MULTI-AGENT MODEL AS AN EXTENSION TO THE CLIENT/SERVER PROCESSING PARADIGM

K. Zieliński, A. Laurentowski, A. Uszok
Institute of Computer Science,
University of Mining and Metallurgy, Cracow, Poland
e-mail: {kz,pinio,uszok}@ics.agh.edu.pl

Keywords: RPC, mobile agents, proxy, CORBA, Agent Meeting Points

Abstract

This paper evaluates advantages and disadvantages of multi-agent systems with comparison to classical client/server paradigm. It presents the idea of extending OMG/CORBA-compliant systems with a mobile agent service. The paper proposes that Agent Meeting Points can be based on CORBA objects infrastructure.

1 Introduction

The most popular paradigm for distributed computing so far, the client/server model, turned out to be insufficient to respond to the challenges incurred by rapidly increasing size and complexity of global computing and information resources. One extension to this model was the proxy mechanism[5], allowing the server to be locally represented at the client's side. This approach, however, has not solved many substantial problems which arose with the growth of the global computer network infrastructure and mobility of users.

We see delegation of function as an important response to these problems, transparently reducing network overheads and usage complexity of widely distributed resources. As opposed to the server-proxy mechanism, we put forward the mobile agent paradigm[1], where an agent represents the client at the server's side. Agents are software entities that migrate among a set of networked servers, carrying out some set of operations on behalf of the client with some degree of independence or autonomy, and in doing so, employ some knowledge or representation of the client's goals or desires.

The paper is structured as follows. First, a brief description of mobile agent-based computing is presented and contrasted to message-passing and RPC – the basic mechanisms for client/server approach. Next, an extension of OMG CORBA-compliant systems with an "mobile agent service" is proposed. Then, its core module – a CORBA-based Agent Meeting Point (CAMP) – an abstraction which supports the interactions of agents with each other and CORBA-based server resources is described. The paper is ended with conclusions.

2 Mobile Agent-Based Computing

The mobile agent concept is illustrated in Fig. 1. A client computer consists of an application environment, which contains one or more applications for interaction with a remote server. When an application needs to perform a transaction, it will assemble information and then pass this via the API into the agent execution environment. This will initiate the execution of an instance of a particular class of agent as a process within the agent execution environment. The agent program may be built from procedural components or from classes of objects. The program may be executed in either machine language or an interpreted language in order to support heterogenity.

The information assembled by the application is accepted by the agent as a part of its initialization and at certain point in its execution, the agent will execute an instruction which has the following effects: (1) Either the current agent process is suspended in the agent execution environment, or new agent child process is created. (2) This process state, stack, heap, and all external references are collected and processed into a message expressed in a machine-independent form, e.g. ASN 1 or SunSoft Java bytecode. If the agent is expressed in an interpreted language, the state data is captured on the stack and there is no need to save registers. This step is further facilitated if the agent is built from object classes. If it is known that the identical classes are resident at the destination, the agent may be reduced to object reference and instance data. (3) The agent message is handed to message subsystem and routed directly or indirectly to the destination server. (4) In the

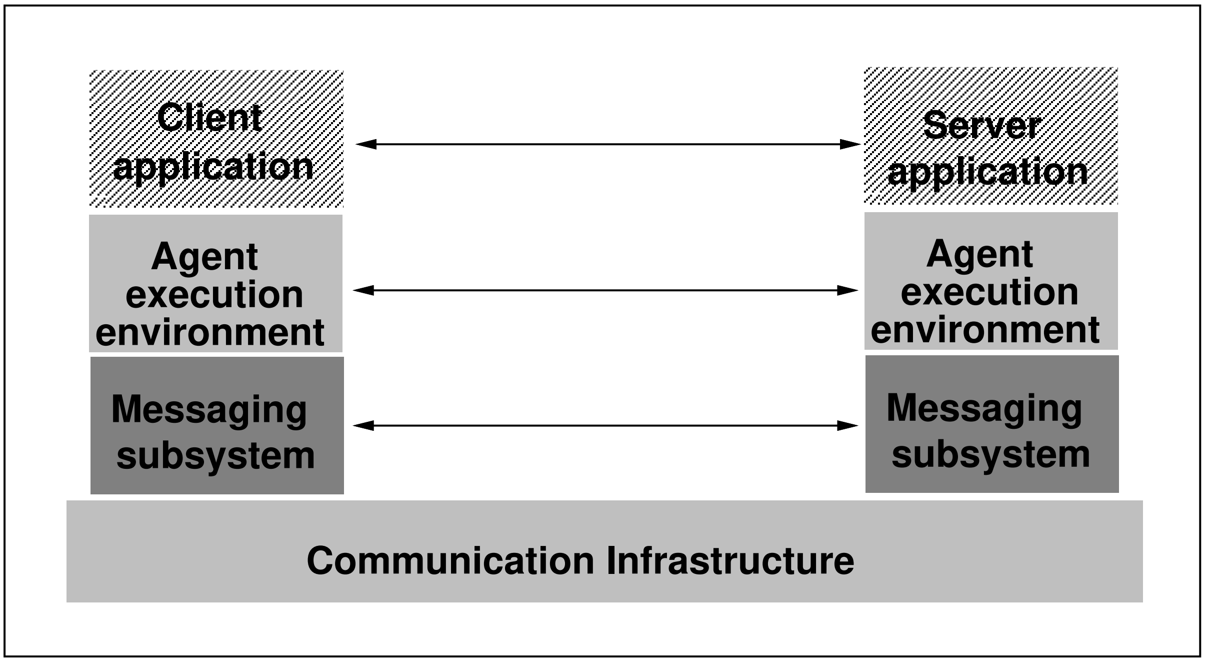

Figure 1: Conceptual model for mobile agent computing[3]

agent execution environment the received message is reconstituted into the executable and the process or thread is dispatched. (5) Execution continues at the next instruction in the agent program.
This is effectively a process migration that moves the agent from a client which has a request to a server which is capable of satisfying it.

The most popular paradigm for client/server computing is built around RPC mechanism. The primary distinction between RPC and mobile agents is that the former mechanism is strictly synchronous while mobile agents provide inherently asynchronous interactions. Next, RPC communication requires a stub implementation that make service invocation possible only when the server is equipped with a corresponding skeleton routine. It makes client/server relation very static that improves performance. And last, but not least, the client and server of a RPC call are totally separate entities. The proxy paradigm[5] was proposed as an extension to RPC, adding a server's representative at the client's address space. It may, however, be inefficient in an environment with a vast number of rapidly emerging services and resources.

Detailed analysis of RPC and mobile agents leads to the conclusion that nearly everything that can be done with mobile agents, can also be done with RPC (with remote real-time control as a possible exception).

The technical advantages of mobile agents include: high bandwidth remote interaction (effectively), support for disconnected operation, support for weak clients, ease of distributing individual service clients, semantic routing, scalability, lower overhead for secure transactions, robust remote interaction.

While none of the individual advantages of mobile agents given above is overwhelming strong, we believe that the aggregate advantage is really strong. While alternatives to mobile agents can be advanced for each of the individual advantages, there is no single alternative to all of the functionality supported by a mobile agent framework. This framework enables new networks services and hence new businesses, so it may appeal strongly to the Internet community. It can provide an effective means for dealing with the problems of finding services and information.

The well-known technical disadvantages of mobile agents are: a need for highly secure agent execution environments, performance and functional limitation resulting from security, virus scanning and epidemic control mechanisms, transmission efficiency.

3 Agent Meeting Point Structure

It is our belief that an agent infrastructure will be most naturally expressed in terms of a set of object services. Objects form a coherent, concrete element of mobility. We can move an entire objects and we can refer to remote objects through proxy objects. We can apply authentication and access controls more readily than to procedural data structures. By using a distributed computing framework such as OMG/CORBA[4] we can delegate a large number of distributed computing issues to this framework.

In general, a mobile agent environment may be structured as a network of Agent Meeting Points (AMPs). AMP[2] plays as a broker between the agents, which are making requests for resources and services, and the servers which implement these resources and services. The basic structure of the AMP is a set of OO classes designed to provide a very lightweight, minimal framework for building specific meeting points.

The functional decomposition of AMP is shown in Fig. 2, with the components comprising:

Base Objects Services are CORBA-based object services or their equivalent. Included are object interface bindings, remote object invocation, marshaling of objects, object-based access control services, persistence and replication, etc.

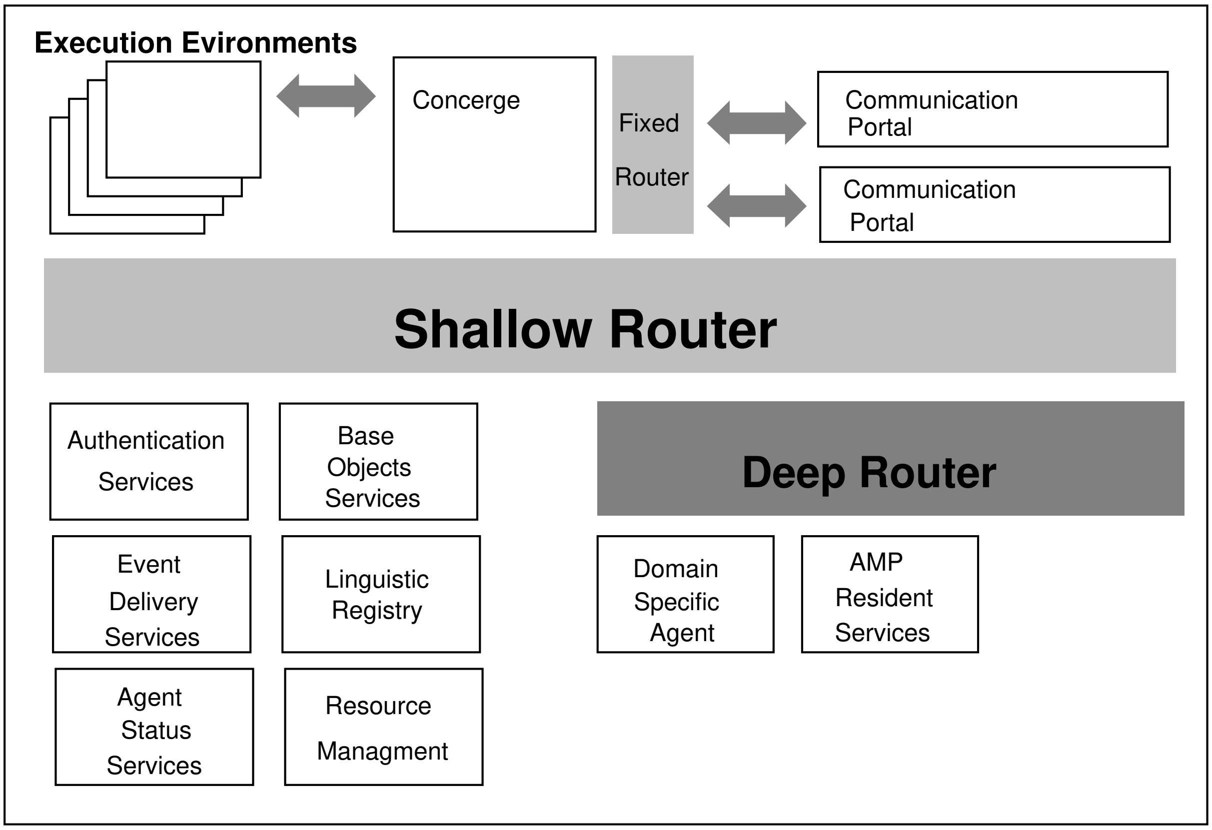

Figure 2: Agent Meeting Point (AMP) - general concept

Communication Portals are responsible for managing the arrival and departure of mobile agents as well as routing messages to the residing agents or services in the AMP. The Communication Portals extract the arriving mobile agent and pass it to the agent Concierge.

Concierge examines an agent's credentials and makes sure that the facilities the agent has requested are available.

Authentication Services are an OO encapsulation of the underlying distributed authentication scheme being used in the agent network.

Shallow Router forms a glue which binds together the AMP. Components of the AMP register themselves to the Shallow Router with a description of services they perform. The Shallow Router then uses this association to route requests from agents to the components which support them.

Linguistic Registry is essentially a database which keeps track of which languages are known and the services, in the AMP, that know these languages.

Deep Router assists the Shallow to deal with more special or difficult requests, providing an extended directory service for them.

Resource Manager acts as a registry for all of active agents and manages the use of resources within the AMP.

Agent Execution Environments are environments which have registered to the AMP, offering to interpret scripts or agents whose encoding they support.

Event Delivery Services are concerned with conveying information from services within the agent AMP to agents which have expressed an interest in this information.

The mobile agent's arrival proceeds in the following steps: (1) Communication Portal receives the agent and assembles it, (2) Concierge makes origin authentication, data integrity check and optional decryption of the agent's body, (3) Concierge registers the agent to the Resource Manager, (4) Concierge examines the agent's table of contents (describing the components of the agent) and passes each of them to the Shallow Router, (5) The Router matches each component's request with the servers available at the AMP, passing more exotic ones to the Deep Router, (6) the agent's elements are then distributed to the services elements which support them and their execution starts.

4 AMP in CORBA-compliant Environment

The architecture of the AMP described in the previous section is very general. We believe that mobile agents processing is natural extension of requests brokering in CORBA compliant systems and it should finally lead to a definition to a future Common Networked Agent Architecture (CNAA) standard. CNAA building blocks and CORBA should complete each other and create the uniform platform for future global network distributed systems.

The standardization process should, however, be advanced by many research prototypes. We are currently developing one of them, called CORBA-based Agent Meeting Point (CAMP). It is based on following assumptions:

1. CAMP is implemented as a CORBA-compliant server which provides the following objects: Communication Object (CO) that encapsulates Communication Portal and Concierge functionality, Authentication Object (AO), Router Object (RO) that combines functions of Shallow and

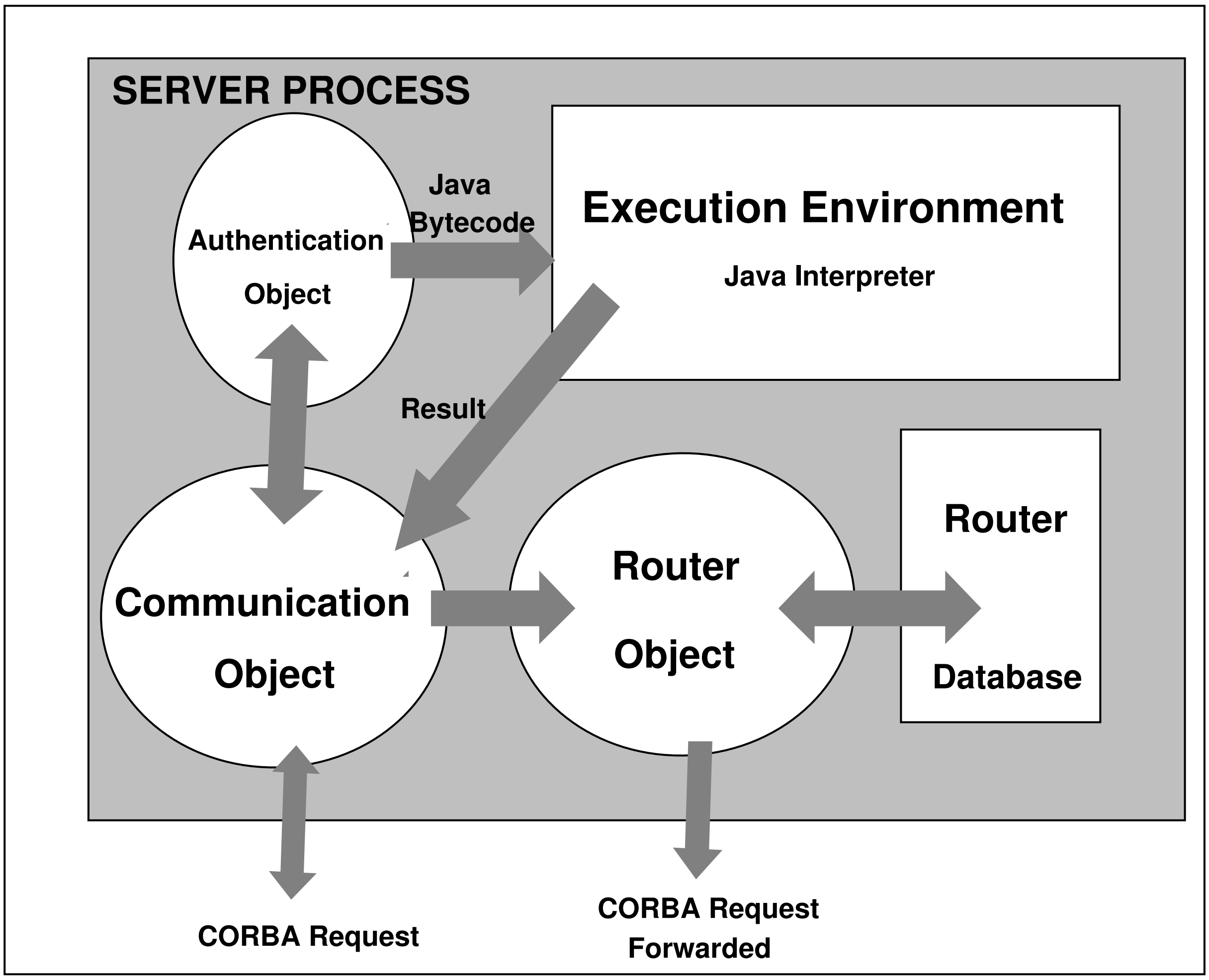

Figure 3: CAMP: a CORBA-based AMP

Deep Routers, and finally Execution Environment Object (EEO). The proposed architecture is shown in Fig. 3.

2. Each mobile agent is encapsulated in a CORBA request and is sent as parameters of the Communication Object invocation.

3. Mobile agent functionality is represented by SunSoft Java bytecode[6]. Thus, the Execution Environment Object is an OO encapsulation of Java interpreter.

In case when the services or resources desired by the agent are not available at the CAMP, the Router Object tries to find another CAMP (not necessarily guaranteeing final execution), using a special database and routing protocol. If successful, the Router Object forwards the request to the chosen CAMP's Communication Object.

The area of problems arise from the connection-oriented nature of CORBA-compliant systems, while the mobile agents can roam among CAMPs. From a CORBA point of view, the client must be informed that its request was forwarded to another server.

To make the proposed scheme useful, the CORBA user programming interface should also be extended with a possibility to declare and pass a bytecode of a mobile agent as an operation parameter. Hence, the client may chose between standard RPC-style object operation invocation or mobile agent dispatching by the CO's interface operation call.

5 Conclusion

The presented analysis shows that mobile agents must be seen as a very important alternative to RPC, giving more freedom to the programmer and assuring more efficient usage of global network resources. We believe that the proposed implementation of the CAMP, encapsulating mobile agents functionality over CORBA architecture, is the first step to a future Common Networked Agents Architecture development.

References

[1] P. MAES, *Agents that Reduce Work and Information Overload*, Comm. ACM, Vol.37, No.7, July 1994.

[2] D. CHESS, B. GROSOF, C. HARRISON, D. LEVINE, C. PARRIS, G. TSUDIK, *Itinerant Agents for Mobile Computing*, IBM Research Report RC 20010, to be published in *IEEE Personal Communications*, March 1995.

[3] C. HARRISON, D. M. CHESS, A. KERSHENBAUM, *Mobile Agents: Are they a good idea?*, IBM Research Report RC 19887, October 1994.

[4] Draft Common Object Request Broker Architecture Revision 1.1, OMG Report 91-12-1, OMG Inc., 1991.

[5] M. SHAPIRO, *Structure and Encapsulation in Distributed Systems: the Proxy Principle*, *Proceedings of the 6th IEEE ICDCS*, Boston, May 1986.

[6] The Java Language - a White Paper, SunSoft Inc., 1995.