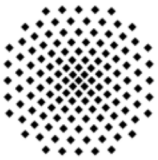

Universität Stuttgart
Fakultät Informatik

Mole - Concepts of a Mobile Agent System

Authors:
Dipl.-Inform. J. Baumann
Dipl.-Inform. F. Hohl
Prof. Dr. K. Rothermel
Dipl.-Inform. M. Straßer

Institut für Parallele und Verteilte
Höchstleistungsrechner (IPVR)
Fakultät Informatik
Universität Stuttgart
Breitwiesenstr. 20 - 22
D-70565 Stuttgart

Mole - Concepts of a Mobile Agent System

J. Baumann, F. Hohl, K. Rothermel, M. Straßer

Bericht 1997/15
August 1997

Mole - Concepts of a Mobile Agent System

J. Baumann, F. Hohl, M. Straßer, K. Rothermel

IPVR (Institute for Parallel and Distributed High-Performance Computers)
Breitwiesenstraße 20-22
70565 Stuttgart
EMail: Joachim.Baumann@informatik.uni-stuttgart.de

Abstract

Due to its salient properties, mobile agent technology has received a rapidly growing attention over the last few years. Many developments of mobile agent systems are under way in both academic and industrial environments. In addition, there are already various efforts to standardize mobile agent facilities and architectures.

Mole is the first Mobile Agent System that has been developed in the Java language. The first version has been finished in 1995, and since then Mole has been constantly improved. Mole provides a stable environment for the development and usage of mobile agents in the area of distributed applications.

In this paper we describe implementation techniques for mobility, present communication concepts we implemented in Mole, discuss security concerning Mobile Agent Systems, and present system services provided by Mole.

1 Introduction

Throughout the past years the concept of software agents has received a great deal of attention. Depending on the particular point of view the term ‘agent’ is associated with different properties and functionalities, ranging from adaptive user interfaces, cooperating intelligent processes to mobile objects. Our particular interest lies in the exploration of mobile agents in the Internet and the key benefits provided by the application of this new technology (e.g. in the area of the WWW).

Mobile agents are defined as active objects (or clusters of objects) that have behaviour, state and location. Mobile agents are *autonomous* because once they are invoked they will autonomously decide which locations they will visit and what instructions they will perform. This behaviour is either defined implicitly through the agent code (see e.g. [Gray95]) or alternatively specified by an - at runtime modifiable - itinary (see e.g. [WongEA97]). Mobile agents are mobile since they are able to migrate between locations that basically provide the environment for the agents’ execution and represent an abstraction from the underlying network and operating system.

With the properties printed out above it has been often argued that mobile agents provide certain advantages compared to traditional approaches as the reduction of communication costs, better support of asynchronous interactions, or enhanced flexibility in the process of software distribution. The employment of mobile agents has been particularly promising in application domains like information retrieval in widely distributed heterogeneous open environments (e.g. the WWW), network management, electronic commerce, or mobile computing. The question what real advantages mobile agents offer has been subject of various papers (e.g. [HaChKe95],

[BaTsVi96], [RoHoRa97]) and ongoing discussions in mobile agent mailing lists. The results of these investigations and discussions show that we are far from a common understanding concerning the pros and cons of mobile agent technology. But all agree on the following:

To support the paradigm of mobile agents, a system infrastructure is needed, that provides the functionality for the agents to move, to communicate with each other and to interact with the underlying computer system. Furthermore this infrastructure has to guarantee privacy and integrity of agents and underlying system to prevent malicious agents attacking other agents or the computer system. At the same time the agents have to be protected against a malicious system to avoid manipulations of the agents while they visit this system.

In this paper we describe the current state of our agent system infrastructure, the Mobile Agent System Mole V2. Mole builds on Java [Sun97] as the environment for the agent system and as the language for the implementation of the agents.

This paper is organized as follows: after a short introduction into our agent model in section 2 we discuss possible mobility concept for mobile agent in section 3 and present the choice we made for Mole. In section 4 we describe the communication concepts of Mole V2. In section 5 security concerning Mobile Agent Systems is examined. After presenting our notion of agent ids in section 6 we discuss the Mole system services in section 7. In section 8 we give an overview over related work. In section 9 we summarize the paper and present our planned future work.

2 Our Agent Model

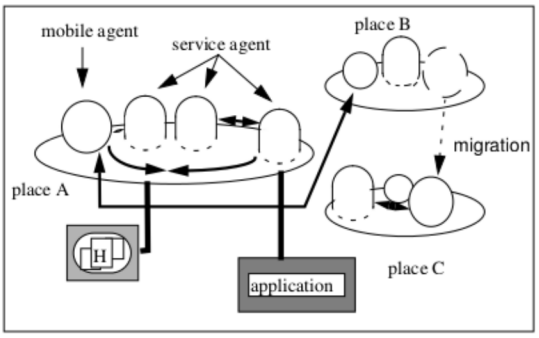

Figure 1: The Agent Model

In this section we will give only a short overview of our agent model, that has been described in much more detail in [StBaHo96] and [BaumEA97a]. Our model of an agent-based system - as various other models - is mainly based on the concepts of agents and places. An agent system consists of a number of (abstract) places, being the home of various services. Agents are active entities, which may move from place to place to meet other agents and access the places' services. In our model, agents may be multi-threaded entities, whose state and code is transferred to the new place when agent migration takes place. Places provide the environment for safely executing local as well as visiting agents.

Each agent is identified by a globally unique agent identifier. An agent's identifier is generated by the system at agent creation time. The creating place can be derived from this name. It is independent of the agent's current place, i.e. it does not change when the agent moves to a new place. In other words, the applied identifier scheme provides location transparency.

A place is entirely located at a single node of the underlying network, but multiple places may be implemented on a given node. For example, a node may provide a number of places, each one assigned to a certain agent community, allowing access to a certain set of services or implementing a certain prizing policy. Locations are divided into two types, depending on the connectivity of the underlying system. If a system is connected to the network all the time (barring network failures and system crashes), a location on this systems is called *connected*. If a system is only part-time connected to the network, e.g. a user's PDA (Personal Digital Assistant), the location is called *associated*.