Proceedings of the Second Workshop on Tests and Fault Tolerance (II WTF 2000), 15th-16th july 2000, Curitiba, Brazil

Reliability Requirements in Mobile Agent Systems

Flávio Morais de Assis Silva, Raimundo José de Araújo Macêdo

Laboratório de Sistemas Distribuídos - LaSiD
Departamento de Ciência da Computação
Universidade Federal da Bahia
Campus de Ondina, CEP 40170-110, Salvador – BA, Brazil
emails: {fassis, macedo}@ufba.br

Abstract

A fundamental issue in the development of mobile agent systems is how to provide support for agent applications reliability. For some agent applications areas (such as electronic commerce or workflow) it is fundamental that mobile agent executions are fault tolerant and exhibit transactional semantics or that groups of mobile agents can coordinate their activities with the use of a reliable communication mechanism. This paper discusses reliability requirements in agent systems and introduces mobile process groups as a suitable underlying concept for fulfilling these requirements. Mobile process groups are an extension of the concept of groups in traditional group communication systems that supports mobility of group members.

1 Introduction

A *mobile agent* (or simply *agent*) is a self-contained software element responsible for executing a programmatic process, which is capable of *autonomously migrating* through a network. An agent migrates in a distributed environment between logical ”places“ referred here to as *agencies*. When an agent migrates, its execution is suspended at the original agency, the agent is transported (i.e. program code, data, execution state and control information) to another agency in the distributed environment, and, after being re-instantiated at the new agency, the agent resumes execution.

The concept of mobile agents began to obtain stronger attention of the research and industry community with Telescript [White 94]. In [White94] migrating active objects were introduced as a suitable paradigm for supporting applications on public networks. After Telescript, many different projects at research institutes and in the industry began to appear, among them Mole, TACOMA, Aglets, Agent Tcl and Grasshopper (a brief description and references to these systems appear, for example, in [Assis Silva 99]). The mobile agent concept is being proposed to support different types of applications, including electronic commerce [White 94] [Straßer, Rothermel, Maihöfer 98], workflow management [Cai, Gloor, Nog 96], network management [Fuggetta, Picco, Vigna 98], implementation of telecommunication services [Magedanz, Popescu-Zeletin 96], distributed information retrieval [Fuggetta, Picco, Vigna 98] and active networks [Fuggetta, Picco, Vigna 98]. Mobile agents have been considered a concept that can be explored to provide, among others, the following benefits: better use of communication resources (both in terms of costs and performance); flexible support for disconnected operation; flexibility for the management of software deployment and maintenance; and adequate support for interactions with human users [Assis Silva 99].

A fundamental issue in the development of mobile agent systems is how to support reliability of mobile agent based applications, specially for those applications that will use mobile agents in open environments such as the Internet. For some types of applications (such as electronic commerce or workflow applications) it is fundamental that mobile agent executions are fault tolerant and exhibit

transactional semantics or that groups of mobile agents can coordinate their activities based on a reliable communication mechanism.

This paper discusses first requirements on reliability of mobile agent applications and discusses existing approaches to address them. Afterwards the paper introduces *mobile process groups* as a suitable underlying concept for fulfilling these requirements. More specifically, in section 2 we concentrate on the issue of making mobile agent executions fault tolerant and guaranteeing their transactional behavior. In section 3 we concentrate on the issue of providing reliable forms of agent communication. In section 4 we introduce mobile process groups. Finally, in section 5 we present conclusions.

2 Making Mobile Agent Applications Fault Tolerant and Transactional

2.1 Requirements

Mobile agent fault tolerance requires at least mechanisms for making agents persistent, for reactivating agents after a failure, for recovering the state of the agent activity after a failure, and for reliably transporting agents between agencies. Additionally, the execution of mobile agent-based applications should be able to tolerate *long-term failures* of agencies. When a mobile agent executing an application moves to an agency, it transfers the control flow execution of that application to that agency. While executing at an agency, an agent is completely subject to the execution rules and conditions of that agency. If the agency where the agent is running fails, the execution of that agent remains blocked while the agency is faulty. An agency may remain faulty for a long time. Long unavailability periods have the obvious undesirable effect of delaying the execution of an agent-based application.

Making the execution of an agent-based application tolerate long-term failures of agencies is achieved by *replicating* agents. With replication, an agent execution is seen as being performed in a sequence of *stages* (see Figure 1). The first stage begins when the application execution starts. A new stage then begins (and the previous terminates) when the mobile agent execution reaches a movement operation, to continue at a new agency. To provide agent fault-tolerance, copies of the agent are sent to a non-empty *set of agencies*.

Consider in Figure 1 that the agent application has to perform actions sequentially at agencies *agency_1_1*, *agency_2_1*, ..., *agency_m_1*, i.e., at the uppermost agencies at each stage represented in the figure. As can be seen, at each stage copies of the agent are sent to the desired agency and additionally to a set of *n-1* other agencies. For example, at *stage 1*, copies of the agent are sent to agencies *agency_1_1*, *agency_1_2*, ..., *agency_1_n*. The *n-1* additional agencies are used primarily for performing exception handling. The execution of the agent application at these agencies, however, can also include accesses to services. When the execution terminates at a stage, copies of the agent with the current state of its execution are sent to the agencies of the next stage and the copies at the current stage are destroyed. An agent application is performed then now by a moving *agent group*, instead of by a single agent.

A group of agents can, during its execution, create other groups of agents, so that each agent group can execute a part of the same global application asynchronously, moving independently one in relation to the others. Each created agent group by its turn can create other agent groups, resulting in a *tree* of mobile agent groups. Fault tolerance must also be guaranteed for these cases. This implies, among other aspects, the need for guaranteeing atomic creation of mobile agent groups and that there must exist mechanisms for coordinating sets of independently migrating agent groups executing the same whole application.

Additionally, in order to use mobile agents in environments such as the Internet, the developed fault tolerance mechanism must be as less restrictive as possible in relation to its assumptions about the environment (for example, in relation to time). In particular, network partitioning should be supported by the mobile agent fault tolerance protocols.

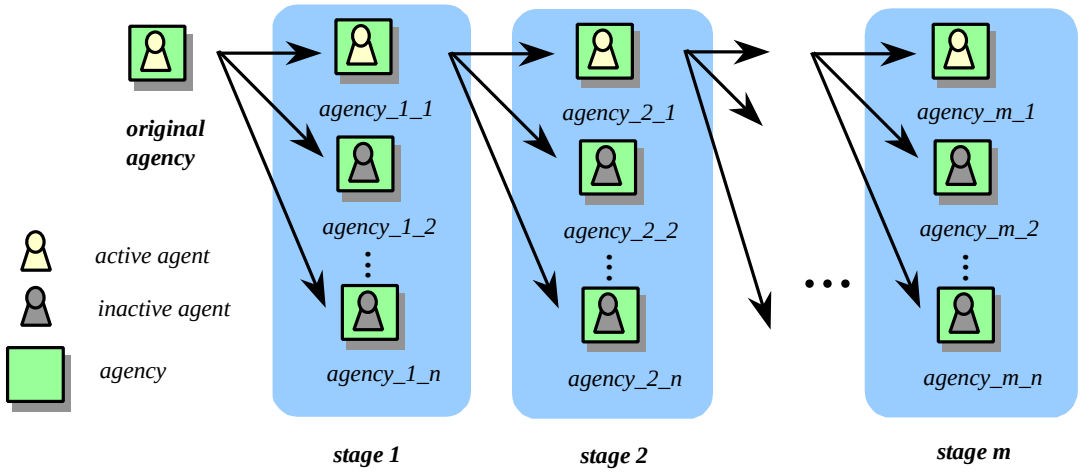

Figure 1: The general agent group execution model

Beyond fault tolerance, agent applications might require the execution of agents to exhibit a transactional behavior. For such applications, it is necessary that accesses to a subset of the services available in a distributed environment can be combined as an unit of work that executes correctly and reliably in the presence of concurrency and failures.

2.2 Existing Approaches

Mobile agent fault tolerance and support for transactional behavior have been considered in the context of some mobile agent projects. In [Schneider 97] [Johansen, van Renesse, Schneider 95] [Johansen, Marzullo et al. 98] [Rothermel, Straßer 98] [Assis Silva 99] the authors consider agent replication as introduced in the previous section. In [Schneider 97], the author concentrates on the problem of protecting agents against being subverted by malicious hosts. In [Johansen, van Renesse, Schneider 95] the authors introduced the so-called *rear guards* as agents left behind while an agent execution moves from agency to agency that perform recovery if some failure causes the agent executing the activity to vanish. The protocol used for managing rear guards is called the *NAP protocol* [Johansen, Marzullo et al. 98]. The NAP protocol is an efficient protocol based on reliable broadcast. The NAP protocol, however, does not tolerate network partitioning. In [Rothermel, Straßer 98] a specific protocol was described for mobile agent fault tolerance that considers network partitioning. In this protocol, however, an agent group cannot create new agent groups. This protocol is coupled with the execution of a single atomic transaction by an agent at an agency. If a new agent is activated to perform recovery of a failed agent, the state of the execution backtracks to the point before the beginning of the execution of the stage. In [Assis Silva 99] an approach for mobile agent fault tolerance and a transaction model (*open nested*) based on mobile agents are presented. The approach supports the dynamic creation of agent groups by other agent groups, recovery from long-term failures of agencies and network partitioning. This approach, based on the use of transactions, fulfills the requirements defined in the previous section.

Another approach for agent fault tolerance is presented in [Dalmeijer, Rietjens et al. 98], but it does not tolerate long-term failures of agencies. In [Vogler, Kunkelmann, Moschgath 97] and [Assis Silva, Krause 97] approaches for executing transactions with mobile agents are presented, but no concept for fault tolerance is discussed.

Finally, approaches for providing mobile agent persistence and reactivation are common in existing mobile agent systems. A mechanism for reliably transporting an agent from an agency to another can be performed, for example, with the use of persistent message queues [Rothermel, Straßer 98].

3 Reliable Coordination of Agents

Components taking part in a distributed application need some form for coordinating their activities. Different applications impose different requirements on coordination mechanisms, resulting in a set of these mechanisms being developed for traditional distributed systems. Beyond direct communication in the form of RPC/RMI, mechanisms such as events, tuple spaces and group communication systems are commonly supported [Tanenbaum 95].

A set of mobile agents can also be used to execute a given distributed application. Not only as a set of replicas cooperating to achieve application fault tolerance, but also as a set of distinct agents cooperating in achieving a certain application goal. Multiple agents executing each asynchronously a part of the same global application (each agent being replicated for fault tolerance or not, as described in section 2) is an example. These agents obviously also need forms of cooperation. Like traditional distributed applications, different mobile agent based applications impose also different requirements on coordination mechanisms. Mobile agent based applications, however, impose an added complexity: the coordination mechanisms must support agents interactions despite their mobility. Mobile agents can even exhibit high mobility, in the sense that a single mobile agent can make many hops while executing an application and interacting with other components. Traditional communication mechanisms must in general be extended to support migration, specially if *migration transparency* should be supported. In a system providing migration transparency, two agents can continue communicating irrespective of the migration of any of them.

Existing mobile agent systems vary greatly in the provided forms of communication. The supported communication mechanisms can be divided into two groups: *direct communication*, where the communicating peers explicitly identify themselves; and *anonymous communication*, where the sender does not know the identity of the recipients. Direct communication can involve either only two peers or a sender and a set of receivers (*multicast*). The form of multicast supported is unreliable. Forms of anonymous communication being supported are *events*, *spaces* and *tuple spaces* [Assis Silva 99].

Some support for reliable communication among mobile agents exists [Murphy, Picco 99]. However, this work describes an algorithm for guaranteeing the delivery of a message to a mobile agent despite its movement in the absence of faults. Support for ordering of messages sent to a group of agents is not addressed. Systems supporting group communication functionality for mobile agents, comparable to those provided by traditional group communication platforms [Renesse, Birman, Maffeis 96][Amir, Dolev et al. 92], are still missing.

4 Common Denominator: Mobile Process Groups

As discussed in the previous sections, fundamental to the development of reliable mobile agent based applications is the existence of appropriate support for mobile agent fault tolerance and mechanisms for reliable communication between groups of mobile agents. A common requirement of both problems is the need for coordinating reliably the actions of members of a group of migrating agents. We intend to address this requirement in a generic way by developing the notion of a *mobile process group.*

A process group is an entity to which an application process refers without knowing the number and location of the members which form it. Essentially, in a group communication mechanism, a message sent by a process must be addressed to all processes in the group and message delivery must be atomic: either all processes receive the message or no one receive it. Additionally, message delivery order guarantees such as total or causal ordering can also be provided by the group communication system [Birman 93]. A mobile process group is an extension of the traditional concept of a group of cooperating processes which can directly support migrating processes as members of the group.

Group communication systems have been traditionally used to support fault tolerance of replicated components in a system (e.g., replicated data bases) and for supporting groupware

applications. The coordination requirements of replicas of mobile agents were made clear in section 2, where replication was presented as a means for addressing mobile agent fault tolerance. The case where not all cooperating agents are replicas of the same agent was also exemplified in section 2 and discussed in the previous section. Beyond the needs for maintaining consistency of replicas or independent groups of replicas, mobile process group services have a broader applicability in agent applications, specially in providing message delivery guarantees (reliable multicast, causal and total order) for any set of cooperating agents (for example, for guaranteeing atomic multicasts to a group of agents that are looking for some information in the distributed environment).

Although the model presented in [Assis Silva 99] fulfills the requirements on fault tolerance of mobile agent applications as described in section 2, this model is based on transactions. Using mobile process groups represents a form of unifying the underlying support for fault tolerance and agents coordination, simplifying the existing solution and potentially increasing the level of fault tolerance and the implied costs. In relation to support for agents coordination, mobile process groups represent functionality not yet present in agent systems.

To the best of our knowledge, support for mobile process groups is not present in existing group communication platforms, such as Horus [Renesse, Birman, Maffeis 96] and Transis [Amir, Dolev et al. 92]. Previous work exists in supporting group communication for mobile device computing (e.g., [Badache, Hurfin, Macêdo 99][Yen, Huang, Hwang 97]), but it is devoted to accommodating existing protocols to the specific properties of the environment (low bandwidth links, low capacity devices, etc.). Some of the results might be applicable to mobile process groups. We are however extending traditional group communication concepts to incorporate new semantics of message delivery in the presence of mobility, not present in those systems. Additionally, special handling of events is being considered to provide a finer tuned way of handling with movement.

5 Conclusions

In this paper we presented first a general overview of requirements for supporting reliability of mobile agent-based applications. Support is needed for mobile agent fault tolerance, for guaranteeing transactional behavior of mobile agent applications and for providing reliable forms of mobile agents coordination.

We introduced afterwards a new concept, called mobile process groups, as a means for fulfilling requirements of mobile agent applications and for improving existing approaches. Mobile process groups are an extension of the group concept supported in traditional group communication platforms and represent a common basis for supporting mobile agent fault tolerance and reliable coordination of mobile agents.

A mobile process group service, which provides message delivery guarantees and synchronization in the presence of mobility, is under development at LaSiD/UFBA. The concepts being developed try to incorporate the mobility of processes in the group communication protocols, thus making it possible to synchronize a process movement action with the delivery of messages and enabling a finer tuned failure detection mechanism.

Bibliography

[Amir, Dolev et al. 92] Y.Amir, D.Dolev, S.Kramer, D.Malki. Transis: a Communication Subsystem for High Availability. in *Proceedings of the 22nd International Symposium on Fault-Tolerant Computing (FTCS-22nd)*. Boston. July 1992. pp.76-84

[Assis Silva 99] F.M.Assis Silva. *A Transaction Model based on Mobile Agents*. PhD Thesis. Technical University Berlin. 1999

[Assis Silva, Krause 97] F.M.Assis Silva, S.Krause, A Distributed Transaction Model based on Mobile Agents. Proceedings of the First International Workshop, MA'97. Berlin, Germany. April, 1997. *Lecture Notes on Computer Science 1219*. K.Rothermel, R.Popescu-Zeletin (eds). Springer-Verlag. 1997. pp.198-209

[Badache, Hurfin, Macêdo 99] N.Badache, M.Hurfin, R.Macêdo. Solving the Consensus Problem in a Mobile Environment. *Proc. of the IEEE International Performance, Computing, and Communications Conference – IPCCC'99*. Phoenix/Scottsdale, USA. IEEE Press. 1999

[Birman 93] K. Birman. The Process Group Approach to Reliable Distributed Computing. Communications of the ACM, Vol. 9, No. 12. pp. 36-53, December 1993.

[Cai, Gloor, Nog 96] T.Cai, P.A.Gloor, S.Nog. DartFlow: A Workflow Management System on the Web using Transportable Agents. *Technical Report PCS-TR96-283*. Department of Computer Science. Dartmouth College. 1996

[Dalmeijer, Rietjens et al. 98] M.Dalmeijer, E.Rietjens, M.Soede, D.K.Hammer, A.T.M. Aerts. A Reliable Mobile Agents Architecture. in *Proceedings of the 1st IEEE International Symposium on Object-Oriented Real-time Distributed Computing (ISORC´98)*. 1998. pp.64-72

[Fuggetta, Picco, Vigna 98] A.Fuggetta, G.P.Picco, G.Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*. Vol.24, No.5. May, 1998. pp.342-361

[Johansen, Marzullo et al. 98] D.Johansen, K.Marzullo, F.B.Schneider, K.Jacobsen, D.Zagorodnov. NAP: Practical Fault-Tolerance for Itinerant Computations. *Technical Report TR98-1716*. Department of Computer Science, Cornell University. USA. November, 1998

[Johansen, van Renesse, Schneider 95] D.Johansen, R.van Renesse, F.B.Schneider. Operating System Support for Mobile Agents. *Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems*. 1995

[Magedanz, Popescu-Zeletin 96] T.Magedanz, R.Popescu-Zeletin. Towards "Intelligence on Demand" - On the Impacts of Intelligent Agents on IN. in *Proceedings of the 4th International Conference on Intelligence in Networks*. Bordeaux, France. November, 1996

[Murphy, Picco 99] A.L.Murphy, G.P. Picco. Reliable Communication for Highly Mobile Agents. In *Proceedings of the ASA/MA'99*. Palm Springs, CA, USA. October 1999

[Renesse, Birman, Maffeis 96] R. van Renesse, K.P.Birman, S.Maffeis. Horus, a flexible Group Communication System. *Communications of the ACM*. April 1996

[Rothermel, Straßer 98] K.Rothermel, M.Straßer. A Fault-Tolerant Protocol for Providing the Exactly-Once Property of Mobile Agents. *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS'98)*. West Lafayette, USA. October, 1998. pp. 100-108

[Schneider 97] F.B.Schneider. Towards Fault-tolerant and Secure Agentry. *Proceedings of the 11th International Workshop on Distributed Algorithms*. Saarbrücken, Germany. September, 1997

[Straßer, Rothermel, Maihöfer 98] M.Straßer, K.Rothermel, C.Maihöfer. Providing Reliable Agents for Electronic Commerce. in *Trends in Distributed Systems for Electronic Commerce - International IFIP/GI Working Conference TREC'98*. Springer. Berlin. 1998. pp.241-253

[Tanenbaum 95] A.S.Tanenbaum. *Distributed Operating Systems*. Prentice Hall, Inc. 1995

[Vogler, Kunkelmann, Moschgath 97] H.Vogler, T.Kunkelmann, M.-L.Moschgath. An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transactions. *Proceedings of the 1997 International Conference on Parallel and Distributed Systems (ICPADS'97)*. 1997. pp.268-274

[White 94] J.E.White. *Telescript Technology: The Foundation for the Electronic Marketplace*. General Magic. 1994

[Yen, Huang, Hwang 97] L.-H.Yen, T.-L.Huang, S.-Y.Hwang. A Protocol for Causally Ordered Message Delivery in Mobile Computing Systems. *Mobile Networks and Applications*. Vol.2. 1997. pp.365-372