International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Study of Execution Mechanism of Intelligent Mobile Agents

Sanjay Yede[1], Dr. V. N. Chavan[2]

[1]P.G. Department of Computer Science and Technology, Degree College of Physical Education, HVPM, Amravati-444605, India

[2]Head, Department of Comp. Sci, Seth Kesarimal Porwal College, Kamtee, Nagpur, India

Abstract: ***Mobile agents are programs that are moved from one source computer and roam among a set of networked servers until they fulfill their task. They have the ability to acquire knowledge by transporting themselves from one place to another. While moving from host to host they execute in agent execution environment called as 'Agent Meeting Point (AMP)' at each host. Upon arrival at an AMP, service requests are resolved to determine if the desired services are available at that AMP. If sufficient resources are available and permitted the constituent parts of the agent are passed to the services and agent starts execution. The central objective of this paper is to study architecture of Mobile Agents and functioning of Agent Meeting Point (AMP) an abstraction that supports interaction of agents with each other and server based resources.***

Keywords: Mobile-program, Agent-Meeting-Point-(AMP), KQML, APL, Execution-Environment

1. Introduction

Mobile agent is a program that is dispatched from one computer and travel in a network until it accomplishes its goal. This is an extension to the client/server model in which the client sends requests to the server for execution, which executes it and responds with results if it can satisfy the request or an error otherwise. The Mobile Agents do migrate among different hosts in a network looking for a host that can perform task at hand or for collecting required information [2,5].

The Internet contains an enormous number of computers, which are capable of providing specific services, and are comprised of a wide variety of processors, operating systems, databases, frameworks, and applications. The mobile agent framework makes these heterogeneous servers to offer many advantages like host-independent execution environment, standard agent communication languages, authenticated access to server resources, and secured auditing and error recovery mechanisms [3]. The central objective of this paper is to discuss architecture and working of '**Agent Meeting Point**' an abstraction that supports the interaction of mobile agents with each other and other resources.

2. Motivations Behind Developing Mobile Agents

There are many different motivations behind development, distribution and use of such Agents. These motivations are divided into two categories:

1. The support for mobile and lightweight devices, such as laptops, palmtops, PDA's etc., and,
2. The emerging need for an asynchronous method of searching for information or transaction services in corporate networks

For example,

1. The ability of agents to make lightweight mobile computers to interact with heavyweight applications without prior, detailed knowledge of the remote server's capabilities.
2. The ability of the agents to integrate knowledge from the client and server and perform inference based on it.
3. Facilitate creation of personalized services by customizing agents which can then reside at server.
4. The agents facilitate high-bandwidth communication to search through large free text databases.
5. Their ability to reduction of overall communication traffic over the low bandwidth networks employed by mobile computers.

3. Working of Mobile Agents

The Mobile agents have the ability to move from place to place and acquire knowledge about the system and available resources. Generally, it is initialized with the user's task and launched from a client application. It is then transmitted through a channel over a network to accomplish the task assigned to it. The sending client may specify a destination service directly, but desirably the client more likely send the agent initially to a yellow page server, which in turn proposes servers to be visited to fulfill the user's task. When the agent reaches a server, it is delivered to an agent execution environment, called as Agent Meeting Point (AMP) [6]. Upon arrival at an AMP, the agent's external wrapper is inspected for authentication. After validation, the AMP examines the agent's description of it self. Ontologically named service requests are resolved to determine if the desired services are available at that AMP. If sufficient resources are available and permitted the constituent parts of the agent are passed to the services. The executable portions of the agent are then started. In some cases the mobile agent interact directly with server resources via proxy objects, which enable access control to be enforced. In other cases, it interact with a *static agent* resident at the AMP.

Static agents enable the server's function to be personalized by the server's owner or by users. When the agent successfully completes its task at this server, it collects its

state, including information acquired at the server, and request to be transported to a new host. Or, it may launch a smaller agent to deliver the acquired information to the sending client or to another server while it terminates. This ability means that the agent is not merely a program executed at a remote host and then returned to its origin, but a moving process that progressively accomplishes a task by moving from place to place.

If the agent proves to be unauthorized, or if the meeting point is unable to provide the agent with the resource it has requested, the AMP takes action based on the agent's header. It may discard the agent, it may send the indicated party a description of the failure, or if it is capable it may propose one or more AMP's to satisfy the request.

4. Structure of Mobile Agent

The basic structure of a mobile agent is divided into three distinct portions: the *Agent Passport*, the *Table Of Contents (TOC)*, and the *Components*.

a) **Agent Passport**: Agent passport consists of the basic information required to permit the agent to move from one AMP to another [6]. It includes,
- Authentication of Originator**:** This certificate includes the name or the authority of the request's owner and the name or names of other authority sanctioning entities. The certificate carries identity of the originator of the agent and the "network name" of the business server.
- Error Actions and Addresses**:** This is the action the AMP should take when an error occurs while processing the agent. Some possible actions include: discarding the agent without comment; delivering an error notification to a specified address; routing the agent to another AMP.
- Goals and Status Information**:** This information includes a representation of the agent's goal, and can include its relationship to other agents and their goals. Note that this agent may be the child of an agent present at the AMP, returning after an assignment from that parent. In some cases, the agent may require the AMP to notify an external entity (another agent or a client) of its status or progress based on some specified condition. The address of the external entity and the conditions are provided in this information.

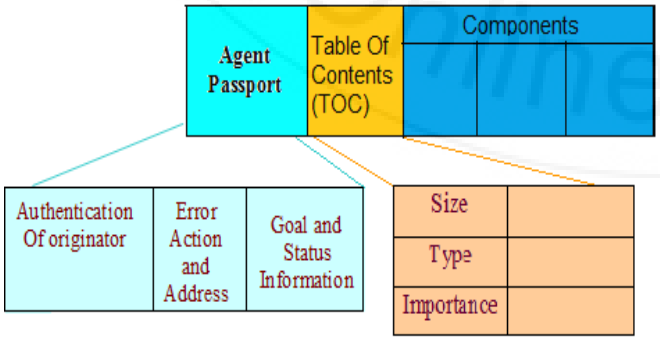

Figure: Agent Format

b) Table Of Contents

TOC for the body of an agent provides a map of its structure. Each component has a size, type and importance. The size, as expected, is the size of the component. The type field contains a simple representation of what is required to process the component. The importance field describes whether the component is necessary for the agent to be instantiated at the AMP. This permits agents to carry obscure components through AMP's that do not support these components, and to avoid unpacking components that will not be used at any AMP.

c) Agent Languages

Agent Programming Languages

One of the basic goals of agent framework is that it should support a broad range of languages. In fact, any executable environment that supports the architecture of the AMP can be chosen. However, some languages are more suitable for writing mobile agents than others [6]. High level and Object Oriented Programming provide several advantages for building agent frameworks. The object abstraction provides a good leverage point for access control and data mobility. Other desirable attributes for an agent programming language are discussed below.

- **Mobility Hooks :** Mobility is a key feature of mobile agents, so it is desirable that an agent language easily support moving the agent [5, 6]. However, agent language support is not enough and some standardized primitives will be needed to invoke the collection of an agent's parts and the movement of the agent from one AMP to another.
- **Fork/Spawn construct :** As with mobility, mobile agents frequently desire to split up their processing among multiple copies of themselves at multiple AMPs or spawn agents to traverse the network. Again it is not necessary that the language expressly include support for this as primitive, as some standardized primitives will have to be provided by the environment.
- **Distributed Computing :** Being mobile, agents have to rely on networking protocols and deal directly with a Distributed Computing Environment. It is desirable for the agent language to provide constructs that help programmers manage issues related [4, 9] to DC. The ability to create proxy objects to represent remote objects, advertising Remote objects to the environment, broking the requests, expressing synchronization points between multiple threads of execution, constructs to manage objects shared between execution environments are the examples of this sort of support.

d) Knowledge Representation Languages

Languages for knowledge representation (KR) provide the means to express goals, tasks, preferences, beliefs, and vocabularies appropriate to various domains. Such goals and beliefs may take the form of facts, rules, other logical formulae, defaults, probabilities and utilities, fuzzy sets and logic, neural networks, and plan and action operators, to name a few. A Language is often associated with vocabularies used to describe predicates and functions out of which belief and goal expressions are syntactically built.

5. Moving Programs and Accumulated State

5.1 Moving Programs

Concerning the movement of mobile agents, two possibilities exist: a simple case where the agent is a program that runs to completion at the agent's destination, and a more complex case in which an agent program begins executing at one AMP and then decides to move, complete with its current state, to another AMP [9]. In the simple case, the program can be encoded, loaded, and run until it signals completion. This can be done with only a few changes to an existing execution engine.

In the more complex case in which a program wishes to move from one server to another in mid execution, an important question arises how to extract the program's variables and state from the execution environment and insert them into another execution engine.

Briefly, there are several approaches to the problem of moving executing programs. One of the easiest is to allocate all information associated with the program's execution on the program stack and transport the stack, along with the engine, from one execution environment to another [5]. Another approach involves keeping a registry of variables as they are created and moving the registry and the variables. In these two cases and other similar schemes, more extensive changes are required to the execution environment.

5.2 Accumulated State

An agent traversing several servers within a network of servers may need to accumulate information derived at each of the servers it visits. Agents accumulate this information through two primary mechanisms: by adding new objects containing the new information and state to its existing collection of objects, or capturing state through changes within its existing collection of objects.

Consider an agent that is searching for information matching some specific request. As the agent traverses the network of servers, it adds these matching documents to its collection of items. Note that many of these items could be marked as cargo in the TOC and not be instantiated as the agent visits various AMPs.

State can also be captured within an agent's existing collection of objects. The degenerate case of this is an agent that consists of a single object, that is, an executable program and its execution context. An agent searching for the lowest price offer for a specific item need only carry the current lowest bid (and the identity of the bidder) in a local variable within its execution context. As the agent moves from AMP to AMP, the program and its context are saved, moved, and restored.

5.3 Agent Meeting Point Structure

Agent Meeting Point consists of a set of classes designed to provide a very lightweight framework for building agent meeting points. The framework provides functionality for basic services like registering the services that a specific AMP will use, a Shallow Request Handler, that will permit components that snap into the framework to express what function they provide in terms of a simple knowledge representation scheme. The framework provides a sufficiently rich and robust description of the AMP while allowing for incremental development and extension of the AMP structure.

The structure supports basic agent interaction and can be easily expanded to support a broad range of service types and interaction models. The approach is to articulate an underlying set of component frameworks that focus on building a minimum set of services for registering and routing work within the AMP.

On top of the Shallow Request Handler and the base components are defined additional parts that provide specific services. These parts center on the Deep Request Handler and the Linguistic Registry, the focus for managing more complex knowledge representation and semantic issues.

5.4 Communication Services

Agents can communicate remotely with other agents or AMPs by exchanging messages. The messages directed to components and agents residing within the AMP are handled by Communication Portals which utilize the Shallow Request Handler to direct such messages into the AMP. They can also manage arrival and departure of mobile agents. The Communication Portals support protocol handlers, which manage communication based on a specific protocol, and location objects, which provide methods to send agents and messages to abstract locations.

The Communication Portals extract the arriving mobile agent and pass it to the mobile agent concierge for authentication, along with a location-object representing its location. They also mask the underlying transport service and transport-specific details such as media, header, trailers, and data representation. The actual communications may be based on session oriented protocols or message oriented protocols.

5.5 Request Handler

The Request Handler acts as the interface between agents and the components of the AMP. Sometimes the job of request handling is seen to be divided into two different levels, the one at which trivial requests are handled and the other at which the requests for detail services are handled. The request handling components at these levels are named as Shallow Request Handler and Deep Request Handler respectively. The Shallow Request Handler uses description of the components form registry to route requests from agents to the components that support them. It uses a limited vocabulary to match user requests with available services. They can also recognize requests expressed in forms of notation beyond its vocabulary. In such cases, it checks its cache for a translated version of the request. If it does not find one it sees if the Linguistic Registry has the vocabulary registered, and if so, passes the request to the Deep Request Handler for translation.

The Deep Request Handler on the other hand helps the Shallow Request Handler deal with more special or difficult requests. It maps a request by an agent into one or more service destinations, which may be another service or another agent, either in this same AMP or at a remote server. The Deep Request Handler provides, in effect, an extended directory service. It can be viewed as a kind of facilitator, and hence as itself a kind of agent. An interesting case occurs when the service destination is another agent that performs a similar task, that is, a facilitator agent. Generally, an alternative aspect of the Deep Request Handler is the provision of a "social directory" of other agents, which may include their identities, interests, languages, vocabularies, and addresses.

5.6 Resource Manager

The proper management of resources in the AMP and control of the services is governed by the Resource manager**.** It serves two primary purposes. On one hand it acts as a registry for all of the active agents within the AMP and the resources associated with these agents; on the other hand it serves as the controller for managing the use of resources within the AMP. It also keeps track of current resource allocation. The Resource Manager can also act as a firebreak against the excessive use of resources by agents. Either the total permitted resources can be limited, or the rate at which the resources are consumed can be controlled.

5.7 Agent Execution Environments

These are execution environments that have registered to the AMP, offering to interpret scripts or agents whose encoding they support. In addition to registering their named environment with the meeting point, the Agent Execution Environments must provide access to the base facilities of the agent meeting point. This includes access to all of the exported object encapsulations.

6. Conclusion

This paper presents an overview of architecture for Mobile Agents and the structure of Agent Meeting Point, a platform for communication for agents within an execution environment. A framework is discussed that supports mobile agents. This framework provides a secure facility that can support interaction between agents requiring diverse execution environments, communicating via multiple languages, and traveling via multiple transport services.

References

[1] Devadithya T, Chiu K, Huffman K and McMullen DF, "The Common Instrument *Middleware Architecture: Overview of Goals and Implementation", in Proceedings of IEEE International Conference on e-Science and Grid Computing (e-Science 2005), Melbourne, Australia, December 5-8, 2005.*

[2] *Foster* I, Czajkowski K, Ferguson D E, Frey J, Graham S, Maguire T, Snelling D and Tuecke S, *"Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF"* , in Proceedings of the IEEE 93(3):604-612, Mar 2005.

[3] Satoh I, "Building and Selecting Mobile Agents for Network Management", Journal of Network and Systems Management, vol.14, no.1, pp.147-169, Springer, 2006.

[4] Rafael P, Cristina S and Jodie F, "Cooperative Itinerant Agents (CIA): Security Scheme for Intrusion Detection Systems", In the proceedings of International conference on Internet surveillance and Protection (ICISP), 2006.

[5] Papastavrou S, Pitoura E and Samaras G, "Mobile Agents for WWW Distributed Database Access", Technical Report TR 98-12, Univ. Of Cyprus, Computer Science Department, Sept. 98.

[6] George S and Evaggelia P, "Computational Models for the Wireless and Mobile Environments", Technical Report TR-98-4, University of Cyprus, Computer Science Department, 2010.

[7] Aridor Y and Lange D, "Agent Design Patterns: Elements of Agent Application Design", in *Proceedings of Autonomous Agents'*98. ACM Press. 1998.

[8] Jim White, "Mobile Agent White Paper", general Magic, 1996.

[9] Satoh I, "Building and Selecting Mobile Agents for Network Management", Journal of Network and Systems Management, vol.14, no.1, pp.147-169, Springer, 2006.

[10] Satoh I, "A Location Model for Smart Environment, Pervasive and Mobile Computing", vol.3, no.2,pp.158-179, Elsevier, 2007.

[11] Satoh I, "Context-aware Agents to Guide Visitors in Museums", in Proceedings of 8th International Conference Intelligent Virtual Agents (IVAb08), Lecture Notes in Artificial Intelligence (LNAI), vol.5208, pp.441-455, September 2008.