GPars - Groovy Parallel Systems

Jon Kerridge

Version 1.0, 2015-10-29

Table of Contents

Mobile Processes: Ubiquitous Wireless Access 1
Node-to-node process Communication Within A Network 1
Potential Use Cases 1
Service Delivery 2

Mobile Processes: Ubiquitous Wireless Access

The previous chapter showed how it is possible to create anonymous network channels over which agents could be transferred.

Node-to-node process Communication Within A Network

In this chapter, we take that concept one stage further and provide a mechanism whereby a process can be communicated from one node to another within the network. The only requirement is that the receiving node has to run a simple process that loads the mobile process. This is further extended to load a process from a server over a wireless network to a mobile device. The mobile device becomes a member of the server's network for the duration of the interaction. The mobile device scans for accessible wireless networks and then is able to download a process from that network by which it can interact with the service provided.

Potential Use Cases

Retail Environment

This technology could be used in a retail environment to let stores make offers to customers, as they walk into the store, based upon their previous shopping patterns. In addition, the store could make offers on surplus items to customers they know might be susceptible to the offer. The only requirement is that the customer has a mobile device into which the process loading process has been installed. The customer would also need to store some means of identifying themselves to the store‟s systems but with loyalty or reward cards this is not a problem.

Medical Situations

The technology could be used in a hospital environment to allow access to electronic patient records by registered users, using their own mobile device. The great advantage being that the location of a person can be determined by the wireless access points that are available and this could result in the most appropriate process being downloaded into the mobile device depending upon the user and their role. Obviously, some form of authentication process would be required to ensure authorised access but the advantage of this style of interaction is that no sensitive data is held in the mobile device.

Public Service Ideas

Finally, it could be used in museums to provide additional resources to visitors about the items on display. In this case, rather than using wi-fi, we could use **Bluetooth** to give more locality of information. The downloaded process could provide additional information in the form of an audio stream giving an aural description of the exhibit, possibly supported by an image that shows the particular part of the object being described. The audio stream could be in any natural language. The particular advantage for the museum is that visitors can use their own mobile devices, provided they

have the process to download other processes.

Service Delivery

The mobile process capability is provided by a mobile package within the `jcsp.net` capability. It deals with the dynamic loading of classes over the network in an efficient manner that is totally transparent to the programmer and of the underlying network technology.

Processes are loaded just like any other object as a **Serializable** data object. The processes will include some of the network channels in their definition that will allow the loaded process to communicate from the mobile device to the server. However, channels that enable communication from the server to the loaded mobile process will need to be created dynamically.

This application is different from others because we are running the systems of processors with different resource capability. In particular, the requirement to run **Groovy** on a mobile device is problematic, given its size and the underlying functionality it requires in terms of reflector requirements. To this end, all the processes that execute on the mobile device are written in pure **Java**. They do, however, interact with server processes written in **Groovy**.

Thus this chapter demonstrates that **Groovy** and **Java** components can be combined into a single application environment. The dominant requirement being that all devices run using a `Java Virtual Machine`, which is the case with most mobile devices.

Want to read more of this chapter? Download this chapter's PDF here.