Towards a Mobile Computing Middleware: a Synergy of Reflection and Mobile Code Techniques

Licia Capra*, Cecilia Mascolo, Stefanos Zachariadis and Wolfgang Emmerich

Dept. of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK
{L.Capra|C.Mascolo|S.Zachariadis|W.Emmerich}@cs.ucl.ac.uk

1 Introduction

In the past decade, middleware technologies built on top of network operating systems have greatly enhanced the design and implementation of distributed applications. In particular, they succeeded in hiding away many requirements introduced by distribution, such as heterogeneity, fault tolerance, resource sharing, and the like, from application developers, offering them an image of the distributed system as a single integrated computing facility.

Recent advances in wireless networking technologies, such as WaveLan and Bluetooth, and the growing success of mobile computing devices, such as third generation mobile phones and PDAs, have opened up the door to new classes of distributed applications that impose challenging problems to developers. Wireless devices face temporary loss of network connectivity when they roam; they need to discover other mobile devices in an ad-hoc manner; they are likely to have scarce resources, such as low battery lifetime, slow CPU speed, etc. Applications running on these devices have to be aware of, and adapt to, frequent and unannounced changes happening in their execution environment, such as high variability of network bandwidth, new physical location, and so on.

Middleware solutions developed for wired distributed systems cannot be successfully applied in this scenario, as the principle of transparency that has driven their design runs counter to the new degrees of awareness imposed by mobility. We believe that middleware capable of supporting the development of mobile applications will play a key role in the success of wireless technologies and mobile applications, the same way traditional middleware did for wired distributed systems.

In this paper, we propose a mobile computing middleware based on the principle of reflection and mobile code. In particular, this middleware is capable of: providing applications the desired level of flexibility necessary to react to changes happening in the execution environment (through reflection); coping with changes that have not necessarily been foreseen by the middleware designer (through mobile code techniques).

2 Why Reflection and Mobile Code

In this section we present an example in the context of image processing and image based services that we are partly developing with Kodak. Our goal is to show the need for new middleware in the context of mobile computing and to describe how reflection and mobile code techniques may help to add levels of flexibility.

* Contact Author: Licia Capra. Address: Dept. of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK. Phone: +44 20 7679 3645. Fax: +44 20 7387 1397. E-mail: L.Capra@cs.ucl.ac.uk.

Let us consider a number of mobile devices (cameras, smart phones, PDAs and laptops) that are able to communicate through an ad-hoc network infrastructure. Some base stations equipped with fixed network connectivity, Internet facilities and high resolution displays are also present. Mobile devices store pictures in their memory. Pictures may be shared among devices and may be displayed and shipped through the Internet to base stations in order to be permanently stored in some available database.

The sharing, displaying, shipping and uploading of the images may happen in many potentially different ways, according to the current network availability and the devices involved, as they greatly vary in terms of memory, processing power, etc. Different compression, rendering, loading, shipping, linking, caching protocols may have to be used by the middleware.

In this scenario, the middleware cannot transparently choose which protocol to apply based on the current configuration of the execution context, as the applications possess vital knowledge that must be exploited during this choice. For example, an application may want to persistently store on a remote database only a specific subset of the images the device is carrying, in order to save battery power, and the middleware has no way to guess this subset. We believe reflection can be used to allow applications to inspect contextual information and to change middleware behaviour accordingly at run-time. In this way, applications have a say in which way images are handled, shipped, cached or linked.

However, reflection alone is not enough. The devices are not able to know a-priori which protocols they are going to need in which situation, and which other devices they are going to encounter. Devices cannot load all the possible behaviours into memory, due to memory constraints; moreover, new behaviours may be delivered from time to time to cope with unforeseen context configurations and new application needs.

Mobile code techniques can be used to load proper compression/cropping/encryption modules on the device only when needed. Moreover, mobile code strategies can be used to move the computation outside the device to a close and more powerful base station, depending on which other hosts "leasing" their resources are in reach.

3 Our Mobile Computing Middleware

In this section we describe the mobile computing middleware model we are developing based on reflection and mobile code techniques.

Reflection allows the flow of environmental information, such as the hosts/services currently in reach, the remaining battery life, the location, the bandwidth condition, between the middleware and the application layer. The application is allowed to inspect this information and to use it dynamically to instruct the middleware on how to behave. Reflective middleware can also prompt the application to determine how the middleware should react to changes in the environment. Middleware and applications have obviously to agree on a specific format for this information; our suggestion is to use XML for the encoding of what we call *application profile*, that is, meta-information that associates policies (i.e., middleware behaviour) to context configurations.

Reflection enables adaptability and flexibility only in those conditions that the middleware designers have considered likely to be unstable at design time. *Mobile code* techniques overcome this limitation as the behaviour of the middleware can be adapted to unforeseen situations by downloading new protocols either from a service provider or from other peers in reach which use the same behaviour. Mobile code would also allow the device to move the computation to more powerful machines in the neighborhood. This requires the hosts

that are in reach to exchange information about what services, code and resources are available, in order to download protocols and exploit resources. Reflection can be combined with mobile code techniques to allow applications select from where to download protocols or where to move computations, based on application-specific information (e.g., trusted hosts, quality of service, etc.).

The combination of the two above mentioned techniques gives strength to our approach as it enables our middleware to cope with both foreseen and unforeseen changes in the context. For example, in conditions of high memory and low bandwidth, a mobile application may wish to replicate remote data locally and work on local copies. In this case, the application uses reflection to instruct the middleware to replicate the data instead of linking it. If the host runs out of memory or the network bandwidth increases, the middleware might have to adapt by removing the replica and instead accessing the data remotely using a different linking protocol. It is obviously not feasible to have all the code needed stored on the device, due to memory limitations. Furthermore, in a mobile ad-hoc setting mobile hosts cannot forecast all the possible conditions they are going to be in, and it would be a very static approach to decide to load all possible behaviours on the device in order to be able to deal with all possible situations. Mobile code offers a reasonable alternative: the middleware checks if the peers in reach have the protocol needed and then downloads it from them, in a very flexible manner. The old protocol for caching can be deleted (as the memory availability is low) and be downloaded again if necessary when the environmental conditions change.

4 Future Directions of Research

The increasing diffusion of new devices and applications for wireless settings calls for the design of mobile computing middleware that fit the new scenario. We believe that a combination of reflection and different mobile code paradigms may enhance the adaptability of mobile computing applications and introduce the required level of dynamism and flexibility into the middleware.

Our research is just at the beginning, although some promising results have already been reached [1, 2]. Many open issues are on our agenda. A first major issue is safety. What happens if two applications ask the middleware to behave differently when executing in the same context? What if the same application requires conflicting behaviors when changes related to different resources happen at the same time (e.g., "disconnect when battery is low" vs. "connect when bandwidth is high")? What if the behaviour requested cannot be found neither locally nor on the hosts in reach? All this questions are currently under investigation.

Another major problem is security. Our current model does not deal with this requirement but we plan to investigate it in the future, in order to enforce access control policies. We also plan to refine our model to handle resource leasing.

References

1. L. Capra, W. Emmerich, and C. Mascolo. Reflective Middleware Solutions for Context-Aware Applications. In *Proceedings of REFLECTION 2001. The Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns*, Kyoto, Japan, September 2001. To Appear.
2. C. Mascolo, L. Capra, and W. Emmerich. XMIDDLE: A Middleware for Ad-hoc Networking. 2001. Submitted for Publication.