Exploiting Logical Mobility in Mobile Computing Middleware

Stefanos Zachariadis, Cecilia Mascolo and Wolfgang Emmerich
Dept. of Computer Science, University College London
Gower Street, London WC1E 6BT, UK
{s.zachariadis,c.mascolo,w.emmerich}@cs.ucl.ac.uk

In the past few years, we have witnessed the growth in availability of mobile computing devices, such as mobile phones, personal digital assistants (PDAs), laptop computers and the like, combined with the advent of new wireless networking systems, such as 802.11b, Bluetooth, and GSM/GPRS. Users of such devices demand access to networked information and services at all times and locations, and, as such, encounter different networking structures and topologies. The introduction and use of mobile code techniques, from the acceptance of Java applets in web pages, to the creation of mobile agent platforms has demonstrated that it is possible to create dynamic systems that support code migration. We argue that logical mobility over physical mobility can offer the flexibility that traditional approaches cannot, helping to provide innovative solutions to mobility issues.

We consider the following forms of mobile interactions, according to [1]: Client/Server interactions (CS), whereby the request of a client triggers the execution of a unit of code in a server and returns the results to the client, Remote Evaluation (REV), where a device can send code to another host, have it executed and retrieve the result, Code On Demand (COD), where a host can request a unit of code from another device to be retrieved and executed, and Mobile Agents (MA), where an agent is an autonomous unit of code that decides when and where to migrate. Moreover, we consider devices that can be nomadically connected to a fixed network (e.g., a laptop dialling up to an ISP), devices that are constantly connected to a fixed network over a wireless connection (e.g. a GPRS-enabled mobile phone), devices that are connected to ad-hoc networks (e.g. Bluetooth piconets) and any combinations of the above.

Related work on middleware supporting code mobility, geared for mobile environments include Sun Microsystems Jini[5], a technology based on Java which employs Code on Demand and Remote Evaluation techniques to deliver distributed services in high-speed and long lived networks. Jini allows clients to discover which services are currently offered and use them. Jini provides a centralised framework, which requires lookup services, functioning as indexes of services offered, to operate. Jini, and especially the JMatos mobile implementation, can be used by mobile devices to deliver context aware services to mobile devices, connected to a fixed network nomadically. Jini is not, on the other hand, particularly suitable for allowing mobile devices to offer services themselves, particularly in ad-hoc environments which lack a centralised lookup service. Lime[3] is another example, which provides application developers with a data-sharing middleware, geared for mobile agents and ad-hoc networks. With hosts acting as containers for mobile agents, Lime offers a Linda Tuplespace implementation for coordinating and sharing data, an implementation extended with location and mobility primitives. It does not currently provide any form of security and the fact that it only offers a flat tuple space as the only common data structure, limits the processing that can be made on the shared information. Moreover, the advantage of having mobile agents in physically mobile hosts is not made clear.

Examples Logical over Physical Mobility

Limited Resources and Dynamic Update. One of the traditional goals of the Human Computer Interaction design principles has been to make computers as easy to use as possible. To approach the stage of ubiquitous computing, the use of computing devices, and mobile devices in particular, must be made transparent; There should be no complex installation procedure for example, in order to allow the device to perform a specific task.

However, as these devices only have limited resources, it is very difficult for manufacturers to preload on to the device the code needed for every possible use that the device might have. Hence,users are forced to follow to complex installation and uninstallation instructions, that can confuse and alienate them. COD can be effectively used to assist in this scenario. Imagine having applications that transparently download audio codecs to play a new audio format. Or devices that are automatically able to use resources that are currently present. The device can download on demand the code that is needed, either from a peer in an ad-hoc scenario, or from a trusted third party (a centralised source). When the code is no longer needed, the device can choose to delete it, conserving resources. Security mechanisms such as digital signatures can be used to ensure the safety and authenticity of the downloaded code.

Location-Based Reconfigurability and Services. A mobile architecture which allows deploying and utilising services similarly to Jini, can allow a mobile user to transparently use any services that are available to his or her current location. COD can allow a mobile user to transparently operate services that are currently available in the user's location. For example a user can be automatically presented with a graphical user interface to order movie tickets, upon entering a cinema's premises.

Communication in Disaster Scenarios. Mobile agents can be employed in an ad-hoc networking structure to deliver best effort messaging and communication in a disaster scenarios. The message can be encapsulated in a mobile agent which migrates from host to host, until it reaches the required destination. In fixed networking scenarios, Mobile Agents can be used to encapsulate the next generation of Short Message Service (SMS) messages: Encapsulating the message in an agent, and delivering it to the recipient through a message centre, to be executed on the recipients device.

Shopping and Limiting Connectivity Costs. One of the reasons why electronic shopping is yet not very popular from mobile devices, is the fact that it usually takes far too long for a user to navigate through a site, usually viewing it from a tiny screen) in order to order the product. Considering that wireless connections are expensive, the cost of shopping from a mobile device can be quite high. Mobile agents could be a solution to this problem, encapsulating the description of the product the user wishes to buy, finding the best price, and performing the actual transaction for the user.

Distributing Computations and Exploiting Computational Resources. As mobile devices usually have limited resources, REV techniques can be used to distribute computations to more powerful hosts that are reachable using the available infrastructure allowing for faster application execution, and a better perceived end-user experience.

Next Generation Mobile Middleware

We argue that a mobile computing middleware supporting mobile code techniques can increase the application developers' flexibility in solving scenarios similar to those shown previously, in innovative ways that the current state of the art cannot support. Next generation middleware should be able to allow devices to migrate between different network infrastructures, use COD techniques to dynamically update itself and offer a protected environment to host mobile agents and serve REV requests. Through the use of context-awareness techniques, the middleware should notify applications of their current context, so that they can adapt accordingly[2]. Different mobile code paradigms could be plugged-in dynamically and used when needed after assessment of the environment and application. We are also currently working on providing such a middleware and integrating it with a design methodology, possibly based on UML, that can be used by application programmers to evaluate the use of each mobile code paradigm, depending on different contexts. A similar approach for the evaluation of mobile code paradigm use in distributed system has been used in [4].

References

[1] A. Fuggetta, G. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Trans. on Software Engineering*, 24(5).

[2] C. M. L. Capra, S. Zachariadis and W. Emmerich. Towards a mobile computing middleware: a synergy of reflection and mobile code techniques. In *Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems*, 2001.

[3] A. L. Murphy, G. P. Picco, and G.-C. Roman. LIME: A Middleware for Physical and Logical Mobility. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, May 2001.

[4] R. M. V. Grassi. Primamob-uml: a methodology for performance analysis of mobile software architectures. In *Proceedings of the Third International Workshop on Software and Performance*, 2002.

[5] J. Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7):76–82, 1999.