WRITTEN BY SILVANO MAFFEIS

Wireless Information Services Need Java Messaging

Preparing for the revolution

This year will be the genesis of mobile devices and wireless applications. In fact, several European and American carriers have begun rolling out high-speed, packet-oriented wireless networks based on General Packet Radio Service (GPRS) as well as other proprietary standards. As well, I've noticed very interesting Java devices, notably communicators and smart phones, based on the Symbian operating system and on the J2ME MIDP environment.

The combination of Java-enabled devices and packet-oriented wireless bearers is a compelling catalyst leading to the rapid adoption of "collaborative" and highly interactive applications, delivering the right information (according to our user profile) in real-time.

What does this mean to us? Maybe a new type of Internet, based on zillions of gadgets exchanging information via wireless networks!

Such communicators and smart phones will always be on and capable of receiving information in real-time. This means a new way of delivering information to the user. Browsing might not be the right model in this mobile world, when you consider that you will carry your wireless companion around all times, and that this little device is able to receive alerts and e-mails constantly. As a user, I don't want to chase or browse information on my mobile device. Rather, I want to be notified when something interesting happens – receive an alert.

Reality check. Developing Java applications that connect mobile devices to server applications in a scalable and reliable manner is a challenge. In the wireless world a mobile application must be able to "speak" various protocols (TDMA, CDPD, GPRS, UMTS, WAP, SMS, you name it) and cope with varying bandwidth and loss of network coverage (imagine issuing a stock purchase transaction from your mobile device while driving through a tunnel). Wouldn't it be nice if one could build mobile Java applications on a platform or middleware to take care of those issues?

Building a highly scalable platform to host mobile Java applications isn't easy either. I've been involved in the development of such a platform during the last couple of years. The biggest challenge was to devise a communications middleware that copes well with the specific aspects of wireless networks, namely, varying bandwidth, intermittent connectivity, and packet loss. We opted to base our middleware on the messaging paradigm and notably on the Java Message Service (JMS) standard because 1) messaging can deal with intermittent communication links using store-and-forward message delivery, and 2) messaging can hide long communication latencies by transmitting messages in asynchronous mode (this means the sender of a message doesn't have to wait until the message has reached its destination). Also, messaging middleware can be implemented in a lightweight manner, meaning that it can be deployed directly on a mobile device, such as a Palm or a Symbian communicator. The wireless JMS is born!

What does this mean? By taking advantage of a wireless-enabled JMS middleware, you can develop mobile applications that deliver information over various wireless bearers, in spite of sudden changes in bandwidth or network coverage. For application developers it truly offers a write-once-go-anywhere possibility irrespective of the bearer. The delivery of important information is guaranteed by the JMS store-and-forward message-queuing mechanism. The timely delivery of important information to large groups of receivers is made possible by the JMS publish-and-subscribe model and by taking advantage of the multicast capabilities of wireless bearers such as GPRS.

This type of JMS middleware acts as a highly versatile bridge between server-side applications running on a J2EE platform and J2ME applications running on a mobile device. If a server application transmits information to a mobile device that's turned off, then nothing is

silvano.maffeis@softwired-inc.com

AUTHOR BIO

Dr. Silvano Maffeis is CTO at Softwired (www.JavaMessaging.com), a leading vendor of application-to-application messaging solutions. He is the codeveloper of iBus, a JMS middleware for wireless and wireline systems.

lost as the JMS middleware will store and deliver the information automatically and transparently. The same occurs when the device transmits information to a server (for example, a purchase transaction). The information, which takes the form of messages, is stored on the device and delivered to the server automatically as soon as a wireless connection is physically possible.

So what problem does this solve? The commonality between mobile commerce platforms, wireless financial data feeds, location services, instant messengers, multiuser games is that all those solutions embody a distributed systems architecture in which various pieces of (Java) code need to communicate by wireless and wireline protocols. JMS has proven viable when complex distributed systems need to be developed and deployed in short time.

This year we'll witness innovative wireless services and applications based on communicator devices, Java, packet-oriented wireless bearers, and application servers at the backend. A wireless messaging middleware could stand at the vanguard of this wireless information revolution!