The Joint Symposium
ASA/MA2000

ETH Eidgenössische Technische Hochschule Zürich

Abstracts

Poster

Contents

1 **Mobile Agent: Enriching Document Management and Distribution for Mobile Design Work**
Mark Allen, Geof Staniford, and A. Taleb-Bendiab

2 **Fault-Tolerant Mobile Agents in Mozart**
Ilies Alouini and Peter Van Roy

3 **A Knowledge-based Internet Agent System with a Formal Verification Facility**
Tadashi Araragi

4 **Active Networking, QoS and Virtual Routers**
Florian Baumgartner and Torsten Braun

5 **Experiences with State-of-the-Art Migration Strategies**
Peter Braun, Christian Erfurth, and Wilhelm Rossak

6 **MobiliTools: A Toolbox for Agent Mobility and Interoperability Based on OMG Standards**
Bruno Dillenseger

7 **Internet Service Delivery Control with Mobile Agents**
Manuel Günter and Torsten Braun

8 **Agent-based Virtual Laboratory**
Goran Kimovski, Danco Davcev, and Vladimir Trajkovic

9 **Application Centric Mobile Agent Systems: Bringing the Focus Back to the Applications**
Paulo Jorge Marques, Luís Moura Silva, and João Gabriel Silva

10 **Xmile: An Incremental Code Mobility System based on XML Technologies**
Cecilia Mascolo, Wolfgang Emmerich, and Anthony Finkelstein

11 **Mobile Agent Platform for Mobile Devices**
Patrik Mihailescu, Elizabeth A. Kendall, and Yuliang Zheng

12 **Simulating Mobile Agent Based Network Management using Network Simulator**
Otto Wittner and Bjarne E. Helvik

Thursday, September 14
16:45 - 18:15

Mobile Agent Platform for Mobile Devices

Patrik Mihailescu, Elizabeth A. Kendall and Yuliang Zheng

Peninsula School of Network Computing
Monash University, McMahons Road, Frankston, VIC 3199, Australia
patrik77@bigpond.com{kendall,yuliang.zheng}@infotech.monash.edu.au

1 Extended Abstract

Up until now, mobile agent platforms (e.g. Aglets, Mole, Concordia, etc) have been confined to operate within high-end desktop environments such as Windows, Unix and Solaris. This poster paper presents work in progress in the development of a mobile agent platform for small mobile devices such as cell phones, pagers, home appliances and personal digital assistants (PDA). These devices are becoming more and more popular with users due to their small size and their ability to be used while the user is on go. We believe that these devices will benefit greatly from the use of mobile agent technology. Typically these devices are connected via a wireless network. Wireless networks, compared to wireline networks suffer from lower bandwidth, have a greater tendency for network errors and have a higher cost of network connectivity. Mobile agents will not only help to overcome these network limitations, but they will also help to enhance the level of functionality offered by applications on these devices. For instance, applications will become more independent and able to operate without constant user input. Applications may evolve during their lifetime, i.e. agent locates additional application component and installs them.

We are in the process of developing a mobile agent platform for mobile devices using the Connected Limited Device Configuration (CLDC) specifications, which is part of the Java 2 Micro Edition (J2ME). The CLDC specification is aimed for devices, which contain around 160KB to 512KB of memory and operate with either a 16 or 32 bit RISC/CISC microprocessor. Devices such as cell phones, pagers, personal organizers and point of sales devices fall into the CLDC specification. There are numerous differences between the J2ME and the Java 2 Standard Edition (J2SE)[1]. One of the strengths of the J2ME is its flexible architecture, which allows it to operate on a large number of technically diverse devices. This is achieved through the use of profiles, which allows a particular family of devices (cell phones) to define common functionality required by the underlying Virtual Machine (VM). Therefore different groups of devices (even toasters) may define different profiles containing functionality that is only appropriate for a particular group of devices.

The CLDC specification also utilizes a new VM called the KVM (Kilo Virtual Machine). This has been designed to operate with as little as 128k memory, which includes the Virtual Machine (VM), the class libraries and heap memory for executing

[1] The whitepaper located at http://java.sun.com/products/cldc/wp/KVMwp.pdf covers these differences

applications. We have decided to embed our agent platform directly into the KVM, as opposed to defining additional Java classes (on top of the exiting KVM classes), which will need to be resolved by the KVM when called. Our approach is in contrast to the approach taken by existing mobile agent platforms. The advantages for us are significant, as our classes will be loaded directly into memory when the KVM loads for the first time. Therefore any access to our classes will be substantially quicker as they will no longer need to be interpreted; they are already in native C code. This saving is crucial as the majority of our intended devices operate with relatively low memory/processing/power and any additional translation of Java classes to native methods will cause notable performance problems. To embed our classes within the KVM, we had to rebuild the KVM. We used the romizing approach, by performing all the linking and resolving of our classes using a program known as the JavaCodeCompact. This program will accept as input all our Java classes and after its execution will produce a C representation of every single one of our Java classes. This is used in conjunction with the original KVM C source code and compiled/linked to produce a mobile agent enabled KVM.

The makeup of our agent platform contains several common features located in existing mobile agent platforms, e.g. mobile agents, messaging, events, etc. However there are also significant differences, as we have aimed to provide specific functionality for the intended devices that use our KVM. This functionality falls into four main areas: XML, security, networking and agent tracking. The majority of our classes are written in Java and then translated to C using the JavaCodeCompact program. However we have been forced to write several native methods in particular for our security routines, as the KVM does not provide any Java security classes or any Java math classes. The XML support we are providing is a SAX based non-validating parser, which can be used to interpret XML based messages. As not all devices will have a TCP/IP stack installed, we are providing support for a variety of protocols for all network functions (messages, sending/receiving of agents) such as the IrDA protocol, Bluetooth, Serial and TCP/IP. As the KVM does not provide any security, we are implementing a security scheme based on public key techniques called Signcryption. Finally we are providing support for tracking of agents, with particular emphasis on making sure agents can return safely back to their original location.

At the moment, we are finalizing our Java classes, making sure they match our functional specifications. The next stage of our work will be the development of several agent-based applications, which will help to evaluate and identify both strengths and weaknesses in our architecture. Our agent platform is being tested on a variety of Palm device (Palm Vx, Palm IIIx, Palm IIIc)[2], but once Motorola release their KVM for cell phones we will also be including these devices in both our testing and development. Further information regarding our development work can be located at the following web site `http://www.pscit.monash.edu.au/~patrikm/`.

[2] Currently the KVM only runs on Palm devices