# Service Location for Mobile Agent Systems[1]

Marcelo Gonçalves Rubinstein and Otto Carlos Muniz Bandeira Duarte

Grupo de Teleinformática e Automação - GTA
Universidade Federal do Rio de Janeiro
COPPE/EE - Programa de Engenharia Elétrica/Departamento de Eletrônica
P. O. Box 68504 - 21945-970 - Rio de Janeiro - RJ - Brazil
phone: + 55 21 2605010 - fax: + 55 21 2906626
e-mail: {rubi, otto}@gta.ufrj.br

*Abstract - One of the main challenges of the mobile agent technology is how to obtain a host that provides the service specified by the agent. This work proposes a service location module for a development environment for agent-based systems named GTA/AG Infrastructure (GTA/AGent). This module makes easier the agent programming since the service location module will provide an agent's destination.*

## 1. Introduction

With the extraordinary development of the computer global network, the Internet, three main facts have been reason of preoccupation by the network community and the application developers: the quantity of information available in the world, the data moving cost, and the little experience on computing of lots of users. The quantity of information available in different sites of the world has not been followed by the user's capacity of searching, selecting, and retrieving this information. Moreover, the movement of great quantities of data on the network has a significant cost. At last, the number of users with little knowledge on computing has grown and will continue to grow very fast. One possible solution to these problems consists in agent utilization.

*Agents* are programs that help users to perform tasks, acting on behalf of the users. If the agents can execute their tasks on the network, they are called *mobile agents*. Mobile agents can move to the place where data is really stored and select the information the user wants by using intelligence. Hereby, the total quantity of data from the various sources is not transferred, saving bandwidth, time, and money. Agents also make easier the life of the users with little knowledge on computing, working as smart user interfaces that can detect and solve problems, learn user's preferences and habits, and automate everyday actions.

An important issue is how a mobile agent chooses a host to move to. In the simplest case, a mobile agent can carry with it a place itinerary. The problem of creating this itinerary is addressed in this paper.

Most agent-based systems do not address the itinerary problem. TACOMA (Tromsϕ and COrnell Moving Agents) [1] introduces the *meet* abstraction to support agent moving. The host is specified by the agent programmer in the *host* folder. Agent Tcl [2] uses the *agent_jump* command to jump from a host to another. The programmer has to specify the agent's destination. Odyssey [3] uses the *go* instruction to make an agent travel from one place to another. This instruction requires a *ticket*, data that specify the agent's destination, but the programmer is responsible for creating a ticket. The IETF (Internet Engineering Task Force) has addressed the itinerary problem by creating a protocol to find service hosts. The Service Location Protocol [4] is a simple, lightweight protocol for automatic maintenance and advertisement of intranet services.

This work proposes a service location module for obtaining hosts that provide the service specified by the mobile agent. This module is an extension of the IETF Service Location Protocol, since it has a global scope and it was adapted to deal with the mobile agent technology.

[1] This work was suported by UFRJ, FUJB, PROTEM-CC, CNPq, CAPES, and COFECUB.

This paper is organized as follows. In Section 2, the structure of a mobile agent is presented. In Section 3, the architecture model of an agent infrastructure is described. In Section 4, the service location module is proposed. Finally, in Section 5, the concluding remarks are presented.

## 2. Structure of Mobile Agents

Mobile agents consist of code, state, and attributes. *Mobile agent's code* is a program that defines agent's behaviour. An agent's code can be written in any language, but an agent should be executed in the same way on any host it can move to [5]. So, directly interpreted languages or languages compiled to an intermediate-based language which can be easily transported and executed without recompilation are the most appropriate for agent implementation. Tcl, Perl, and Java are examples of such agent programming languages.

Besides code, an agent carries a *state* that allows subsequent actions to be based on past actions. Then, an agent can resume its work at the suspended point after moving to another host. Agent's state contains the execution point on the new host and the necessary variables to its execution.

The third part of a mobile agent consists of *attributes* that describe the agent, its requirements, and its history for the infrastructure. They include data such as a unique agent identifier, the agent's owner, error messages, and movement history.

## 3. Architecture Model

Agents must interact with their hosts in order to use their services and to negotiate services with other agents. Mobile agents must also be able to move on a network environment. An environment called agent infrastructure allows these actions and an entity named agent server provides support for the agents on a particular host. Besides these basic services, an agent infrastructure must also provide security in order to protect the host and agents from non-authorized access and eavesdropping.

This infrastructure named GTA/AG Infrastructure (GTA/AGent) aims to provide generic services such as migration, communication, execution, security, fault tolerance, and service location, allowing an agent-based application developer to focus on application details, without worrying about the infrastructure. Since it is a modular and flexible structure, it is possible to choose and configure modules by system characteristics, and create new modules. According to Lingnau *et al.* [5], a general infrastructure should provide mechanisms, not policing to be accepted in a global way. Most infrastructures, while differing considerably in realization, use the same basic solution for portability and security: they do not run agents on the real machine of processor, memory, and operation system, but on a virtual machine [6].

An *agent server* is the basis of the architecture model used by the agent infrastructure (Fig. 1). This server is in charge of providing various functions like agent transport, security, communications of agents with the host, other agents, and their owners, and fault tolerance. An environment suitable to agent execution named *runtime environment* is the interface between an agent and its host. Their main functions are to protect the host, to capture agent's state, and to interact with the server. For each agent being executed in a server, a runtime environment is created by the server. An agent does not have direct access to other agents being executed in the same host. The environment provides resources to the agent in a controlled way. The agent's owner interacts with the infrastructure by an entity called *client*.

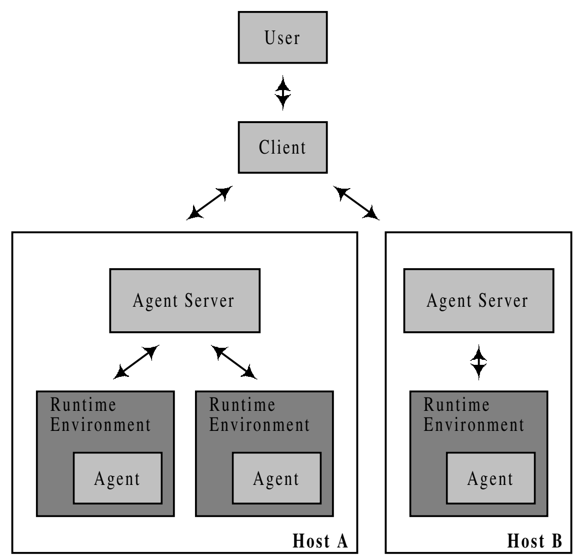


Fig. 1. Architecture Model.

The agent server is a program run on each computer accessible to the agents and it is in charge of the agents being executed on that computer. Its tasks include accepting agents, creating the appropriate runtime environment, supervising agent executions, and finishing them when necessary. The agent server must also organize agent transfer to other hosts, manage communications among agents and between agents and their owners, authenticate and control access for agent operations, recover agents and the information carried by them in case of network and computer failures, and provide a list of service location hosts.

Agent server functions are separated in modules: Security Module, Fault Tolerance Module, Communication Module, Transfer Module, and Service Location Module.

Perhaps the most important issue for the total acceptation of the agent-based systems is *security*. In a network environment, neither the agent nor the hosts are trustworthy. A lot of people associate mobile agents to the threats imposed by malicious programs like viruses and Trojan Horses. An agent can attack a machine and obtain access to local resources. A server must not corrupt agents in its charge or even look the code implementing a competitor negotiation strategy.

Recovering mechanisms for the agents and the information carried by them are used in case of network and computer failures, in order to resume execution at the suspended point. *Persistence* is used by storing from time to time agents, its data, and its states in non-volatile memories. This storing is based on checking points, which normally are critical points in an agent execution. When one of these points is reached, the previously described information is stored in the non-volatile memory.

In order to perform useful work, agents must be able to *communicate* with the host, other agents, and their owners. An agent can communicate with another agent in the same host.

One of the main characteristics of the mobile agents is their *movement* from one host to another. This movement happens in accordance to an agent explicit requirement. The agent implementation language includes a sentence or a library call in the form of *move-to <host>*. When this call is executed, the agent is suspended, encoded for transmission, transmitted to the new host, decoded, and installed to resume execution in this new host, preserving the actual state and the agent integrity.

Another important issue is how an agent chooses a host to move to. In the simplest case, a mobile agent can carry with it a place *itinerary*. The problem of creating this itinerary is addressed in the next section.

## 4. Service Location Module

The service location module helps an agent to perform its task, by pointing it to some host that provides the specified service. There is no need for the user to know the name of that host.

Services are described by configuring values for the attributes which are possible for those services. For instance, a store can be described as a bookstore.

### 4.1 Service Discovery

In every host, there is a list of the service location hosts. A mobile agent migrates to one of the specific hosts and starts communicating with a stationary agent, the *discovery agent*, that provides the service location service. After obtaining the itinerary, the mobile agent migrates to the first location pointed by the itinerary (Fig. 2). In case of not finding some information about the required service, the mobile agent migrates to another service location host, an alternate or replicated host pointed by the previous one, in order to try to discover a host that provides the specified service.

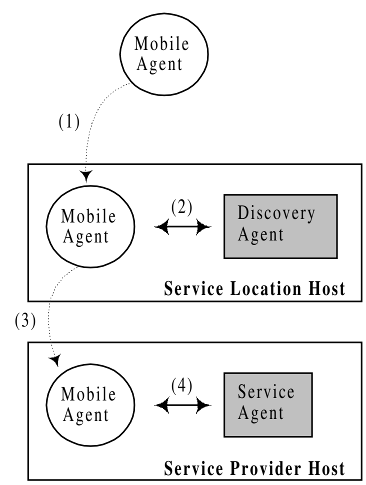


Fig. 2. Service Discovery.

The agent uses keywords and specific parameters to describe the specified service. These keywords and attributes can be combined into boolean expressions with: AND, OR, and NOT operators, common comparators (=, >, <, >=, <=), and substring matching. The URL (Uniform Resource Locator) of the server responsible by the service is returned to the mobile agent. The main reasons to choose a URL as an answer to the agent's question is compatibility with web browsers and independence of address families (like IP, IPv6, or IPX). One simple example of the use of the service location service follows: if a user needs to know a book's price, it sends a mobile agent with the keyword *book* to a service location host. This host will provide to the mobile agent an URL of a bookstore. Then, the user will send another mobile agent to the bookstore host in order to find the book's price.

### 4.2 Service Registration

Services are advertised by registering with an agent server and including in the registration a list of all the keyword and specific parameters that describe the services. Each register has a life-cycle in order to avoid cases where service hardware breaks but the service continues to be advertised and explicit deregistration can also be performed.

Each service provider sends a mobile agent in order to register its service (Fig. 3). The mobile agent includes in its parameters all the keywords and specific parameters that describe the service to be provided. The mobile agent communicates with a stationary agent, the *registration agent*, which is responsible for updating the list of service location hosts.

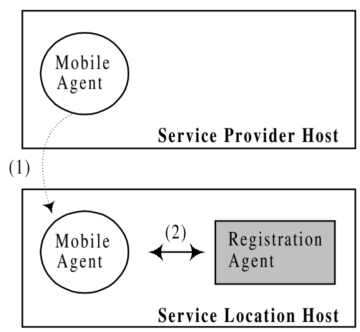


Fig. 3. Service Registration.

Explicit deregistration is performed by sending a mobile agent to a service location host with all the keywords and specific parameters that describe the service to be deregistered. The mobile agent communicates with a stationary agent, the *deregistration agent*, which is responsible for updating the list of service location hosts.

## 5. Concluding Remarks

The main mobile agent applications are information retrieving, network management, electronic commerce, and mobile computing. The development of an infrastructure that can be used for communication between agents and hosts, and agent mobility turns easier a great number of services and network applications.

The GTA/AG Infrastructure provides generic services such as migration, communication, execution, security, fault tolerance, and service location, allowing an agent-based application developer to focus on applications details, without worrying about the infrastructure. Since it is a modular and flexible structure, it is possible to choose and configure modules by system characteristics, and create new modules.

This work proposed a service location module for the GTA/AG Infrastructure in order to obtain hosts that provide the service specified by the mobile agent. This module makes easier the agent programming since the service location module will provide an agent's destination.

## References

[1] D. Johansen, R. van Renesse, and F. B. Schneider, "An Introduction to the TACOMA Distributed System", technical report, University of Tromsϕ/Cornell University, Tromsϕ/Ithaca, Norway/USA, June 1995.

[2] R. S. Gray, "Agent Tcl: a Transportable Agent System", *in CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management*, Baltimore/Maryland, USA, 1995.

[3] General Magic, "Odyssey", available from http://www.genmagic.com/agents, 1997.

[4] SVRLOC Working Group of the Internet Engineering Task Force (IETF), "SLP White Paper Topic", technical report, available from http://playground.sun.com/srvloc/slp-white-paper.html.

[5] A. Lingnau, O. Drobnik, and P. Dömel, "An HTTP-based Infrastructure for Mobile Agents", *in Proceedings of the 4th International WWW Conference*, Boston, USA.

[6] H. Peine and T. Stolpmann, "The Architecture of the Ara Platform for Mobile Agents", *in 1st International Workshop on Mobile Agents,* Berlin, Germany, pp. 50-61, April 1997.