MOBILE AGENTS AND ACTIVE NETWORKS

Fritz Hohl
Institute of Parallel and Distributed High-Performance Systems (IPVR), University of Stuttgart, Germany

Key words: Mobile agents, Mobile code, Active networks

1. INTRODUCTION

With the advent of mobile code, i.e. code that is sent to someone else's computer and executed there, a number of new technologies emerged that radically change some of the basic models of how networks and distributed applications work. Two of these new technologies are mobile agents and active networks. Mobile agents allow the programmer to employ "software robots" which wander around in the network, and fulfil tasks for a user. Active networks allow to add the ability to execute arbitrary code on routers and switches, thus gaining much more flexibility in the three lowest OSI layers.

In this tutorial, these two technologies will be presented and their relations exhibited. We start by defining the terms mobile code, mobile agent and active networks, and by disclosing the differences between them. Afterwards, we will examine the area of mobile agents and active networks in more detail.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35581-8_35

T. Yongchareon et al. (eds.), *Intelligence in Networks*
© IFIP International Federation for Information Processing 2000

2. MOBILE CODE, MOBILE AGENTS, AND ACTIVE NETWORKS

Mobile code allows to move code, i.e. procedures or even programs, to remote sites, and for executing the transferred code at these sites. Examples for mobile code systems are Java applets, remote evaluation (RPCs where the procedures are transferred before calling them) or even Postscript files (which are programs that are normally executed on printers).

Mobile Agents are program instances (or processes) capable of moving within the network under their own control. As a mobile agent contains code, mobile agents are special mobile code entities. The difference to mobile code lies in the inclusion of state (i.e. data state and execution state). This state is transported within the mobile agent, whereas mobile code entities are not able to transport state. However, the borders between both technologies are soft: if a mobile entity moves only once, it is hard to determine whether the entity is a mobile agent or just mobile code, especially if the entity is provided with some initial data. Even if the mobile entity travels along some nodes, we can think of this migration as a consecutive number of remote evaluations where the final state of one evaluation is (manually) taken as the parameters to the next evaluation. Apart from the technical characteristics, mobile agents also have another aspect: the usage as a programming paradigm. In this sense, mobile agents are a model where the programmer imagines to employ a breed of "software robots" wandering through the network, interacting with other agents and users. Note that the usage of this programming paradigm does not require a special technical realisation (just as object-oriented programming does not require object-oriented machine code), but an environment that allows the programmer to use this paradigm easily. Currently, the aspect of mobile agents as a paradigm is rarely addressed (see [BV99] and [GV97] for some exceptions).

An *active network* is a network where the transporting components (i.e. routers) are able to execute arbitrary code. This code is provided in some systems by special network packets (active packets) that can be injected by normal users. These active packets are of course mobile code entities. The difference to mobile agents lies on the one hand in the lack of transported state, on the other hand, active packets are executed in components that belong to the first three layers of the ISO OSI model, whereas mobile agents are layer seven entities (i.e. executed on end systems). However, the most radical view on active networks dissolves the OSI layering by arguing that active packets can be executed on every component of a network, be it a router, a switch, or an end system, thus allowing to dynamically place arbi-

trary functionality on any node. In this unified view, an active network is a more general model that builds the basis for mobile agents.

3. MOBILE AGENTS

Mobile agents are program instances that consist of three parts: code, data state, and execution state. They are able to migrate from one agent platform to another, taking all three parts with them. When it comes to migration, two types of agent systems can be distinguished: such that offer strong (or transparent) migration, and such that allow only for weak migration. Strong migration makes the migration process transparent to the agent programmer: the next statement after the migration command is executed on the target platform and the state of the program remains exactly the same as before, i.e. all three parts of the agent are transported automatically. Weak migration transports only code and data state of the agent, i.e. the execution state of the agent has to be en- and decoded manually by the agent programmer in case of a migration. Therefore, after a migration, often a start procedure is called.

In principle, mobile agents are a subtype of software agents, i.e. agents that have the ability to move. Practically the term software agent (or even agent) often denotes intelligent agents, i.e. agents that use AI techniques such as knowledge processing, although this usage is also an orthogonal attribute that may or may not apply to an agent.

Agent platforms play an important role for mobile agents, since agents can migrate only between them. These platforms protect the underlying computer system from attacks by the mobile agents on the one hand. On the other hand, agent platforms also execute the agents and provide functionality to them. This functionality includes:

communication

To communicate with other mobile agents, traditional mechanisms are offered by the systems, such as RPCs (or RMI), messages, tuple spaces etc. To address other agents, either a combination of a locally unique id plus the name of an agent platform are used (often for RPCs and messages) or communication elements (such as tuples) were specified. To cope with the problem of migrating communication partners, often either name services are used, or transparent proxy objects are employed that remain on a platform and relay communication requests to the target platform.

migration

If an agent requests a migration to another platform, the current platform stops the agent, encodes it into a transportable form, and sends it to the

target, often after having authenticated the target. More advanced version of migration ensure that the agent does not get lost even in case of node or communication failures during the migration.

Apart from these basic services, platforms sometimes offer other services such as persistency (i.e. storing data or agent snapshots), services lookup (i.e. finding agents that offer a service), locating agents (either locally or on a larger scale), integration of legacy systems such as databases, and the like. There is a larger number (currently over 60) of agent systems that have been designed by universities and industry (see [MAL] for an up-to-date and almost complete web listing of mobile agent systems). Most of them are research prototypes, only a few have been designed as a commercial product. Although nearly all of them use Java as the underlying basis, almost none of them is able to migrate mobile agents to another type of mobile agent system. There are standardization efforts (namely MASIF [MBB98] by the OMG and the FIPA proposals), but they do not specify migration interaction (in case of MASIF) or there were not implemented by agent systems (only 2 of them claim to implement MASIF, 1 implements FIPA).

Being the technical basis for applications, mobile agents offer a number of advantages. These include the *saving of network bandwidth* and *increasing of the overall performance* by allowing the application to process data on or near the source of data (e.g. a database), *asynchronous processing*, i.e. the possibility to fulfil a task without the need to have a permanent connection from the client to a host, *achieving true parallel computation* by employing a number of agents working on different nodes, *the replacement of a fault model* where network failures can interrupt in every phase of the computation by one where network failures can influence only the migration of an agent (as the rest is then done locally on the same node), and so on. Additionally, mobile agents "inherit" the advantages of mobile code systems, especially the possibility to transport functionality automatically to nodes where it has not been installed before.

Security is a very important aspect for mobile agent systems as both the system provider and the agent owner need to be sure that neither agents can attack the platform on a node nor that single platforms can attack agents, robbing e.g. transported digital money. Fortunately the one aspect, the protection of nodes against attacks by agents can be solved using techniques known from existing mobile code systems like Java applets. The other aspect, the protection of agents from attacks by the nodes, is not solved yet and constitutes a problem for the employment of mobile agent technology in open environments.

There are a number of application areas in which the usage of mobile agents is considered as being advantageous. These areas include among others: Information dissemination, mobile computing, information retrieval,

electronic commerce, software deployment and especially the management of telecommunication networks. Although mobile agents seems to offer many advantages and can be employed in some important application areas, currently, there is no broadly used real-world application that uses mobile agents.

The whole area of mobile agents was mainly inspired by the General Magic white papers [Whi94a],[Whi94b],[Whi95] where Jim White described Telescript, an early, but very mature commercial mobile agent system designed for electronic commerce applications. Another early paper that influenced mobile agent research very much is [HCK97]. If you are interested in the security aspects of mobile agents, [Vig98] can be recommended as a book that contains articles that represent some of the most important approaches in this area.

4. ACTIVE NETWORKS

Targeted at the lower layers of the OSI model (in contrast to mobile agents which work at the application layer), active networks try to overcome the shortcomings of traditional, passive networks: the difficulty of integrating new technologies into the existing network structure, poor performance due to redundant operations at several protocol layers, and the difficulty of accommodating new services in the existing architectural model. Additionally, applications have emerged recently that require computations within the network (e.g. firewalls, Web proxies, multicast routers). As no support for these application can be offered by the existing network infrastructure, computing power on regular, i.e. non-network nodes has to be allocated in order to fulfil these needs. In contrast to this, with active networks, the functions of a network component are no longer restricted by the built-in software of the vendors, but can be specified even dynamically by the applications that use the network.

Architectures for active networks can be distinguished according to the answer to the question of how activity is achieved. The first group of architectures, the "active packets" approaches, transport code in special packets, the second group, called the "active nodes" approaches place code in active nodes, allowing packets to carry code identifiers and parameters only. The third group uses flexible approaches that allows the user to choose between both approaches. Active packets approaches include the Smart Packets architecture (BBN Technologies) where a program has to be completely self-contained and has to fit into one unfragmented packet, thus allowing for only very compact programs, the Active IP Option (MIT) that allows to execute code fragments as an extension to the option field in the IP

protocol, and the M0 architecture (UCB, University of Zurich) which allows active packets to be very complex and powerful programs. Active nodes approaches include an architecture by the Georgia Institute of Technology where functions on active nodes are offered by network service providers, the DAN architecture (Washington University and ETH Zurich) that allows active nodes to load needed code from special code servers, and the ANTS architecture (MIT) where needed code is requested from neighbouring nodes. Approaches that combine both, active packets and active nodes include the SwitchWare project (University of Pennsylvania) which uses a layered architecture that allows the programmer to chose the trade-offs between the flexibility of the lower layer code execution and the speed and security of higher layers.

Security and safety are major concerns for active networks, as foreign programs were allowed to run on extremely security sensitive components. Breaches of security and even incorrect programs may result in a malfunctioning network, in privacy breaches, in attacks against other network parts to list just some possible threats. Some techniques that are used to achieve an adequate level of security are authentication, monitoring and control, limitation techniques, proof carrying code, as well as fault tolerance techniques and encryption. Authentication of active packets does not guarantee a program to be harmless, but allows to identify the author and the identity of a program, as well as the integrity of the code using public key techniques. Based on the identity of the program and other criteria, a reference monitor is then able to restrict the access of active packets to system resources and services. These restrictions may include also time and range limits (e.g. the number of hops). Proof carrying code allows programs to carry a proof with them that proves the correctness of the program given a specification.

There are a number of applications that profit from an architecture based on active networks. These applications include:

network management

Active networks can be used here to track and even to try to repair problems quickly without the need to maintain a communication connection from the management server to the managed components, which, in turn may be broken due to the problem it should handle. Further, the usage of active packets or nodes allows for tailoring management data and management policies.

multicasting

Using active networks for this application allows e.g. for dynamically moving multicast and repair packets caches to such routers that are "strategic" points, e.g. before wireless links etc., for suppressing negative

acknowledgements (NACKs) for originators that are known to be repaired in a short time, and for selectively sending repair packets to only those hosts which have requested it (which is a problem for multicast-like communication).

caching

With active networks, self-organizing caches are possible that determine dynamically where to place themselves depending on the current demand for data on different locations of the network.

See [Pso99] and [TSS98] for overviews of active network research.

5. LITERATURE

[BV99] Bryce, Ciaran; Vitek, Jan: The JavaSeal Mobile Agent Kernel, in: Proceedings of the First International Symposium on Agent Systems and Applications (ASA '99) /Third International Symposium on Mobile Agents (MA '99) featuring the Third Dartmouth Workshop on Transportable Agents (DWTA '99). To appear.

[CHK97] Chess, David; Harrison, Colin; Kershenbaum, Aaron: Mobile agents: Are they a good idea?. In Jan Vitek; Christian Tschudin (eds.): Mobile Object Systems: Towards the Programmable Internet, pages 25-45. Lecture Notes in Computer Science No. 1222. Springer-Verlag, April 1997. http://www.research.ibm.com/massive/mobag.ps

[GV97] Ghezzi, Carlo; Vigna, Giovanni; : Mobile Code Paradigms and Technologies: A Case Study, in: Rothermel; Popescu-Zeletin (eds.): Proceedings of the First International Workshop on Mobile Agents, MA'97, Springer-Verlag, 1997

[MAL] The Mobile Agents List. http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/ mal.html

[MBB98] Milojicic, D.; Breugst, M,; Busse, I.; Campbell, J.; Covaci, S.; et al: MASIF: The OMG Mobile Agent System Interoperability Facility, in: Rothermel; Hohl (eds.): Proceedings of the Second International Workshop on Mobile Agents, MA'98, Springer-Verlag, 1998

[Pso99] Psounis, Konstantinos: Active Networks: Applications, Security, Safety, and Architectures, in: IEEE Communications Surveys, Vol. 2, No. 1, 1999

[TSS98] Tennenhouse, David; Smith, Jonathan; Sincoskie, David; Wetherall, David; Minden, Gary: A Survey of Active Network Research, IEEE Commun. Mag., July 1998, vol. 36 no. 7, 1998

[Vig98] Giovanni Vigna (Ed.): Mobile Agents and Security. Springer-Verlag, Germany, 1998

[Whi94a] White, James: Telescript Technology: The Foundation for the Electronic Marketplace. General Magic White Paper, 1994

[Whi94b] White, James: Telescript Technology: Scenes from the Electronic Marketplace. General Magic White Paper, 1994

[Whi95] White, James: Telescript Technology: An Introduction ot the Language. General Magic White Paper, 1995