of the handheld wireless devices market, however, is stimulating the computing research community to clone almost any technology developed for desktop networked computers to handheld wireless devices connected to a wireless network. Wireless networks are, however, unreliable and suffer from low bandwidth and have a greater tendency for network errors. In addition, wireless connections can be lost or degraded by mobility.

Several solutions have been proposed and several technologies have been developed to address the constraints of mobile devices and the wireless environment. The solutions and technologies that are available can be classified into two categories [Beaulieu, 2002 & Burkhardt et al, 2002]:

Browser-based

This is similar to the current desktop-browser model where the device is equipped with a browser. The browser-based approach is used by the Wireless Application Protocol (WAP).

Native applications

Compiled applications where the device has a runtime environment to execute them. Sophisticated wireless services are only possible with this model

Another category is the *hybrid application model* that aims to incorporate the best aspects of both categories above. The browser is used to allow the user to enter URLs to download native applications from, and the runtime environment is used to let these applications run on the device. In this paper we are concerned with downloadable interactive applications built using the fast growing Java 2 Micro Edition (J2ME) platform [Sun Microsystems, 2000].

The rest of this paper is organized as follows. Section 2 provides an overview of the J2ME and its Mobile Information Devices Profile (MIDP) that targets cell phones and two-way pagers. Section 3 provides a solution on how J2ME and WAP can be integrated together providing a useful environment for the mobile enterprise. The details of the responsibility-driven architecture are presented and discussed in Section 4. Finally, concluding remarks are presented in Section 5.

H. Mahmoud Q. (2004).
A RESPONSIBILITY-DRIVEN ARCHITECTURE FOR MOBILE ENTERPRISE APPLICATIONS.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 125-129
DOI: 10.5220/0002641001250129

2 J2ME

The Java 2 Micro Edition (J2ME) is aimed at the consumer and embedded devices market. It specifically addresses the rapidly growing consumer space that covers commodities such as cellular telephones, pagers, palm pilots, set-top boxes, and other consumer electronics. The J2ME provides a complete set of solutions for creating state-of-the-art network applications for consumer and embedded devices. It enables device manufacturer, service providers, and application developers to deploy compelling applications and services to their customers. The J2ME platform is targeted at two product groups:

1. Personal, mobile, connected, information devices, such as cellular phones, two-way pagers, and organizers.
2. Shared, fixed, connected, information devices, such as set-top boxes, Internet TVs, and in-car entertainment and navigation systems.

The Java 2 Micro Edition (J2ME) can be used to enhance the user experience on mobile devices. It provides a Java runtime environment for cell phones, palmtops, and two-way pagers; also includes libraries and APIs for:

- Programming user interfaces using the Mobile Information Device Profile (MIDP) [Sun Microsystems, 2000] user interface API. This API provides high-level components offering standard interactions such as lists and forms, and low-level components for customized look-and-feel (canvas). This API enables developers to create highly interactive mobile applications and flexible user interfaces.
- Storing data persistently on a device using RMS. Such APIs also enable developers to write applications (such as mobile games) that work even when the device is not connected to a wireless network.
- Establish network connections with remote servers or other devices using the CLDC Generic Connection Framework. This API enables developers to write mobile network-aware thin-clients that connect to enterprise services using standard networking protocols such as HTTP.

MIDP implementers are required to provide support for the HTTP protocol; this is mainly because HTTP can be implemented on IP-enabled and non IP-enabled devices. In addition, the HTTP protocol is firewall-friendly and already has support for error codes. The HTTP protocol can be easily used to communicate between mobile clients and enterprise applications.

3 WAP MIGHT BE DEAD, BUT WHAT HAVE WE LEARNED?

WAP and MIDP solve similar problems but each can learn a couple of things from the other. There are special features that are available in WAP but not in MIDP and vice versa. Here, we summarize the important features:

- WAP has support for additional phone functionality such as setup and integration with address book. Despite the fact that not all WAP-enabled devices support this feature, there are no comparable APIs available for this in MIDP. It is very likely that this will be possible in a future version of MIDP.
- MIDP provides a low-level graphics APIs that enable the programmer to have control over every pixel of the device's display. This is important for entertainment applications (such as games) in a wireless environment. MIDP is the ideal technology for mobile games. The very nature of MIDlets (they exist on the device until they are explicitly removed) allow users to run them even when the server becomes unavailable (support for disconnected operations).
- WAP's Wireless Markup Language (WML) provides the tags and the possible presentation attributes, but it doesn't define an interaction model. For example, WML defines a `SELECT` element for providing a list. Some WAP-enabled devices interpret the `SELECT` tag as a popup menu list while others interpret it as a menu that can be used for navigation. Therefore, there is no standard interaction model defined for this element. If a developer uses it, the application may run well on some devices and poorly on others. MIDlets, on the other hand, provide a clearly defined standard for interaction using commands.

3.1 But a Micro-Browser is Essential

MIDlets combine excellent online and offline capabilities that are useful for the wireless environment, which suffers from low bandwidth and network disconnection. Integrating WAP and MIDP opens up possibilities for new wireless applications and over the air distribution models. Therefore, WAP and MIDP shouldn't be viewed as competing technologies but rather as complementing ones. If

you have any experience working with MIDlets, and more importantly downloading MIDlets on real physical devices (such as the Motorola/Nextel i85s cellular phone) then you might be aware that provisioned MIDlets over the air is still a work in progress. In order to facilitate over the air provisioning, there is a need for some kind of an environment on the handset that allows the user to enter a URL for a MIDlet Suite, for example. This environment could very well be a WAP browser as shown in Figure 2.

Figure 1: Integrating WAP and J2ME

Similar to Java Applets that are integrated in HTML, MIDlets can be integrated into a WML page. The WML page can then be called from a WAP browser and the embedded MIDlet gets installed on the device. In order to enable this, a WAP browser (with support for over the air provisioning) is needed on the device. Another alternative approach for over the air provisioning of MIDlets is the use of Short Message Service (SMS) as has been done by Siemens where the installation of MIDlets is accomplished by sending a corresponding SMS. If the SMS message contains a URL to a Java Descriptor (JAD) file specifying a MIDlet Suite then the recipient can install the application simply by confirming the SMS.

4 A RESPONSBILITY-DRIVEN ARCHITECTURE

Responsibility-Driven design [Wirfs-Brook et al, 2002] is a design approach that emphasizes behavioral modeling using objects, responsibilities and collaborations. In a responsibility-based model, objects play specific roles and occupy well-known positions in the application architecture. Each object is accountable for a specific portion of the work. They collaborate in clearly defined ways, contracting with each other to fulfil the larger goals of the application. By creating a "community of objects", assigning specific responsibilities to each, we build a collaborative model of our application. Here objects are service-providers, information holders, controllers, and interfaces to the outside world! Each must know and do its part.

The responsibility-driven architecture for wireless applications makes applications easier to implement, test, and maintain. This is mainly because each object provides a specific service and therefore the separation between responsibilities is clear.

Any architecture for wired/wireless integration will consist of at least three-tiers. The WAP architecture, for example, consists of 3-tiers: the WAP User-Agent (or the client), the WAP Gateway, and the Web Server that hosts the HTML, WML, and other server-side scripts. The responsibility-driven architecture that we propose here consists of at least three-tiers; a high-level view is shown in Figure 2. The architecture is simple but yet effective: the client accesses and uses Java servlets that will in turn communicate with Enterprise JavaBeans (EJBs) to manipulate data on behalf of the client. In other words, the responsibilities in this architecture are divided as follows: clients (they want to use remote enterprise applications); middle-tier Java servlets that act on behalf of the user; and EJBs that implement the business logic of enterprise applications. This division of responsibilities makes mobile applications easier to implement and maintain. In addition, it would be easy to add other plug-n-play components that support authentication, for example.

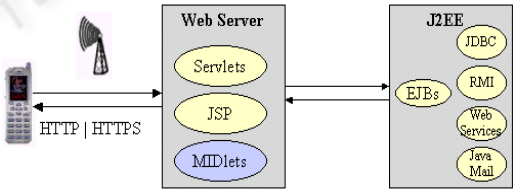

Figure 2: High-level view of the architecture

Now, let's look at the components of this architecture in mode details:

- A mobile client application (based on WAP) or a MIDlet that provides the user interface and allows the user to interact with the application.
- The MIDlet communicates with a Java servlet or JavaServer Page (JSP) over HTTP, or HTTPS if necessary. Servlets and JSPs act as mediators between the mobile thin-client and enterprise applications. A mediator is a service that functions simultaneously as a server on its front end and as a client on its backend [Wiederhold, 1992]. It is much like a proxy; however, it performs some useful processing on the request.

In this model, the client makes a request to the mediator, which then contacts the original enterprise service to satisfy a request or invoke a remote method; the mediator may produce an XML response which it serves to the client. Mobile devices can benefit greatly from mediators. All enterprise applications available today can be made available to mobile users' handheld devices through mediators that act as a middleware between mobile devices and enterprise applications.

- A servlet receives requests from MIDlets and act as a client of an EJB component. Based on this interaction, the servlet generates a response and sends it back to the client (WAP browser or MIDlet).
- An EJB component is the application's business logic. It provides standard services such as transactions, security. It allows developers to concentrate on implementing business logic.
- The servlet and EJB components may use additional APIs to access enterprise information and service.

This architecture supports multiple clients. The client can either be a Java-based application (MIDlet) running on a Java-enabled device, or an application running on a browser-enabled device. In this case, the servlet or JSP page would be able to present the client with the appropriate reply (WML content). This really means that while the MIDlet and the WAP browser need to access different Servlets and JSP, they would use the same EJB components for business logic. As a result, this architecture supports multiple clients without any impact on the business logic of the application.

This architecture can be implemented using the J2ME MIDP for the client-side and the J2EE for the server side. The client-side application of this architecture can use the classical Model View Controller (MVC). The model would contain the data for the application; the view would contain code for managing data when interacting with the user; and the controller would look after the logical flow of the user interface. Figure 2 shows the interaction between the mobile device and the central server.

4.1 Client-Server Communication

As we have mentioned above, MIDP prefers the use of the HTTP protocol for messaging over any other communication mechanisms such as sockets or datagrams. As a result, all MIDP-enabled devices are required to support the HTTP protocol and therefore any application that uses the HTTP protocol for messaging would be portable across all devices. The use of HTTP is actually a good thing for mobile enterprise applications because many enterprise servers are separated from mobile clients by firewalls, and HTTP is a firewall friendly protocol.

It may not be reasonable to assume that all mobile clients within one enterprise will be using the same servlet. For scalability issues as well as other issues such as authentication, several servlets can be deployed and some can be made location-aware so that different mobile clients would access different servlets that may be offering the same or different services.

4.2 Disconnected Operations

The characteristics of the wireless environment, namely: low bandwidth and greater tendency for network errors and disconnection call for mechanism that reduces the load on the wireless link and supports disconnected operations. One way to support disconnected operations is through the use of Java Message Service (JMS), which can be easily integrated into the architecture.

Mobile Agents [White, 1997] can aid in providing a reliable technology for message transport. Mobile agents are well suited for many applications that are communications-centric such as processing data over unreliable networks (such as wireless networks). In such environments, the low reliability network can be used to transfer agents, rather than a chunk of data from place to place. In this scenario, the agent can travel to the nodes on the network, collect or process information, without the risk of network disconnection, then return home.

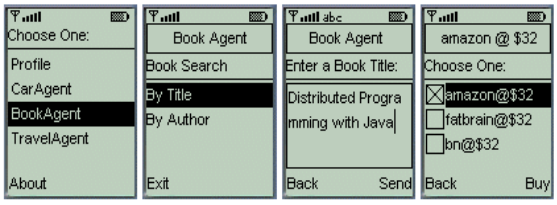

Figure 3: MobiAgent's BookAgent service

Our responsibility-driven architecture can be easily extended to support mobile agents as follows: the mobile device would have an interface agent that allows the user to access, configure, and dispatch mobile agents. The interface agent communicates with a servlet that is responsible for dispatching mobile agents; such agents can be implemented using Java Remote Method Invocation (RMI). When the agent finishes its work, the results are sent back

to the mobile device (through the servlet); if the device is disconnected from the network, the results are saved in the user's profile and an SMS message is sent to the user informing her that the results are available. This strategy has been implemented in the MobiAgent [Mahmoud, 2002] system architecture. Figure 3 shows a snapshot of how it works.

4.3 Personalization

In the responsibility-driven architecture, new modules can be easily added to be responsible for tasks such as location-awareness and personalization. Personalization means that only relevant information should be downloaded to the mobile devices and then present that information effectively taking into account the user's preferences and history along with the task at hand. In personalization, however, data is persistent by nature and therefore this is different from maintaining session states where data is transient.

Personalization is important in mobile enterprise applications; as an example consider a login screen where the user needs to enter a valid user name and password in order to use an application. Such screen login doesn't change from session to session. Thus, the user should be asked to enter the login information only once.

Personalization is not without privacy and security issues, however. An automatic login feature is one way to improve the user experience and avoid tedious authentication procedures. Personalization may require storing data on the device and therefore special care should be taken to protect its integrity and confidentiality. As an example, consider the case where a piece of sensitive information is stored on the device. It should be ok to automatically send the sensitive information to the server but not display it on the screen. If the user wishes to view such sensitive information, the operation should require the user to enter a password. This is important so that if the device is lost and someone finds it, the sensitive information cannot be retrieved out of the device.

4.4 Benefits of the Proposed Architecture

The benefits of this responsibility-driven architecture can be summarized as follows: First, it is simple but yet quite effective and easy to implement using existing technologies; it enables multiple mobile clients to access enterprise services; the architecture is transparent as the client is not aware of the location of the enterprise services; it provides anytime, anywhere access to enterprise services; it is based on open industry standards, such as HTTP and XML; it promotes loose coupling between mobile thin-clients and enterprise servers; it is inherently scalable since it is based on proven scalable architectures such as Servlets; and finally it is Web services ready: the latest release of the J2EE (J2EE 1.4) enables developers to expose existing J2EE enterprise applications as Web services that would be ready to be consumed by mobile clients.

5 CONCLUSION

In this paper we have discussed the characteristics of the wireless environment and the technologies that are available for developing mobile applications. A responsibility-driven architecture for mobile enterprise applications was discussed. This architecture provides a software infrastructure for accessing enterprise applications from mobile clients. The architecture is simple yet effective since responsibilities are divided among its components; such division makes it easier to develop, test, and maintain mobile enterprise applications. The architecture is scalable and capable of handling growth since it is based on proven scalable server solutions such as servlets and EJBs. Our future work includes extending this architecture with mobile middleware that will play a crucial role in mobile enterprise computing and mobile commerce systems.

REFERENCES

Beaulieu, M., 2002. Wireless Internet Applications and Architecture. Addison-Wesley.

Burkhardt, J., Henn, H., Hepper, S., Rintdorff, K., 2002. Pervasive Computing Technology and Architecture of Mobile Internet Applications. Addison-Wesley.

Mahmoud, Q.H., 2002. An Agent-based Approach to the Wireless Internet. In Journal of Internet Technology, Vol. 3, No. 2, pp. 153 – 158.

Sun Microsystems Inc., (J2ME), 2000:
http://java.sun.com/j2me

White, J., 1997. Mobile Agents. In Bradshaw, J.M. (ed.), Software Agents, AAAI/MIT Press, pp. 437-472.

Wiederhold, G., 1992. Mediators in the Architecture of Future Information Systems. In IEEE Computer, 25(3), pp. 38-48.

Wirfs-Brook, R., McKean, A., 2002. Object Design: Roles, Responsibilities, and Collaborations. Addison-Wesley.