Mansion: A Structured Middleware Environment for Agents

Guido van 't Noordende, Frances M.T. Brazier, Andrew S. Tanenbaum
Division of Mathematics and Computer Science, Vrije Universiteit Amsterdam.*

Abstract

Developing processes intended to roam in large-scale heterogeneous distributed systems is difficult: their environment is unstructured, and interoperability issues often emerge. Mansion is a new paradigm to provide a large-scale structured environment for mobile agents (autonomous migrating processes). Security, transparency, and interoperability are important design guidelines for the development of Mansion.

1 Introduction

Mansion is a system aimed to support heterogenous, large-scale distributed mobile agent applications. Mobile agents have a number of well-described advantages over traditional distributed systems [1]. The most significant of these is that an agent can move its computation to the resource or data which it needs, which alleviates problems due to latency or bandwidth limitations. Further advantages are related to security, reliability and fault-tolerance, among others.

There are a number of solutions for Multi-Agent Systems (MASes), which focus on the above-mentioned properties of mobile agents [2, 3]. However, none of the existing solutions offer a well-defined paradigm or framework to help application designers produce coherent, well-structured environments. Mansion is a MAS which provides a clear paradigm for designing multi-agent applications. In addition, Mansion provides a middleware system which is designed to be scalable, provide location and distribution transparency, and offer security to both agents and hosts in the system. Mansion supports heterogenous agents and is designed to work on different platforms in a worldwide distributed system.

2 The Mansion Paradigm

An application in Mansion is modeled as a world consisting of hyperlinked rooms. Each room contains agents, objects, and hyperlinks to other rooms. Agents are active entities and objects are strictly passive entities. Agents can communicate with other agents (using point-to-point communication) in the world, and can interact with objects in their current room.

Each world has one or more entry rooms, through which agents can enter the world. At any instant, an agent is always in one room, but can later traverse a hyperlink to another room. Each world may also have an attic, which is a room which contains global services that can be used by any agent in the world. An agent cannot move to the attic.

Examples of typical applications include:
- a library world: the rooms contain books on specific subjects, and are completely interconnected;
- a game-world: the rooms form a maze through which agents roam to achieve some goal;
- a shopping world.

For example, a shopping world may consist of many separate stores, each occupying a separate room, accessible from one or more entrances. Objects may represent items (for example clipart or music) for sale, and agents roam through this world to find and possibly buy items to their liking. Agents can communicate with each other to speed up their search or notify each other of interesting bargains.

3 The Mansion Middleware

Each world has its own Mansion middleware, which can be adapted to the specific needs of the application. The middleware is distributed over the hosts on which rooms are available.

Mansion is designed to be scalable, with respect to the number of agents, rooms, hyperlinks, and objects in a world. The Mansion middleware provides location and distribution transparency for logical entities in the world.

To represent rooms and objects in a world, we use the The Globe object architecture [4], which makes it possible to distribute rooms and objects over multiple hosts. This can provide reliability (availability) and locality of data in the system.

The Mansion middleware extends the functionality provided by the AgentScape OS (also described in these proceedings). A world description language specifies basic rules, including constraints about hyperlink layout and reliability of communication. On the basis of a world specification, Mansion middleware is configured, after which the world can be deployed.

* {guido,frances,ast}@cs.vu.nl

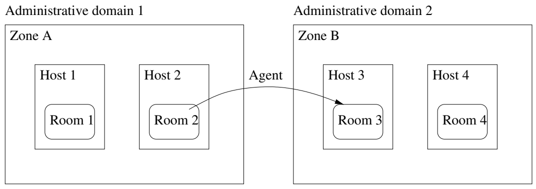

Figure 1: Example of agent migration

4 Constraints in Mansion

Mansion provides a closed world in which constraints can be enforced, e.g. constraints on agents entering a world, agents traversing hyperlinks between rooms, agent communication, and agents interacting with objects. An important constraint is that an agent may only access objects in its room. In addition, information carried by agents may be subject to (security) constraints. The Mansion middleware is aware of the logical and security constraints of a world and enforces these.

An administrative domain is represented by a group of hosts, called a *zone*. Rooms are deployed in a zone. A zone may contain multiple rooms, but each room is only accessible from within one (its own) zone. Before an agent can enter a room, it has to move to one of the hosts in that room's zone. Figure 1 shows an example. An agent wants to follow a hyperlink to room 3. As a result of following the hyperlink, the agent is moved (physically) to host 3, from which room 3 is accessible.

To protect hosts in the system, an administrator can place a firewall around its zone. Each zone then has an entrance policy, which defines what agents may enter the zone and the rooms in that zone. A decision on accepting an agent can be made based on knowledge about the agent's owner or creator, the language in which the agent was written, or the possession of certain certificates or credentials. To protect the host agains potentially malicious agents, the middleware needs to start the agent up in a secure execution environment. An example of such an environment is a Java JVM, which can be used to sandbox a Java agent.

An agent's physical representation (code) and data is stored into an Agent Container (AC), which is the unit in which the agent is physically moved from host to host (as it moves from room to room). The agent's code may be true code, for example a binary, but it may also be the name of a class object. It is up to a receiving host to determine whether and in what form it wants to accept and execute an agent.

As an example of a mechanism which helps to protect an agent against attacks (e.g., by the host on which it runs), the middleware contains a mechanism which makes it possible to trace (malicious) alterations to an AC [5].

5 Discussion

In this paper we gave an overview of basic functionality offered by Mansion. Mansion can be used to design distributed applications in an intuitive natural way, based on its paradigm of hyperlinked rooms.

In Mansion, numerous aspects related to security, scalability and deployment can be controlled. Security policies can be defined to protect agents, administrative domains (hosts), rooms and objects in the system. Constraints on hyperlinks make it possible to control the logical topology of a world. The configurability of the Mansion middleware using the world design language makes it possible to adapt the middleware to the specific needs of an application.

6 About the Authors

Guido van't Noordende is a Ph.D. student in the Computer Systems group of the Department of Sciences at the Vrije Universiteit Amsterdam; Frances Brazier is a full professor in the Intelligent Interactive Distributed Systems (IIDS) group; Andrew Tanenbaum is a full professor in the Computer Systems group. IIDS is (financially) supported by Stichting NLnet.

We wish to acknowledge Benno Overeinder and Niek Wijngaards (both IIDS) for their useful contributions to the model and this paper.

References

[1] D. Chess B. Grosof C. Harrison D. Levine C. Parris G. Tsudik. Itinerant agents for mobile compputing. *IEEE Personal Communications*, pages 2(5):34–49, October 1995.

[2] R.S. Gray, D. Kotz, G. Cybenko, D. Rus. D'Agents: Security in a Multiple-language, Mobile-agent System. *Mobile Agents and Security*, pages 154–187, 1998. LNCS 1419, Springer-Verlag.

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

[4] A.S. Tanenbaum M. van Steen, P. Homburg. Globe: A wide-area distributed system. *IEEE Concurrency*, January-March 1999.

[5] N. Karnik and A. Tripathi. Security in the ajanta mobile agent system. *Software - Practice and Experience*, 2001.