AgentScape: Middleware, Resource Management, and Services

F.M.T. Brazier, D.G.A. Mobach, B.J. Overeinder
S. van Splunter, M. van Steen and N.J.E. Wijngaards

Intelligent Interactive Distributed Systems Group and Computer Systems Group
Department of Computer Science, Vrije Universiteit Amsterdam
http://www.iids.org/

Abstract

The AgentScape project is geared to support large-scale distributed systems at three levels: middleware, services, and applications. This extended abstract presents the basic AgentScape architecture, resource management, and one specific service for code mobility in more detail.

1 Introduction

The AgentScape project aims to support large-scale distributed systems. The project encompasses three well-defined research areas: (i) the AgentScape middleware, (ii) services in AgentScape, and (iii) applications designed and implemented in the AgentScape environment.

Large-scale distributed systems are often heterogeneous systems: heterogeneous with respect to the host architecture (Sun SPARC, Intel x86/i64), the supported operating system (Solaris, Linux, Windows NT), and the communication infrastructure (bandwidth and latency). The major challenge in the AgentScape project is to realize a scalable, secure, and fault tolerant system, that supports multiple distributed applications, heterogeneity, and multiple qualities of services.

AgentScape specifically deals with large-scale distributed systems on which, mobile, *autonomous* processes run. The mobile, autonomous processes are called agents, the passive, possibly distributed, entities are called objects. An important characteristic of agents, which distinguishes them from traditional processes, is autonomy: agents are in control of their own behaviour. From a management point-of-view, this has implications for the management of these mobile autonomous processes.

This extended abstract presents the middleware, resource management, and one specific service for code mobility in more detail.

2 AgentScape Operating System

The AgentScape operating system (AOS) provides a platform with which mobile, autonomous processes (agents) can be managed. It is, in fact, a virtual machine distributed over a wide-area network consisting of heterogeneous hosts. AOS kernels host agents, objects, and provides service access (see Fig. 1). All calls are filtered by the middleware and appropriate calls are dispatched to the underlying operating system, services, etc.

A location in the distributed system is a set of hosts run by a single administrative entity. Each host runs a *minimal* AOS kernel, and zero or more agent servers, objects servers, and service access providers. An agent server hosts agents, an object server hosts objects, and a service access provider makes external services accessible within AgentScape. A location is implemented by the distributed AOS kernels, the agent servers, the object servers, and service access providers.

The current architecture for this system is a middleware layer, on top of which agent applications and agent platforms can be developed (see Fig. 1). The current prototype implements the basic functionality required.

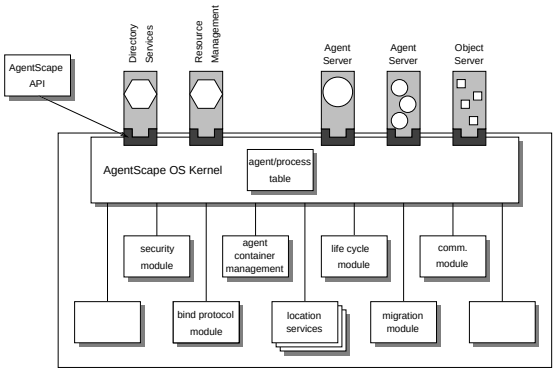

Figure 1: AgentScape middleware architecture.

3 Resource Management

AgentScape's resource management system, based on the OSI management model, needs to be able to to regulate agents without interfering with their autonomy.

The OSI management model defines five functional management areas: performance, security, fault-tolerance, accounting, and configuration.

These areas can also be applied to the management of multi-agent systems such as AgentScape:

- Performance management: managing the resource usage of agents on hosts within a multi-agent system.
- Security management: ensuring security of both agents and hosts in a multi-agent system.
- Fault-tolerance management: ensuring the availability of agents and objects within a multi-agent system, as well as the multi-agent system as a whole.
- Account management: maintaining administrative information about the agents within a system.
- Configuration management: managing the configuration of agents and objects that are under management control.

The management system in the AOS needs to create, delete, migrate, etc., agents. An agent life-cycle model forms the basis for the definition of these operations on agents by defining states of the agent and transitions between these states. The central state in the life-cycle model is the suspended state, in which the agent is inactive and can be manipulated. This is in contrast to other life cycle models for agents, in which the active state of an agent is the central state.

4 Services

One specific AgentScape service extends the basic mobility of agents to true heterogeneous environments, i.e., different operating systems, different agent platforms, and different programming languages. The service that makes this possible is called the Agent Factory of which a number of prototypes have been implemented.

Mobile code is often not optimized for a wide variety of heterogeneous hosts (including differences in operating systems, hardware configuration, and available resources). Rewriting mobile code for each host encountered in the network, is not a feasible solution, requiring too much effort from system administrators and developers.

The Agent Factory service automatically adapts mobile code to a specific host: a form of generative mobility. In this approach, the mobile code need not be sent to another host, but a blueprint of the mobile code's functionality is sent, together with information needed to resume work. At each host, a service is available which inspects a blueprint and generates the corresponding program code. The program code generation process uses libraries of "building blocks" to reconfigure agents.

An additional advantage of regenerating mobile code, is that the risk of malicious behaviour (or acquiring viruses en route) is reduced by regenerating the mobile code, e.g., using trusted libraries of building blocks. The data (state) of the mobile code may also be inspected, if required.

About the Authors

Frances Brazier is a full professor in the Intelligent Interactive Distributed Systems group at the Vrije Universiteit Amsterdam. Benno Overeinder and Niek Wijngaards are assistant professors in this group which focuses on the interdisciplinary area between Computer Systems and Artificial Intelligence. Sander van Splunter and David Mobach are PhD students in the same group. Maarten van Steen is a full professor in the Computer Systems group at the same university. The authors are grateful to Andy Tanenbaum and Etienne Posthumus for their significant contributions to this research and Stichting NLnet for their financial support.

More Information

[1] F.M.T. Brazier, B.J. Overeinder, M. van Steen, and N.J.E. Wijngaards. Agent factory: Generative migration of mobile agents in heterogeneous environments. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002)*, pages 101–106, Madrid, Spain, March 2002.

[2] F.M.T. Brazier, M. van Steen, and N.J.E. Wijngaards. On MAS scalability. In *Proceedings of the Second International Workshop on Infrastructures for Agents, MAS, and Scalable MAS*, pages 121–126, Montreal, Canada, May 2001.

[3] M. van Steen, P. Homburg, and A. S. Tanenbaum. Globe: A wide-area distributed system. *IEEE Concurrency*, 7(1):70–78, January–March 1999.

[4] N.J.E. Wijngaards, B.J. Overeinder, M. van Steen, and F.M.T. Brazier. Supporting Internet-scale multi-agent systems. *Data and Knowledge Engineering*, 2002. in press.