NOVI SAD J. MATH.
VOL. 30, NO. 2, 2000, 113-123

Proceedings of the TARA 2000 Conference
Novi Sad, Yugoslavia, September 6-7, 2000

MOBILE AGENTS - A NEW AND ADVANCED CONCEPT?

Zoran Putnik[1], **Zoran Budimac**[1]

Abstract. Interest in network-centric programming and applications has surged in recent years thanks to the exponential growth of the Internet and the widespread popularity of World Wide Web. In response, new techniques, languages, and paradigms have evolved to facilitate the creation of such applications. One of the most promising is the paradigm of mobile agents. Mobile agents are a new approach to the architecture and implementation of distributed systems. A mobile agent is a program that moves around a network and can communicate with its environment and other agents. Possible applications for mobile agents include network management, information retrieval, distributed simulation, electronic commerce, and mobile computing.

AMS Mathematics Subject Classification (1991): 68N25

Key words and phrases: mobile agents, internet, network, distributed systems

1. Introduction and possible definition of mobile agents

Mobile-agents technology originally emerged from advances made in distributed systems research. They are also known as itinerant agents [2], transportable agents [9], or re-locatable objects [7]. Still in the phase of development and introduction to the wider audience, mobile agents lack generally accepted definition and commonly defined features. Still, it is rather easy to find a lot of encouragement for the development of systems based on mobile-agent concept. In [11], a few features are distinguished as the primary advantages of mobile-agents:

- they facilitate high quality, high performance, economical mobile applications;
- they enable use of portable, low-cost, personal communications devices;
- they permit secure Intranet-style communications on public networks;

[1]Institute of Mathematics, Faculty of Science, University of Novi Sad, Yugoslavia, e-mail: {putnik,budimac}@unsim.ns.ac.yu

- they efficiently and economically use low bandwidth, high latency, error prone communication channels, and last and most important
- no application-level protocol is created by the use of agents. Therefore, compatibility is provided for any agent-based application.

Mobile agents may be easily connected to a previously known technique for client-server computing - remote procedure calling methodology. This approach enabled one computer to invoke a procedure at some other machine, send it a set of data to be processed and wait for a result before sending the next set of data. So, each message exchange involved a request for service or an acknowledgment of information receipt. Before beginning this process, computers involved had to agree upon the format of the exchanges, establishing protocol. Consequently, this method required persistent connection, with very high communication overhead [12].

Significantly different, mobile-agents method supplies not only the data to be processed, but also the procedures to be performed as well. Key distinction lies in the fact that once the agent has been exchanged between client and server, "it" can act without further help from the network. Unlike remote procedure calling, mobile-agents do not require ongoing communication, thus reducing overall network load. As much as the "exchange protocol" is concerned, for a mobile-agent to be executed on the different host environments, each host must have specialized support software, which will be explained later in more detail.

A term computational agent (mobile or not) has its background in the early work in the field of artificial intelligence, in the time when researchers concentrated on creation of artificial "entities" mimicking human abilities [6]. The term was applied to a wide range of entities - software systems, as well as autonomous robots or biological organisms. The agenthood concept was summed up by the following definition [4]: "An agent is a computational entity which: acts on behalf of other entities in an autonomous fashion, performs its actions with some level of proactivity and/or reactiveness and exhibits some level of the key attributes of learning, co-operation and mobility." In simple words, a need for software agents emerged because:

- more and more everyday tasks are computer-based, while
- increasingly more users are untrained, and therefore;
- users require agents to assist them in understanding the technically complex world created.

Mobile-agent based computing may be viewed as an extension of two well-known methods: remote dispatch of script programs and remote submission of batch jobs. Or, it can be considered a generalization of the client-server paradigm of distributed computing. Also, it is necessary to mention other alternatives earlier developed in the area of client-server interaction: messaging,

simple datagrams, sockets, remote procedure calls, and conversations, for example.

Let us now try to present in brief some of the definitions that can be found in recent research of mobile-agents and propose the best of them, show and explain some of the most often needed characteristics, and also point out to some of the most common misunderstandings and amplifications about mobile agents.

A technical, strict and precise definition of a mobile agents we came across is the following [5]: "Mobile agents are programs, typically written in a script language, which may be dispatched from a client computer and transported to a remote server computer for execution." The same, but shorter: "A mobile agent is a program that can migrate from one computer to another for remote execution" [16]. An AI-based definition can be found in [14]: "Agents are computer programs that simulate human relationship, by doing something that another person could otherwise do for you." Another interesting "working" definition of an agent [10], less philosophical and more technical in nature, is: "... (an agent is) a program that helps a user to perform some task (or set of tasks), possibly by maintaining persistent state and communicating with its owner, other agents or its environment in general."

Concentrating only on mobile agents, a good definition is given in [3]: "A mobile agent is an autonomous program that can migrate under its own control from machine to machine in a heterogeneous network. In other words, the program can suspend its execution at an arbitrary point, transport itself to another machine, and resume execution on the new machine from the point at which it left off. On each machine, it interacts with service agents and other resources to accomplish its task." A practical, commercial definition may be found in [11] claiming that ". an agent is an independent software program which runs on behalf of a network user. . (and so) . in addition to being an independent program executing on behalf of a network user, it can travel to multiple locations in the network."

Still, as the best one, we suggest the following definition that covers most of important aspects of mobile-agent technology, while still remains short and understandable. Since in a broad sense, an agent is any program that acts on behalf of a user, then "... a mobile agent is a program that represents a user in a computer network and can migrate autonomously from node to node, to perform some computation on behalf of the user" [8].

Through a "historical retrospec", distributed applications have traditionally relied on the client-server paradigm, in which client and server processes communicate either through message-passing or remote- procedure calls. In this model, an operation is split into two parts across the network, with the client making requests from a user machine to a server who services the requests on a large, centralized system. A protocol is agreed upon and both the client and server are programmed to implement it. A network connection is established between them and the protocol is carried out. This communication model is usually synchronous: the client suspends itself after sending a request to the server,

waiting for the results of the call. The client-server model has the advantage of enabling the removal of the client to smaller remote machines, and it works well for certain applications. However it breaks down under other situations, including highly distributed systems, slow and/or poor quality network connections, and especially in the face of changing applications. In [15], an alternative architecture is proposed called remote-evaluation (or, remote-programming), where instead of invoking a remote procedure, the client sends its own procedure code to a server, requesting that the server execute it and return the results. Two computers, whose communication follows the remote-evaluation approach, agree in advance upon the instructions that are allowed in a procedure and the types of data that are allowed in its state, constituting a language.

Unlike the standalone applications that popularized the personal computer, the communicating applications that will popularize the personalized usage of a network have components that must reside in servers. The server components of remote-procedure-call-based applications must be statically installed by the user. The server components of a remote-evaluation-based application, on the other hand, are dynamically installed by the application itself - most likely, each one being an agent.

2. General characteristics of mobile agents

Information search and filtering applications often download and process large amounts of server-resident information while generating comparatively small amounts of result data. Using mobile agents instead, which execute on server machines and access server data without using the network, reduces the bandwidth requirements. Some applications involve repeated client-server interactions, thus requiring either maintaining a network connection over an extended period or making several separate requests. With mobile agents, the client need not maintain a network connection while its agent's access and process information. This permits regular disconnection between the client and server, saving time, space, and energy for example, while not losing any piece of information.

There can be distinguished five essential characteristics of agents [12], [4]: mobility, response method, autonomy, learning and cooperation. Let us briefly explain each of them:

Mobility refers to agent's ability or inability to move among different environments. A static agent is the one that is constrained to work in a single type of a machine and is unable to move through the network. An agent is mobile if it is able to move from host to host AND to work in a variety of host environments.

Response method refers to the means that an agent uses to react to a requests - classifying them either as deliberative or reactive. Deliberative agent has a model of reaction for every possible environment in which it can exist. Reactive agent, on the other hand, has preset ways in which it can react.

Autonomy is a characteristic defining whether agent can operate on user's behalf without explicit guidance.

Learning is a feature that defines if an agent can take whatever information it gained about the environment in which it is deployed and use it to modify its own behavior.

Cooperation among multiple agents represents the ability of avoiding redundant efforts of several agents by conducting communications between them.

Agents' intended role influence highly its characteristics - an interface agent needs both autonomy and ability to learn, and does not need mobility, while collaborative agent requires autonomy, cooperation and mobility as very desirable features.

The most important consideration for agents is that a program code should run in an identical fashion on every host it can move to. To fulfil these requirements it is much easier to use programming languages that are either interpreted directly or compiled to a portable intermediate interpreter-based language that can then be easily transported and executed without recompilation. For both approaches, only the (intermediate-code) interpreter must be ported once to allow for the direct execution of all the programs written in that language. Interpreting the program as opposed to executing it in a system's native machine code naturally slows the execution speed a little bit. Yet the benefits of easier mobility, the ability to offload the work into local libraries running at the full speed and decrease of the speed difference between compiled and interpreted code execution at modern, strong machines, outweighs the problems. Currently, there is no consensus about which programming language is "best" for mobile-agents. Still, in [16], an excellent characterization of a good mobile-agents programming language(s) is given: "The characteristics that make a language useful for writing mobile agents are: support of agent migration and agent-to-agent communication; ability to allow agents to interact with local resources; security mechanisms; execution efficiency; language implementation across multiple platforms, and the language's ease of programming of the tasks mobile agents performs.

Besides its code, an agent carries persistent state that allows "it" to take up its work after a move to another host at the point where it left off before moving. Yet, the last part of a previous sentence is not necessarily true - whether this is actually possible, depends on the implementation language of the agent. As will be explained in more detail latter, Java, most widely used programming language for mobile-agents construction, does not provide mechanism for continuing "...at the point where an agent left off before moving."

An important question can be raised about how an agent picks a host to move to. There is a simple case where mobile agent carries along an itinerary of places that "it" has to visit. It is of course much more interesting to let the agent modify its destination according to the "discoveries" it has made during its previous searches, creating a really helpful artificially-intelligent assistant. Also, in the case when an agent is moving through a trustworthy network, we

can use help from a server - a server could advise an agent about interesting places in the "neighborhood".

Considering all being said in this section, we can conclude that a mobile agent is hardly a program - it requires an environment on potential hosts to run on, an agent server. This agent server acts like an operating system for mobile agents and is responsible for [16]:

- providing an environment for the agent to run in;
- transferring and receiving agents to and from different agent servers;
- implementing an application programming interface (API) for messaging between agents and agent transfer requests;
- protecting the host computer from hostile mobile agents, and
- interpret the programming language the agent is written in.

3. Problems and concerns of today systems

One of the main concerns about mobile agents lies in the area of security, since the most important goal of the research is to enable unconstrained electronic commerce, electronic-mail communication and searching through far-away data-bases - or, in one sentence, communication between two parties without prior definite agreement. So, the main problem with the mobile-agents usage lies in a question whether the benefits they brought compensate for the concerns they raise.

Security in agent systems consists of two parts, the security of the system against a possibly malicious and dangerous agent and the security of an agent against a system [1]. Other possible problems include reducing migration and communication overhead, limiting an agent's total resource consumption, isolating agent against possible machine or network failures, and developing the techniques for resource discovery, network sensing, navigation and planning services, that will allow the agent to identify and reach the desired services [3].

If the agent is larger than the intermediate data, it will obviously consume more bandwidth than in traditional client-server implementation. Besides, if the network between client and server has high bandwidth and low latency, for example, time needed for the server to start up the necessary execution environment for the incoming agent becomes a significant part of the whole communication time. Finally, since the agent will presumably use a lot of processing time at the server, if the server is low-powered, heavily loaded, or highly attended by agents, a CPU- intensive agent can easily take longer to process the task than the corresponding cross-network calls. Taken together, all of the mentioned leads to a single conclusion: mobile-agents present a much greater performance advantage as network bandwidth decreases, network load increases,

the amount of intermediate data increases, and server power increases [3]. If conditions move in the opposite direction, mobile agents can display a severe performance disadvantage.

Second existing problem concerning mobile-agents system is a problem of reduction of communication costs. No matter how it looks like, agent technology does not reduce communication cost by itself. Yet, in certain situations, mobile programs/agents may reduce this cost considerably. Two types of reduction can be distinguished: reduction of the number of remote interactions and reduction of the amount of data communicated over the network [13]

The first type of reduction can be achieved by bringing two entities that highly interact with each other to the same location. For the second type of reduction a client/server relationship in which a client includes a filtering function for the data retrieved from the server. If this filter function (or the entire client) is moved to the server, the function is performed before the data is communicated over the network. Of course, moving an agent is not for free. An overall cost reduction is only obtained if the performance gains exceed the extra overhead for transferring agents. So far, the biggest concern with mobile agents is security. Neither the agent, nor the machine(s) are necessarily trustworthy. The agent might try to access or destroy privileged data, or use more than allowed and paid of some resource. The machines might try to pull privileged data out of the agent, modify data or code of an agent, thus changing its behavior, or try to collect more money (real or electronic) from an agent for non-given services. A system that does not detect and, more importantly, prevent such malicious actions, can never survive as a real-world application. In an open network, intentional attacks on both agents and machines are cruel reality, but even in a closed network with trusted users, mis-programmed agents can cause significant damage accidentally.

Being the most critical and yet unsolved issue in a mobile-agents system, security requires more attention, and is divided into four different, complementary and interconnected issues: machine protection, protection of an agent, protection of other agents and a protection of a group of machines - an agent might consume excessive resources in the network as a whole, even if it consumes a few resources at each machine.

It is important to be aware that the level of trust to an agent may differ considerable depending on the agent. For instance, a locally developed agent is going to be rather more trustworthy than some random, unknown piece of software which has just arrived over the net.

Currently, there exist four research directions in the field of mobile-agent security:

- the organizational approach - eliminates the problem by allowing only trustworthy institutions to run mobile-agent systems, and does therefore not allow open systems;
- the trust/reputation approach - allow agents to migrate only to trusted

hosts or such with good reputation - which is again a restrictive term for the openness of the system;

- the manipulation detection approach - offering mechanisms to detect manipulation of agent data or the execution of the code, but does not protect against read attacks, and
- the blackbox protection approach - which "puts a code in a blackbox", which forces attacker to spend time to analyze the code, and thus protecting a code for a certain time interval. After this period, an agent and the data it is carrying becomes invalid. Still, all of these approaches are subject of ongoing work, none of them is currently used in real-world commercial applications.

4 Why exactly mobile agents

Mobile agents are best viewed as a tool for implementing distributed applications, rather than as an enabling technology. In other words, their advantage lies not so much in making new distributed language possible, but rather in providing a unified programming model and improving the performance of existing applications. While there still may be defined a case for static agents, in circumstances where static agent passes data from one site to another for processing by the agent already resident at the second, target site, a hidden problem prohibits this approach - the target site is rarely known in real-time environments. Further, static agents require that all participant nodes have support for all agent messages - which is highly unrealistic and unfeasible.

Mobile agents have several advantages. By migrating to the location of a needed resource, an agent can interact with the resource without transmitting any intermediate data across the network, significantly reducing the bandwidth consumption in many applications [3]. In addition, by moving to the location of a user, an agent can respond to user actions rapidly. In either case, the agent can continue its interaction with the resource or user, even if the network connection goes down. For stationary clients and servers, agents offer the possibility to offload work to each other, according to machine capabilities and current loads.

What would be characteristics of an ideal mobile-agents system? Consulting the latest research papers and empirical studies, the overall goal would be an efficient, robust and secure mobile-agent system, allowing the programmer to choose most appropriate among several programming languages for a given task and fast-enough develop any-scale distributed application. Such a system provides (or uses existing):

- efficient and secure "transport mechanism", i.e. communication protocol;
- server performing the following tasks: keeps status of running agents and provides their administration, enable acceptance and authentication of in-

coming agents, allows communication between agents and provides means for storing an agent's state and its restoration in event of machine failure;

- provides language-independent core, connecting agents to a server and providing necessary operations;
- provides interpreters for each language, enabling addition of new languages, as necessary.

The top level of this architecture would, naturally, consists of agents themselves.

5. Summary

Mobile agents are in the process of "graduating" from being limited to research systems to being a practical technology in network computing. The main advantage of mobile agents is that they can bring a program closer to the information resources. The mobile agent makes the use of the server's basic services in the way and quantity that its owner intends. Mobile-agents provide no new functionality that cannot be achieved with traditional client-server interaction, however, they make implementing any new functionality much easier.

Right now, a computer is merely a passive entity waiting to execute specific, highly detailed instructions, it provides (relatively) little help for complex tasks or for carrying out actions that may consume a large proportion the user's time. Consequently, researchers and software companies have put high hopes on these so-called software agents, which "know" the user's interests and can act autonomously on their behalf. Instead of exercising complete control, people will be engaged in a cooperative process in which both human and computer agents initiate communications, monitor events and perform tasks to meet the user's goal.

Also, under certain conditions, it is more efficient for an agent to remain stationary and interact with a resource from a remote location, rather than migrate to that resource. Unfortunately, these conditions can not be defined in advance, they depend entirely on the resource and the agent's current task. An agent that should invoke, for example, only a single operation on a remote resource should usually remain stationary, thus avoiding migration overhead. A notable exception would be the operation providing extremely large result, of which the user need only a small piece. Other points affecting migration decision are the machine loads, the relative speed of agent against native code, the reliability, latency and bandwidth of the network links, or the average size of operation result [3]. In the future, supposingly it will also be a question of metering and charging for services. When agent travel through a network, it consumes certain resources such as CPU time and disk space at different servers, which might legitimately expect monetary compensation for providing such resources. So, mechanisms will be available for an agent to carry "digital

cash" and use it to pay for resources it uses, which will in turn influence its decision whether to migrate or not.

For the sake of truth, it must be said that with today's conditions on a conventional network, when compared with traditional client/server implementations, the total task completion time is longer for mobile agents except in low-performance networks [3]. Reasons for this are high migration overhead, slower execution of interpreted languages and need for security checks. While there are a lot of straightforward optimizations to mentioned problems, it is obvious that there will always be conditions under which it will be better for an agent to remain stationary and act like a traditional client. In the same paper, it has been mentioned that certain formulas are developed for an agent to decide when and where to migrate.

If we now take a look at all the advantages and disadvantages mentioned above, we may conclude two important things. First, for every problem, a comparable solution can be found that does not require mobile agents. Yet, if we take a look at all of the advantages together, a fundamental argument emerges [5] " whereas each individual case can be addressed in some manner without mobile-agents, a mobile-agent framework addresses all of them at once". While alternatives to mobile agents can be proved better for each individual case, there is no single alternative to all of functionalities supported by mobile-agents framework. Or, as put in [13]: "...it can be said that a mobile agent system does not offer new mechanisms compared to existing technology, but currently, no existing system offers the same set of advantages and if there would be a such a system, it is a mobile agent system since it then has the same functionality."

References

[1] Joachim Baumann, Agents: A Triptychon of Problems, Proc. of ECOOP '95 Workshop "Objects and Agents: Love at First Sight or a Shotgun-Wedding"

[2] Chess, D., Grosof, B., Harrison, C., Levine, D., Parris, C., Tsudik, G., Itinerant Agents for Mobile Computing, Technical Report, IBM T.J. Watson Research Center, 1995.

[3] Robert Gray, Agent Tcl : A Flexible and Secure Mobile Agent System, Doctoral Thesis, Dartmouth College, Hanover, New Hampshire, 1997.

[4] Shaw Green, Leon Hurst, Brenda Nangle, Padraig Cunningham, Fergal Somers and Richard Evans, Software Agents: A review, Technical Paper, Broadcom Eireann Research Ltd. and Trinity College Dublin, 1997.

[5] Colin G. Harrison, David M. Chess, Aaron Kershenbaum: Mobile Agents: Are they a good idea ?, Research Report, IBM Research Division, T.J.Watson Research Center, Yorktown Heights, New York, 1995.

[6] Hewitt, C., Viewing Control Structures as Patterns of Passing Messages, Artificial Intelligence 8 No.3 (1977), 323-364.

[7] Joseph, D., deLespinasse, A., Tauber, J., Giffor, D., Frans Kaashoek, M., Rover: A Toolkit for Mobile Information Access, In the Proceedings of the 15th ACM Symposium on Operating Systems Principles, 1995.

[8] Neeran Karnik and Anand Tripathi, Design Issues in Mobile-Agent Programming Systems, IEEE Concurrency (1998), 52-61.

[9] Kotz, D., Gray, R., Rus, D., Transportable Agents Support Worldwide Applications, In Proceedings of the 7th ACM SIGOPS European Workshop, Ireland, 1996.

[10] Anselm Lingnau and Oswald Drobnik, An Infrastructure for Mobile Agents: Requirements and Architecture, J.W.Goethe Universitat, Frankfurt, Germany

[11] Mitsubishi Electric ITA, Mobile Agent Computing - A White Paper, Horizon Systems Laboratory, 1998.

[12] Patricia Morreale, Agents on the Move, IEEE Sprectrum, April 1998, 34-41.

[13] Rothermel, K., Hohl, F., Radouniklis, N., Mobile Agent Systems: What is Missing?, Institute for Parallel and Distributed High-Performance Systems, Stuttgart, Germany, 1997.

[14] Selker, T., A Teaching Agent that Learns, Communication of the ACM 37, 7 (1994), 92-99.

[15] Stamos, J.W., Gifford, D.K., Remote Evaluation, ACM Transactions on Programming Languages and Systems, Vol.12, No.4 (1990), 537-565.

[16] Steven Versteeg, Languages for Mobile Agents, Doctoral Thesis, Department of Computer Science, Melbourne University, 1997.