Benefits and Requirements of Using Multi-agent Systems on Smart Devices

Cosmin Carabelea[1], Olivier Boissier[1], and Fano Ramparany[2]

[1] SMA/SIMMO/ENS Mines de Saint-Etienne, France
{cosmin.carabelea, olivier.boissier}@emse.fr
[2] DIH/OCF/France Télécom R&D
Fano.Ramparany@rd.francetelecom.com

Abstract. Due to the emergence of Internet and embedded computing, humans will be more and more faced to an increasing intrusion of computing in their day-to-day life by what it is called now *smart devices*. Agent characteristics like pro-activeness, autonomy and sociability and the inherent distributed nature of multi-agent systems make them a promising tool to use in the smart devices applications. This paper tries to illustrate the benefits of using multi-agent systems on smart communicating objects and discusses the need of using multi-agent platforms. We then present an overview of multi-agent platforms created for use on small devices (i.e. devices with limited computing power) and for each reviewed multi-agent platform we try to identify what are its main characteristics and architectural concepts.

1 Introduction

In the near future it is to be expected that humans will be more and more faced to an increasing intrusion of computing (in the form of *smart devices*) in their day-to-day life. The term *smart device* designates any physical object associated with computing resources and able to communicate with other similar objects via any physical transmission medium, and logical protocol, or with humans via standard user interface. The scale spans from big smart devices such as PDAs, to small ones such as RFID Tags.

We are mainly interested in devices that share the following characteristics: tight memory constraints, limited computing power (due to low power consumption), limited user interface peripheral, embedded in the physical world and exhibiting real time constraints. The increasing number of such devices makes us envision a myriad of applications in which users will shortly need to be intelligently assisted in their interactions with these computing entities. The evolution surely will produce several users in interaction with several devices, so we need to introduce in these intelligent objects some cooperating and social reasoning capabilities.

For many years, multi-agent systems have proposed a new paradigm of computation based on cooperation and autonomy. An intelligent agent is "a computer

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 1091–1098, 2003.

system, situated in some environment, that is capable of flexible and autonomous action in order to meet its design objectives" [8]. A multi-agent system is a federation of software agents interacting in a shared environment, that cooperate and coordinate their actions given their own goals and plans. Many properties of this metaphor of computing make them attractive for tackling with the requirements presented above:

- *Pro-activity*: agents can pro-actively assist their users and try to achieve goals given the evolution of the environment; they can help them in discovering information, in interacting.
- *(Adjustable) autonomy*: agents are able to try to achieve their own goals, but are also able to take into-account and adapt their autonomy according to their users' constraints.
- *Self-awareness*: agents are conscious of themselves in the sense that they have a model of their goals, of their own competences, of their state.

Although the multi-agent approach may seem suitable for the smart communicating objects, one has to solve the problems of the actual deployment of agents on those small devices. This paper tries to identify the requirements for executing agents on such devices (operating system, communication facilities, computational power, etc.) and, in the meantime, we try to identify the constraints that the use of small devices is imposing to the multi-agent architectures and services.

The remainder of the paper is structured as follows. In the next section we will present our arguments on why the utilization of multi-agent systems on smart devices is benefical. Section 3 describes what a multi-agent platform is and what are the FIPA standards for those platforms; we then present some of the main characteristics of several multi-agent platforms for small devices. Finally, in Section 5 the utility of using multi-agent systems and platforms on small devices is discussed and we are drawing some conclusions and tracing directions for future work.

2 Why Use Multi-agent Systems on Smart Devices?

Pervasive and ubiquitous computing will involve more and more objects endowed with computing and communication ressources, called smart devices. These smart devices should be able to sense their environment and react to changes in it. In order to alleviate the user information overload, it will be necessary to enable these objects to communicate directly with one another, while in the same time they should be able to communicate with the user.

Due to the high number of these devices, we will have to reduce the number of objects needing explicit control from the user, thus making these devices autonomous. Another way of simplifying the human – smart device interaction is to provide smart devices with knowledge and reasoning capabilities, so that most of the repetitive tasks involved in the interaction could be automated.

An intelligent agent is "a computer system, situated in some environment, that is capable of flexible and autonomous action in order to meet its design objectives" [8]. Beside the (adjustable) autonomy, an agent has other properties. It should be able to

detect and react to changes in its environment. Agents can pro-actively assist their users and try to achieve goals given the evolution of the environment; they can help them in discovering information, in interacting. We can have several agents acting in a common environment, thus obtaining what is called a multi-agent system. From the multi-agent point of view, an agent should be able to interact with other agents, i.e., communicate, cooperate or negotiate with them. Also, an agent should be able to reason about the existing organizational structures in the system.

It is no surprise that the multi-agent applications try to solve the same problems as the applications on smart devices. Some of these problems are concerned on how to realize the interaction between the diverse entities in the system, while others on how the agents or the smart devices should adapt themselves to new situations. The smart devices environment is a very dynamic and open one, devices can enter or leave it at any time. Multi-agent systems research is also interested in the modelization of open systems: what check-in/check-out procedures are needed, how to ensure the service discovery in an open system and so on. The problem of trust and reputation is also tackled in both domains.

It is clear for us that intelligent agents have characteristics desirable for smart devices. Moreover, the multi-agent systems already solve or try to solve similar problems with those of smart devices, so one can beneficiate from these solutions. Of course, an application for smart devices can be created without the multi-agent technology, but by using it, one can take advantage of the methodologies and solutions provided by the multi-agent paradigm. And we should not forgot the reusability: a multi-agent system created for a specific problem can be very easily used to solve another one. In other words, using agents on smart devices is not the the only perspective of solving the problem, but it might prove better than others.

3 How to Use Multi-agent Systems on Small Devices?

A *multi-agent platform* is a software infrastructure used as an environment for agents' deployment and execution. It should provide comfortable ways for agent programmers to create and test agents and it can be viewed as a collection of services offered to developers, but also to agents at runtime.

As depicted in the previous characterization, the platform is an execution environment for the agents in the sense that it should allow one to create, execute and delete agents. Secondly, the platform should act as a middleware between the operating system and the applications (agents) running on it, as depicted in Fig.1. The developer will then be able to create agents for a platform and use them on all the systems that support the platform without changing the code.

More than that, a platform should hide from the developer the communication details between the agents. If, for example, there are several communication protocols available on the device (e.g. HTTP, RMI, Bluetooth, etc.), the platform should choose the appropriate protocol for each situation and use it. This will allow one to create agents that are independent of the communication protocols available on the machine, leaving the actual sending of the information as a task for the platform.

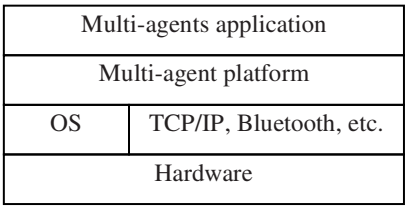

Fig. 1. Coarse-grained view of a device executing an agent platform and an agent application

Because the choice of an agent platform has a great influence on the way the agents are designed and implemented, FIPA[1] has produced standards that concern how an agent platform should be structured [6], i.e. what are the basic services it should offer (e.g. a message transport service, an agent management service, a directory service). These standards exist to ensure a uniform design of the agents, independent of the multi-agent platform. Thanks to it, for instance, at the moment several agent platforms are deployed in several cities of the world on which agents can go and execute and cooperate with other agents [1].

In the next chapter we will give a brief description of several agent platforms designed for small (and wireless) devices, on which the limitations of computing resources have a great influence on the complexity of the applications. For each platform we will try to put in evidence the targeted devices and the wireless technologies it works with, as well as if it is FIPA-compliant or not and what are its most relevant characteristics. All the platforms we have investigated are Java-based, so we will try to specify for each one what is the Java Virtual Machine (VM) it needs.

4 Multi-agent Platforms for Small Devices

Due to the limited resources available, the agent platforms for the small devices need to have a light architecture. Most of these platforms are designed to run completely (i.e. with all the services they provide) on the small device, while others are keeping some of the services (or sometimes all) on a server and they are downloaded on the device only when needed. From this point of view, we can classify existing platforms in three categories replicating the conceptual classification presented in [14]:

- *Portal platforms* that don't execute the agents on the small devices, but on other hosts. The small device is used only as an interface with the user.
- *Embedded platforms* are executed entirely on the small device(s), together with one or several agents.
- *Surrogate platforms* that are executing only partially on the small device, while some part of them is executed on other hosts.

[1] Non-profit organization aimed at producing standards for the interoperation of heterogeneous software agents. *http://www.fipa.org*

In what follows, we will present the basic characteristics of several platforms from all categories.

4.1 Portal Platforms – MobiAgent

The MobiAgent [13] system architecture consists of three main components: a handheld mobile wireless device, an Agent Gateway and the network resources. The Agent Gateway is executing an agent platform and the agents for that platform. When the user wants to delegate a task to an agent, the mobile device connects to the gateway and *downloads an interface* that will configure the agent for her. The agent will perform its task and it will later report the results via the same mechanism.

Note that neither the platform, neither the agents are running on the device, the only application running there is a MIDlet that configures the agent. This approach does not impose anything to the platform or to the agents running on the Agent Gateway; the platform running there may be FIPA-compliant, but this is not mandatory. This infrastructure is available on any device supporting J2ME/CLDC/MIDP, including mobile phones. The connection between the device and the gateway is done via HTTP over a wireless network. Unfortunately, no links for downloading the platform are currently available.

4.2 Surrogate Platforms – kSaci

kSACI [9] is a smaller version of the SACI platform [15] suited to the kVM. SACI is an infrastructure for creating agents that are able to communicate using KQML [10] messages. Each SACI agent has a mailbox to exchange messages with the others. Its architecture also contains a special agent, called the *facilitator*, offering the white- and yellow-pages services of the system. The architecture of the platform is not FIPA-compliant.

The kSACI platform is usable on small devices running the kVM, but the platform is not entirely situated on the small device. In fact, the facilitator is on a server and on every small device there is only one agent running. This agent should contain a mailbox used to exchange messages, but it doesn't know how to pass messages to the others; it communicates via HTTP with a proxy running on a desktop machine that further passes the messages to and from other agents. This solution makes the agent lighter, so this platform can be used with devices as small as a mobile phone or a two-way pager.

4.3 Embedded Platforms

Several multi-agent platforms exist. Among them few are suited to be embedded on small devices. We have investigated four of them, but, due to space limitations, only two are presented here. The interested reader is invited to [4] for more information about the two other platforms, MAE and microFIPA-OS.

4.3.1 AgentLight

AgentLight [2] is an agent platform with the objectives of being capable of deployment on fixed and mobile devices with various operating systems and that can operate over both a fixed and a wireless network. Its authors are using a set of profiles that allows one to configure the platform for several devices and/or operating systems. Also, this platform aims to be FIPA-compliant and operating system agnostic.

The architecture of AgentLight is called "half-daisy" and consists of an *agent container* running on each device on which several agents can be placed. If the two agents on the same device communicate with each other, the agent container does the communication internally, but if the agents are on separate devices, the agent container acts as a proxy.

Each agent is seen as a set of rules and it is using for execution an inference engine provided by the agent container. This architecture is not entirely FIPA-compliant (e.g. the directory service is missing) and it is still work in progress. It can be run on devices with J2ME/CLDC/MIPD and with the kVM and the smallest device targeted is a mobile phone, although for the moment there are not any tests to prove a mobile phone can support the platform, especially the inference engine that can be quite heavy at the execution time.

4.3.2 LEAP

The Lightweight Extensible Agent Platform (LEAP 3.0) is probably the most known agent platform for small devices. It is the result of the LEAP project [12] that had the objective of creating a entirely FIPA compliant platform with similar objectives to the AgentLight Platform: be capable of deployment on fixed and mobile devices with various operating systems and deployed in wired or wireless networks. Since the last version, 3.0, LEAP is an add-on of the JADE platform [7] and it uses a set of profiles that allows one to configure it for execution on various machines, OS and Java VM.

The architecture of the platform is modular and is organized in terms of modules among which some can be mandatory, requested by FIPA specifications, while other are optional. The mandatory ones are the *kernel module* that manages the platform and the lifecycle of agents and the *communication module*, which handles the heterogeneity of communication protocols.

Some of the modules are device-dependent (e.g. the communication module), while others are independent of the characteristics of the device (e.g. a security plug-in for the communication module). The platform is operating system agnostic and it can run on devices ranging from mobile phones and PDAs to workstations. It can work over wireless networks that support TCP/IP, like WLAN or GPRS.

As in AgentLight, the architecture of LEAP splits the platform in several containers, one for every device/workstation used, with one or several agents in each container. These containers are responsible for passing the messages between agents by choosing the appropriate communication protocol available. One of these containers has a special role and it is called *main-container*; it contains two special agents that implement the FIPA specifications to provide the white and yellow pages services. This main-container cannot be executed on a small device, but only on a PC. In order to allow the execution of containers even on very limited devices, LEAP has

two execution modes: stand-alone and split. In the former, the container is completely embedded on the small device, while in the former, a part of the container is executed on a PC.

Table 1. Characteristics of multi-agent platforms for small devices

Platform	MobiAgent	kSACI	AgentLight	LEAP	MAE	micro FIPA-OS
Connection to SD	portal	surrogate	embedded	surrogate / embedded	embedded	embedded
Smallest targeted device	mobile phone	mobile phone	mobile phone	mobile phone	PDA	PocketPC
FIPA-compliant	it can be	no	yes (?)	yes	no	yes
No. of agents on device	0	1	several	several (pref. 1)	several	several (pref. 1)
Available for download	no	yes	yes	yes	no	yes
JavaVM	kVM	kVM	kVM	various	various	PersonalJava

5 Discussion and Conclusions

In this article we have presented some benefits of using multi-agent systems with the smart devices. But this usage of multi-agent systems on small devices has its drawbacks, i.e. the need for an existing infrastructure that will allow the creation and execution of agents and the communications between them: a multi-agent platform. We have presented a set of multi-agent platforms for small devices, underlining some of their major characteristics and architectural concepts. We have used the three main directions proposed in [14] to bind smart devices and multi-agent platforms: portal, surrogate and embedded. It was beyond the scope of this paper to identify one of these platforms as being the best. We believe there doesn't exist such a platform, but each one of them is suited for different applications. Table 1 summarizes the most important characteristics we have identified for each platform.

As future work we intend to set up a benchmark to evaluate these platforms within several scenarios; the results obtained will then allow us to analyse the platforms' performances in different situations and what is the real amount of computational resources (processing power, memory size) needed by each one. We have begun these practical tests by deploying LEAP on a Siemens S55 mobile phone. So far, the results are encouraging: the phone supports the execution of the platform and of an agent, although with a low execution speed.

When using a multi-agent platform on a small device, there is a high risk that the device resources will not be enough for both the platform and the agent(s) running on

it. Although the use of a platform can greatly reduce the developing cost and increase portability, it is clear that a trade-off should be found between the services the platform offers and its size. One might eventually arrive in the situation of using a small device like a mobile phone and a platform designed to work on it (like LEAP) and to discover that the platform is too big for the device's capabilities or there is no room left for any agents [3].

As for the FIPA-compliancy, some projects such as Extrovert-Gadgets [5] for instance, question its utility for multi-agent applications on small devices. The advantage of compelling to some standards is clear, but these standards have not been designed for the particular characteristics of the small and wireless devices. A good example is presented in [11] where the LEAP platform is used, but some of its mandatory modules, which make it FIPA-compliant, are useless for some applications (e.g. ad-hoc networks).

References

1. AgentCities: http://www.agentcities.org
2. AgentLight – Platform for Lightweight Agents: http://www.agentlight.org
3. Berger, M. et al.: Porting Distributed Agent-Middleware to Small Mobile Devices. In Proc. of the Ubiquitous Computing Workshop, Bologna, Italy (2002)
4. Carabelea, C., Boissier, B.: Multi-agent platforms for smart devices: dream or reality?. In Proc. of the Smart Objects Conference (SOC'03), Grenoble, France (2003), p. 126-129.
5. E-Gadgets: http://www.extrovert-gadgets.net
6. FIPA Abstract Architecture Spec.: http://www.fipa.org/repository/architecturespecs.html
7. JADE – Java Agents DEvelopment Framework: http://jade.cselt.it
8. Jennings, N.R., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. Int. J. of Autonomous Agents and Multi-Agent Systems 1 (1) (1998), p. 7–38
9. kSACI: http://www.cesar.org.br/~rla2/ksaci/
10. Labrou, Y., Finin, T.: A Semantics approach for KQML – A General Purpose Communication Language for Software Agents. In Proc. of Int. Conf. on Information and Knowledge Management (1994)
11. Lawrence, J.: LEAP into Ad-Hoc Networks. In Proc. of the Ubiquitous Computing Workshop, Bologna, Italy (2002)
12. LEAP – the Lightweight Extensible Agent Platform: http://leap.crm-paris.com
13. Mahmoud, Q.H.: MobiAgent: An Agent-based Approach to Wireless Information Systems. In Proc. of the 3rd Int. Bi-Conference Workshop on Agent-Oriented Information Systems, Montreal,Canada (2001)
14. Ramparano, F., Boissier, O.: Smart Devices Embedding Multi-agent Technologies for a Pro-active World. In Proc. of the Ubiquitous Computing Workshop, Bologna, Italy (2002)
15. SACI – Simple Agent Communication Infrastructure: http://www.lti.pcs.usp.br/saci/