Small Mobile Agent Platforms

Extended Abstract

Niranjan Suri, Marco Carvalho, Robert Bradshaw, and Jeffrey M. Bradshaw
Institute for Human & Machine Cognition - University of West Florida
40 S. Alcaniz St. - Pensacola, FL 32501 USA
{nsuri,mcarvalho,rbradshaw,jbradshaw}@ai.uwf.edu

1. INTRODUCTION

Mobile agents can benefit small devices such as PDAs and cellular phones in numerous ways. They can be used to download customized, context-sensitive services to small devices on demand [1]. Mobile agents can also help alleviate bandwidth limitations and support disconnected operation, both significant problems in wireless and mobile environments. However, to achieve these capabilities, small devices must be able to support mobile agents. This paper describes our efforts to date on building small mobile agent platforms. Note that our approaches are based on the Java language and architecture. While it is certainly possible to use other languages for mobile agents, we chose Java for its suitability to mobile agents, its popularity, as well as due to our prior work on Java-based mobile agent platforms.[2]

2. REQUIREMENTS AND CHALLENGES

Mobile agent platforms have several requirements: Platform independence, authentication, secure execution, dynamic class loading, network connectivity, and resource control are some important ones. Depending on the nature of the application, other capabilities such as state capture (for strong mobility) could also be important requirements. Satisfying these requirements on small portable devices raises several challenges.

The size and weight restrictions have a direct influence on battery capacity, processing power, network connectivity, memory capacity, persistent storage, and display capabilities. Most mobile agent platforms, including ours, use an interpreted execution environment like Java. Interpreted execution adds a performance penalty that is amplified by the low processing power available on small devices.

Another challenge is dynamic class loading. While the Java VMs that comply with the Java 2 Standard Edition specification provide built-in support for dynamic class loading, the Java 2 Micro Edition VMs (including the KVM from Sun) do not provide this capability.

Limited network bandwidth and connectivity is actually not a challenge, but a requirement satisfied by mobile agents. Mobile agents can, in some circumstances [3], reduce network bandwidth usage and support disconnected operation. However, small memory and persistent storage capacities place additional demands on bandwidth and connectivity. For example, strategies such as class caching cannot be used resulting in multiple transfers of the agent code.

Secure execution is yet another challenge. Security is divided into two categories: protecting the platform from malicious agents and protecting agents from a malicious platform. The former can be achieved through sandboxing (already available in Java) and resource control mechanisms (which are not yet available). The latter, protecting agents from malicious platforms, is problematic. There are no comprehensive solutions to this problem. Moreover, the solutions that have been proposed to date are computationally expensive, or make other assumptions such as availability of trusted hosts, etc. that make them impractical for small devices (see [4] and [5] for a comparison and critique of many existing solutions).

It is worth noting that device capabilities and battery capacities are constantly improving, which may in due course alleviate some of these challenges. However, wireless network bandwidth is not improving at the same rapid rate as wired bandwidth. There is a 3:1 ratio in improvements of wired versus wireless bandwidth. For example, even in UMTS (which provides a 2.0 Mbps theoretical bandwidth), the bandwidth is shared amongst the users of the cell, thereby introducing significant variability in the available bandwidth.

3. DESIGN

We are pursuing two different approaches to developing small mobile agent platforms. The first approach is targeted towards platforms such as PocketPC, which can run the PersonalJava VM The second approach is targeted towards platforms such as the Palm OS and mobile phones, which can run the K Virtual Machine.

3.1 Personal Java-based Platform

PersonalJava™ was initially introduced by SUN as the Java™ technology for connected portable devices. It is comprised of an optimized Java Virtual Machine and a set of extended java libraries specifically designed for small portable devices such as PDA's and web-phones. One of its primary design goals was to reduce static memory footprint of the virtual machine as explained in detail by SUN.

A Java based mobile agent system designed to run on small devices must comply with its limited capabilities and yet be flexible enough to allow agents to securely operate and move between hosts. Our initial approach was to provide a hybrid agent system that would rely on the standard capabilities of the PersonalJava™ platform while running on small devices and still take full advantage of the system resources and J2SE capabilities after moving back to computers or servers. This capability is available in the NOMADS agent system.

The NOMADS agent system [2] comprises two distinct and yet integrated implementations of a common API. The system

comprises its own java virtual machine and two execution environments with different capabilities and characteristics.

The Oasis execution environment provides NOMADS with strong mobility and resource control capabilities. Oasis instantiates its own virtual machine (Aroma), designed to enforce resource control and strong mobility. At this moment, Aroma is compatible with JDK 1.2.2 but it is being ported to support JDK1.3.1 as well.

Spring is a java only execution environment implemented for NOMADS. It is cross-platform and fully compatible with Oasis. Spring relies on the standard java classes and JVM. It supports only weak mobility and cannot provide resource control. It was designed to allow great portability and interoperability.

One of the great advantages of this approach for small devices is that NOMADS agents can move seamlessly between the two execution environments. Agents can move between Oasis and Spring environments using weak mobility and still use strong mobility to travel between Oasis environments. Such capability alone provides great flexibility to the agents and extended portability for NOMADS.

Since all communication and transport mechanisms implemented in Spring are based on TCP sockets and object serialization, the porting of the execution environment from J2SE to PersonalJava™ was easily and successfully achieved without major code modification.

NOMADS agents running on small devices have complete access to all the PersonalJava™ classes and are still bound to the security policies defined. The agents can use graphical resources based on the AWT libraries, network resources and even advances features such as reflection and remote method invocation (RMI).

Preliminary tests have indicated that such approach is advantageous and allows the agent to identify and adapt to constrains of the environment (through the execution environment in which it is running).

One of the drawbacks observed was that the resources required by the PersonalJava™ platform would still impose strong limitations on the devices that could be used to run Spring. We could successfully run the system on small devices like HP Jornada Palm PC with only 16MB of RAM. It is expected though that lower end devices would present performance issues. Besides that, the results obtained by using PersonalJava™ were very encouraging and worked as a proof of concept to this initial approach.

After the announcement of Java 2 Micro Edition (J2ME) with its scalable design based on the use of customized profiles, SUN has created a Java Specification Request (JSR62) aimed to replace PersonalJava™ by a *personal profile* in J2ME. This personal profile, as presented on the JSR-62 will target the same platforms initially intended for the PersonalJava™ with minimum memory requirement of 1MB, using the KVM instead of the PersonalJava 1.1 Virtual Machine. The K Virtual Machine from SUN is designed to operate on very small memory devices such as mobile phones.

3.2 Aroma-based Platform

The Aroma VM is a clean-room implementation of the Java Virtual Machine specification. Aroma provides two key capabilities over standard Java VMs: state capture and resource control. State capture allows the execution state of Java threads to be captured and moved from one instance to another instance of Aroma. This capability can be used to provide strong and forced mobility. Resource control allows limits to be placed on the rate and quantity of resources used by threads (including CPU, disk, and network). The Aroma VM has already been ported to the Win32 platform and Linux and Solaris UNIX platforms. We are now in the process of porting Aroma VM to the Windows PocketPC platform. Once this is completed, Oasis will be able to operate on PocketPC platforms, providing an alternative to Spring and PersonalJava. Oasis will provide strong mobility and resource control on small devices also.

4. SUMMARY AND FUTURE WORK

Small devices can benefit from mobile agent technologies in several ways. However, they present several challenges that need to be addressed. We are currently in the process of developing small mobile agent platforms for such devices. Currently, we have an implementation that runs on PersonalJava. We are continuing to port the Aroma VM and the Oasis mobile agent environment to the PocketPC environment as well as develop a KVM-based implementation of the mobile agent platform.

5. REFERNCES

[1] Adler, M. Bradshaw, J.M., Mahan, M., and Suri, N. Applying Mobile Agents to Enable Dynamic, Context-Aware Interactions for Mobile Phone Users. In *Proceedings of the Third International Workshop on Mobile Agents for Telecommunications Applications (MATA 2001).* Lecture Notes in Computer Science, Vol. 2164. SPRINGer, 2001.

[2] Suri, N., Bradshaw, J.M., Breedy, M.R., Groth, P.T., Hill, G.A., & Jeffers, R. (2000). Strong Mobility and Fine-Grained Resource Control in NOMADS. *Proceedings of the 2nd International Symposium on Agents Systems and Applications and the 4th International Symposium on Mobile Agents (ASA/MA 2000).* Zurich, Switzerland

[3] Gray, R.S., Kotz, D., Peterson, R.A. Jr., Gerken, P., Hofmann, M., Chacon, D., Hill, G., and Suri, N. Mobile-Agent versus Client/Server Performance: Scalability in an Information-Retrieval Task. Dartmouth College Computer Science Technical Report TR2001-386.

[4] Knoll, G., Suri, N., Bradshaw, J.M., Path-based Security for Mobile Agents. First International Workshop on Security in Mobile and Multi-Agent Systems (SEMAS 2001). Montreal, Canada 2001.

[5] Jansen, W. and Karygiannis, T. Mobile Agent Security. NIST Technical Report. 1999.

[6] Suri, N. Bradshaw, J.M., Breedy, M.R., Ford, K.M., Groth, T., Hill, G.A., and Saavedra, R. State Capture and Resource Control for Java: The Design and Implementation of the Aroma Virtual Machine. White Paper