Coordination of Mobile Intermediaries Acting on behalf of Mobile Users

Norliza Zaini* and Luc Moreau*
{ nmz00r, L.Moreau }@ecs.soton.ac.uk

Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ UK

Abstract. We introduce the notion of a mobile intermediary, called Shadow, which is a mobile agent located in the network infrastructure, interacting with complex applications on behalf of mobile users. Due to intermittent connectivity, multiple Shadows may simultaneously coexist. In this paper, we introduce a protocol capable of coordinating Shadows and we present an abstraction layer, hiding away communication and coordination details, which offers a substrate to build distributed applications across mobile devices and fixed infrastructure.

1 Introduction

The context of this paper is the "ubiquitous computing environment" [7] where embedded devices and artifacts abound in buildings and homes, and have the ability to sense and interact with devices carried by people in their vicinity. Mobile devices' networking capabilities offer opportunities for a new range of services, such as customised access to news updates or exchange of information with other mobile users discovered dynamically. However, communications between mobile devices and the infrastructure have some limitations, in the form of intermittent connectivity and low bandwidth. Furthermore, processing power and memory capacity of compact mobile devices remain relatively small. As a result, such an environment would prevent the *large scale* deployment of *advanced* services to mobile users, as they tend to be communication and computation intensive.

We believe that applications can be offloaded to the fixed infrastructure, and act semi-autonomously on behalf of the user. Such an approach does not rely on permanent connectivity with mobile devices, which can save device's resources and take advantage of the available resources on the wired network [3]. Here, we introduce an *intermediary process* in the fixed infrastructure, whose responsibility is to spawn applications in reaction to user's requests and to store and forward messages between devices and applications, according to the available connectivity. Our vision is that of a *mobile* intermediary, which is a mobile agent [2], acting as a *Shadow* of the mobile user, migrating to the user's vicinity when

* This research is funded in part by QinetiQ and EPSRC Magnitude project (reference GR/N35816).

prevailing conditions permit it. This benefits from a number of advantages: *(i)* Shadow and mobile device can communicate using specialised protocols, possibly dynamically chosen according to the current location or to a negotiation between parties; *(ii)* newly created applications would run in the user's vicinity, making use of the local infrastructure; *(iii)* local services on a local network could be accessed; *(iv)* Shadows and applications can communicate reliably using transparent routing of messages to mobile agents [4, 5].

When a user moves to a new location, their mobile device will request the Shadow to migrate to a new location. However, this may fail when the local network is not connected with the user's previous location. To support services in the current vicinity, we opted for a solution where new Shadows can be created dynamically. As result, a user may be associated with multiple Shadows that need to be coordinated. The purpose of this paper is to describe the interactions between mobile devices, Shadows, applications and fixed infrastructure. Our specific contributions are: *(i)* An architecture supporting multiple Shadows; *(ii)* A coordination protocol between mobile devices and Shadows; *(iii)* An abstraction layer, encapsulating migration and coordination offering a substrate to program applications directly between mobile devices and fixed infrastructure.

In the next section, we overview the architecture. Then, we present the algorithms to be implemented by all its components. In Section 4, we discuss related work, and describe work in progress on the implementation.

2 Architecture Overview

Our proposed architecture is composed of three major components, namely a mobile device, a Shadow and a Shadow Manager, which we describe with the assumptions we make concerning their communication capabilities. A mobile device has the ability to connect to a network in its vicinity and we assume that it is allocated an address, which can be used by networked entities to communicate with it. Shadow Managers and Shadows are agents that need to run on agent platform which is a runtime environment that is able to perform the tasks of supporting the agents' creation, execution, localization, migration, communication and security control. A Shadow Manager acts as a local daemon in a local network, first contact point of a mobile device with the local network. It is responsible for starting or migrating Shadows on behalf of devices.

A Shadow is a mobile agent, acting as an intermediary between a mobile device and infrastructure applications. Being able to migrate allows it to move "closer" to the mobile device, and to communicate with it using the address allocated by the local network. Shadow functions include: *(i)* to create applications on behalf of the mobile device; *(ii)* to send messages to the applications on behalf of the mobile device; *(iii)* to store and forward messages for the mobile device; *(iv)* to migrate to a location closer to the mobile device, whenever the mobile device changes its location, network connectivity permitting.

Our architecture may be summarised as follows. When connected to a network, a mobile device makes contact with a Shadow Manager, and requests its

Shadows to migrate to the manager's location. In the simplest case, there exists a single Shadow. If successful, the Shadow can start interacting locally with the mobile device after its migration. The Shadow spawns new applications as requested by the device and forwards messages to and from them; in essence, the Shadow acts as a router of messages to the applications. Communications between Shadow and applications are robust to the migration of Shadows, based on a transparent routing algorithm [4, 5]. If migration failed for all Shadows, a new Shadow is spawned locally, and the device keeps a log of all created Shadows. When several Shadows are requested to migrate to a specific destination, the first Shadow to reach the location is assigned to be the "main Shadow"; the others coordinate with it to offload information about applications they were routing messages to.

In the following section, we describe the algorithm of each component. Our goal is to define an abstraction layer, which hides the details of communication and coordination between mobile devices, Shadows and applications. On top of this abstraction layer, we will be able to construct applications involving mobile devices: in the mobile device, a programming API will be provided to communicate transparently with fixed infrastructure applications, while applications will be given the possibility to interact transparently with mobile devices; the abstraction layer takes care of all necessary routing and coordination.

3 The Algorithm

In this section, we describe the algorithm coordinating the interactions between mobile devices, Shadows, Shadow Managers and applications.

Mobile Device When connected to a network, a mobile device sends a "MigrateRequest" message to a discovered Shadow Manager requesting its Shadows to migrate "closer" to its current location. Then, if it receives a "ShadowInformation" message, it sets the sender as the main Shadow by sending an "MSAssignment" message. To each subsequent "ShadowInformation" message received, the sender is notified about the current main Shadow by using an "MSInformation" message. The application layer on a mobile device may request an application to be created or a message to be sent to a particular application on the fixed infrastructure. This request would be forwarded to the main Shadow.

A mobile device may receive "TerminationMessage" from the main Shadow which informs about a Shadow that has recently terminated. On every attempt to send message to a Shadow, a failure handler is provided which adds any message failed to be sent to a queue of outgoing messages. In parallel to other activities, messages from the queue of outgoing messages are sent to the respective receivers. On failure the messages are added back to the queue.

Shadow Manager When a Shadow Manager is started, it advertises its presence through e.g. Jini or LDAP, and then waits for messages. A Shadow Manager may receive a request from a mobile device to migrate Shadows; the Shadow

Manager then sends a "MigrateRequest" message to all Shadows requesting them to migrate to the platform on which it is operating. If no Shadow was able to migrate, it starts a new Shadow to which an "MDInformation" message which contains information on the requesting mobile device is sent.

Shadow In a Shadow, there is a hook for intelligent decision making about migration, in which the output of this decision making process is obtained by the "callback" canMigrate(), which returns true if the application layer decides to migrate. A Shadow may create application on behalf of mobile device. Each application has an identifier and an address. The application identifier to application address mappings are placed in a list (LAM). Messages failed to be sent to the mobile device are added to the list of outgoing messages (LOM), while messages failed to be sent to applications are added to a list of incoming messages (LIM). In parallel to other activities, messages from these two lists are attempted to be sent to the respective receivers.

On creation, a Shadow waits for an "MDInformation" message, which contains information about a mobile device. Then the Shadow sends a "ShadowInformation" message to the mobile device. If it receives an "MSAssignment" message, it is assigned to be the main Shadow and responsible for sending "LocationInformation" messages to all other active Shadows of the device. The message indicates current location of the mobile device. If a Shadow receives an "MSInformation" or a "LocationInformation", another Shadow is acting as the main Shadow and it has to handover its function to the main Shadow by transferring its LAM, LOM and LIM. Then the Shadow informs all applications it is interacting with, that the main Shadow is the new intermediary to communicate with the mobile device. When this is done, the Shadow is ready for termination; before terminating itself, it sends a "TerminationMessage" to the main Shadow.

A Shadow may receive a "MigrateRequest" from a Shadow Manager. If canMigrate() returns true, it migrates to the platform on which the Shadow Manager is running. On arrival at the new platform, it sends a "ShadowInformation" message to the mobile device. Otherwise, the Shadow stays on the same platform and may receive a "LocationInformation" from the main Shadow. As usual, on receiving this message, a Shadow has to hand over its function to the main Shadow before terminating itself.

A main Shadow is expected to receive LAM, LIM and LOM from other Shadows. When received, these lists are extended to the Shadow's local lists. The main Shadow may also receive a "TerminationMessage" which requires it to relay this message to the mobile device indicating the termination of a Shadow. As for messages coming from the mobile device, a main Shadow may receive requests to create application or send message to an application on the fixed infrastructure. An application is started according to the type and identifier included in the "CreateApplication" request, while its identifier to address mapping is added to LAM. A "SendMessage" request includes identifier of an application to which a message should be forwarded. Before sending the message, the application address is extracted from LAM. If the Shadow failed to create application or send a message, a failure notification is returned to the mobile device.

4 Discussion and Related Work

This paper has focused on the coordination of multiple Shadows, and on the communication between devices and Shadows. We have not given details on how applications can communicate with Shadows: this issue has been studied extensively in previous papers, for instance by using a message routing algorithm for mobile agents [4]. In our approach, we wish to promote the flexibility of the system, by allowing multiple Shadows to be created, according to the prevailing network conditions, and by allowing Shadows to make intelligent decisions as whether to migrate. The handover of function of a Shadow to the main Shadow shortcuts chains of forwarding pointers when the Shadow does not migrate.

There are other projects applying mobile agent technology to support applications for mobile users. In TACOMA on PDAs [1] and MobiAgent [6], there exists a stationary proxy communicating with mobile devices and mobile agents. This offers less flexibility than ours as it requires a connectivity to exist to the stationary proxy regardless of the distance.

As far as implementation of this system is concerned, we are currently prototyping the coordination algorithm. The transparent routing of messages to mobile agents is already available in the Southampton Framework for Agent Research (SoFAR). In this coordination algorithm, we have not yet considered robustness to failures: in complement to [5], we would like to introduce some redundancy to become tolerant to failures of intermediary nodes. We are planning to develop two applications on this abstraction layer: an application to share documents with mobile users in virtual meeting rooms, and a virtual briefing room, where documents from multiple (mobile) sources are collated and presented to the user.

References

1. Kjetil Jacobsen and Dag Johansen. Mobile software on mobile hardware – experiences with tacoma on pdas. Technical Report 97-32, Department of Computer Science,University of Troms, Norway, 1997.
2. Danny B. Lange and Mitsuru Ishima. *Programming and Deploying Java Mobile Agents with Aglets.* Addison-Wesley, 1998.
3. Patrik Mihailescu and Walter Binder. A Mobile Agent Framework for M-Commerce. *Computer Science 2001*, GI/OCG annual Convention:2:959–967. .
4. Luc Moreau. Distributed Directory Service and Message Router for Mobile Agents. *Science of Computer Programming*, 39(2–3):249–272, 2001.
5. Luc Moreau. A Fault-Tolerant Directory Service for Mobile Agents based on Forwarding Pointers. In *The 17th ACM Symposium on Applied Computing (SAC'2002) — Track on Agents, Interactions, Mobility and Systems*, Madrid, March 2002.
6. Mahmoud Q.H. MobiAgent – An Agent-based Approach to Wireless Information Systems. In *Proceedings of the 3rd International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2001)*, Montreal, 2001.
7. Mark Weiser. Some Computer Science Problems in Ubiquitous Computing. *Communications of the ACM*, 36(7):74–84, July 1993.