Distributed Transaction Processing as a Reliability Concept for Mobile Agents

Hartmut Vogler, Thomas Kunkelmann, Marie-Louise Moschgath
Darmstadt University of Technology
Information Technology Transfer Office
Alexanderstr. 10
D-64283 Darmstadt, Germany
{vogler, kunkel, malu}@ito.th-darmstadt.de

Abstract

Mobile agents offer a new possibility for the development of applications in distributed systems and are no longer a theoretical issue since different architectures for their implementations have been proposed. With the increasing market of electronic commerce it becomes an interesting aspect to use autonomous mobile agents for electronic business transactions. Being involved in money transactions, supplementary security features for mobile agent systems have to be ensured.

In this paper we present an architecture for a mobile agent system which offers fault tolerance for the whole agent system at a high level. This architecture additionally guarantees security for the host as well as security for the agent. To handle these issues for mobile agents we use various encryption mechanisms and we apply a novel method for mobile agent systems by using distributed transactions in our architecture. Due to this security architecture an agent will be enabled to carry out money transactions.

Keywords: mobile agents, fault tolerance, reliability, security, electronic commerce, distributed transactions

1 Introduction

The paradigm of mobile agents [HCK95] became one of the most interesting topics in today's research work of computer science. Since several research institutes and companies developed high-quality prototype systems for mobile agents, e.g. *ARA* [PeSt97], *Mole* [SBH96] or *Telescript* [Whi94], mobile agents are no longer a theoretical issue. In the areas of information retrieval, network management, workflow, mobile computing [CGH+95] or telecommunication [MRK96] a lot of interesting applications for mobile agents emerge.

Unfortunately, mobile agents involve some significant security problems for the agent as well as for the host [FGS96]. There exist some approaches to cover these problems like the usage of "safe"-languages like *Safe-TCL* [BoRo93] or *Safe-Python* [Pyth94], *Java* [ArGo96]. For those languages certain commands are considered as unsafe and hence are not made available to untrusted programs, e.g. file system access. For the agent security and authentication, the most suitable mechanism is the usage of *digital signatures* [Riv92], which guarantee the integrity for those parts of the agent which remain static. However, these methods don't guarantee sufficient security, only static parts of the agents can be signed. On the other hand they restrict the capabilities of an agent, e.g. the available commands for the agent are not sufficient for many scenarios. For new and useful applications, mobile agents must also provide features for electronic commerce [KaWh96].

Another important topic in the paradigm of mobile agents is the fault tolerance and the reliable agent transfer. These security issues will be discussed in Section 2. Based on this analysis we explain our architecture of a secure mobile agent system and discuss its benefits in Section 3. After describing our realization using TIP in Section 4 we conclude the paper with a summary and an outlook on our future work.

2 Reliable Agent Transfer

Fault tolerance for mobile agent systems is an unsolved topic, to which more importance should be attached. Besides the security problems by intended attacks it is very important to realize that an agent can simply get lost by errors of the network or the hosts. If an agent itinerates autonomously in the network, there is no instance which can guarantee that the agent reaches the next host correctly and won't get lost. For example, when using SMTP (E-mail) as a transport medium there are no mechanisms for the reliability of the transport or any recovery handling in case of errors. There are even no standardized mechanisms to detect this fault situation.

Besides the network errors, fault situations also can be originated by problems or a breakdown of the host. In case of such a failure, there have to be suitable recovery mechanisms for the agent.

Based on network or host errors an agent can also be duplicated. Such an inconsistent state of the agent system can cause significant problems and must be avoided.

2.1 Classes of fault semantics

The problem of fault tolerance for agents is an application level topic. After a reliable transfer an agent has to be initiated at the new host. This offer various fault situations, which can easily cause an inconsistent state for the whole agent system.

The transfer of an agent can be compared with a *remote procedure call* (RPC), which guarantees the *exactly-once-semantic* (only-once-type-2-semantic) [MüSc92].

In the context of an agent transfer the other classes of fault semantic aren't sufficient. An agent transfer with the *maybe-semantic* would guarantee that the agent is transferred only once, but we don't have the guarantee that the agent is received and initiated correctly at the new host. In the class of the *at-least-once-semantic* we have the guarantee that the agent is received and initiated correctly by the new host. However we can not avoid that the agent may be duplicated if any communication errors occur and the agent has to be resent. The *at-most-once-semantic* (only-once-type-1) guarantees the atomicity of the agent transfer. In this class the agent will either be transferred correctly or, in case of a fault, an error message will be sent to the sending host, and no further action takes place at the receiving host. Even this fault semantic isn't sufficient for an agent transfer due to a possible host crash. Only the *exactly-once-semantic* guarantees a consistent recovery after a host crash, so an agent will always be transferred exactly once and will not get lost or duplicated. This can only be achieved with persistent storage and a protocol for distributed transactions.

After this analysis we see that only *distributed transaction processing* (DTP) offers the possibility to control the state of a mobile agent system and also offers the scalability of the system to involve further parties.

2.2 Introduction in DTP

The usage of *transactions* [BeNe96] is a very popular concept for the management of large data collections. Transactions guarantee the consistency of data records when multiple users or processes perform concurrent operations on them. In general, the properties of transactions are known as the *ACID* properties (**A**tomicity, an indivisible set of operations; **C**onsistency is guaranteed for the data; **I**solation of parallel data access in different transactions; **D**urability, the events are stored permanently or completely rejected) [GrRe93].

The access of distributed resources, e.g. databases on different computers, within a transaction is called a distributed transaction. For committing the result, the peers involved in a distributed transaction usually communicate via the *two-phase-commit* protocol (2PC).

Nowadays the concept of transaction processing is used - apart from the "classical" database applications - in various applications, e.g. for the management of large distributed systems [Vog96] [MaHa96]. There also exist first experiences and implementations using transaction processing on the Internet [DEC96a] [TIP97], which offer a wide range of opportunities to build new secure and fault tolerant applications. Furthermore it is important to recognize that DTP systems are not monolithic, rather they are open by standardized protocols.

3 Architecture of a secure agent system

In the previous section we outlined the importance of transactions for a reliable agent transfer. However, the usage of distributed transactions and the 2PC protocol contain some significant drawbacks like the bad performance of the 2PC protocol due to its blocking characteristics. DTP also introduces implementation overhead and additional software components. Analyzing these pros and cons of DTP in the context of mobile agents we came to the conclusion that the benefits of distributed transactions should be used for additional valuable features of an agent system. Therefore we build a DTP-based enhancement of existing systems for host security as well as agent security, which additionally allows agent management and control. With our system we also equip an agent for electronic commerce.

3.1 Trust Between Host and Agent

Analyzing the situation in the mobile agent paradigm we came to the conclusion that it is not possible to achieve security with complete functionality for the agent without trusting each other. By aid of a third party, which has information about all instances of the closed system, a kind of trusted situation or contract can be achieved. In our architecture we call this instance *trust service*. This trust service is the core of our architecture for agent security and fault tolerance. On the one hand, our concept of a trust service is based on the concepts of *Kerberos* [NeTs94] in the way that we also use a trusted third party for a session key generation and distribution. On the other hand it is important to see the difference between our trust service and well-known security services like the security service of the *Distributed Computing Environment* (DCE) [Hu95]. Our trust service is an extension to handle the

specific problems of mobile agents and is based on common security concepts and services.

Besides a special security protocol, which is explained in the following section, our trust service logs data about the agents and the hosts. These data include a record with the agent's identity, the host ID and the time interval of the visit. The trust service and the protocol additionally guarantee the correctness and consistency of these data. With these data we have enough information about the agent and the host so that in case of a breach of trust we can ascertain the originator and call him to account. Our system also offers the basis for agent control and management.

3.2 Life-Cycle of an agent

The basis of our secure protocol for the agent migration is a combination of data logging, encryption mechanisms and DTP. To use PGP [Gar94] in our system we assume that every participant in the agent system possesses a certified public key [Kal93]. We also assume that every instance, hosts as well as users of the agent system, are registered at the trust service.

In conjunction with the registration a user announces to the bank that he will participate in the electronic cash system and transmits via a secure channel his bank account number and his credit card number. The electronic cash system establishes a virtual account and notifies the user about the virtual account number (VAN). All purchases with this virtual account number have to be confirmed by the owner before the credit card will be charged. This concept introduced by First Virtual [FV96], with freely accessible data for the payment, seems to be the suitable mechanism for mobile agents.

Other electronic cash systems are based either on a secret (credit card number) like the *iKP* (Internet Keyed Payment Protocol) of IBM [IBM96] or on electronic proxies for money (electronic coins or coupons) like *DigiCash* [Dig96] or *Millicent* [DEC96b]. In either of these cases the agent has to carry out a secret information. So these solutions are not suitable for mobile agents.

The following items describe the life-cycle of an agent in our architecture.

Registration

The originator announces a new agent to the trust service. The trust service generates a unique ID for the agent, which is used in conjunction with the registered user and which is sent back. This communication is protected by public key encryption. With the ID for the agent and the VAN, the agent is equipped to carry out money transactions.

When the agent is initiated at the host of its originator, it looks for a new target host by aid of special traders for agents like it is suggested in [SKL97]. To find a suitable new host a trader must have additional information concerning the resources, environment conditions and services at the offered host.

Agent transfer

When the contract between the target host and the agent is negotiated, a copy of the agent will be transferred to the new host. To achieve security we use two different mechanisms. Distributed transaction processing guarantees a consistent state of the whole system during transport. Encryption guarantees protection against modification during the transfer of the agent in an unreliable network.

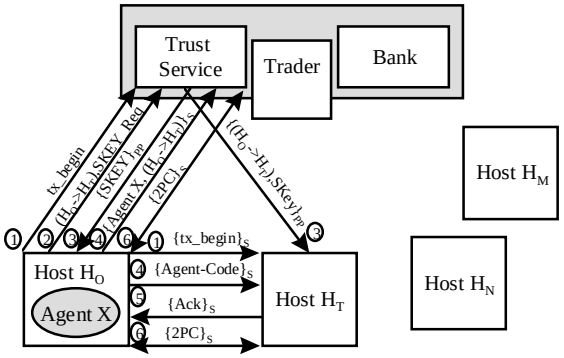

Figure 1: Protocol steps of an agent transfer

Figure 1 illustrates the different protocol steps of an agent transfer:

1. The originator host (H_O) demarcates the begin of a distributed transaction (*tx_begin*), in which the trust service and the target host (H_T) will be involved. This implies that all action and commands will be executed in a transactional context.
2. The originator host requests a *session key* (S) [Sch96] for the secure transfer of the agent to the target host.
3. The trust service generates a session key and propagates it to the originator and the target host by using public key protocol (PP).
4. The originator host transfers a copy of the agent, encrypted with the session key, to the target host. The open design of our architecture allows to use various mechanisms or protocols for the agent transfer. So we can use E-mail [Bor94], HTTP like it is used in [LDD95] or the suggested *Agent Transfer Protocol* [Lan96].
5. After decrypting, the target host initializes the copy of the agent and acknowledges the receipt.
6. The originator host initiates the 2PC protocol to conclude the transaction. A successful conclusion of the transaction implies that the results, i.e. deletion of the old agent, start of the new agent and update of data at the trust service, are made permanently visible. With

the end of the transaction the session key is invalidated.

Money transactions

Being transferred successfully the agent can be involved in a business transaction either as a buyer or as a seller. After an agreement is arranged, the buyer gives his virtual account number to the seller, who contacts the bank and transmits the virtual account number together with information concerning the business, the identifiers of both parties and the host. The bank checks the consistency of the information, acknowledges the validity of the business transaction and transfers the money to the seller's account.

Termination of an agent

There exist various methods to eliminate a mobile agent in our system. If an agent finished its work it will either return to its owner or terminate at the current host. In both cases an appropriate message has to be sent to the trust service, which logs this information. From this moment the agent becomes invalid.

An agent can also be eliminated on behalf of its owner or the trust service. Therefore the trust service sends a message to the current host, which deletes the agent. Similar to the first case the agent is now invalid and can not perform any further actions. Also the virtual account number in conjunction with the agent ID is invalid.

3.3 Benefits for mobile agent systems

Using distributed transaction processing as a reliability concept brings various benefits into mobile agent systems.

Fault tolerance and reliable agent transfer

The most significant benefit of our architecture is the reliability of the whole agent system. With the usage of distributed transaction processing we achieve a reliable agent transfer and guarantee the *exactly-once-semantic*. This new technique for the agent migration avoids the duplication of an agent as well as its loss. In case of error situations and host crashes the recovery and rollback mechanisms of the DTP system cater for a reliable and consistent resumption of the agent transport. During the critical phase of the transport of an agent the 2PC protocol guarantees the preservation of consistent states at the hosts and the trust service.

The usage of distributed transaction processing in our protocol offers high-level fault tolerance for a mobile agent system. With additional checkpoints (with a copy of the agent), the agent can be recovered or rolled back to this point. Checkpoints are useful e. g. if a host cannot recover an agent after a crash occurs. A rollback mechanism for the agent might be useful in case of cycle detection and a possible reinitialization of the agent to the state before the cycle occurred.

Security for the host

With our architecture we guarantee that an agent is only introduced in the system by a trusted user and can only itinerate on a chain of trusted hosts. We log all the relevant data about the visiting agent, like information about the owner and the life cycle of the agent. In case of a breach of trust by the agent the originator of the agent can be ascertained and called to account for the actions of his agent. With our architecture we achieve an indirect security for the host without any restriction to the capabilities of the agent in contrast to the concept of "safe" languages. So applications which also need critical commands (e.g. write operations) like remote software update or network management can be performed by mobile agents.

Security for the agent

Our guarantee for the security of the agent include various aspects. We guarantee that besides the scope of trusted hosts no other instance has access to the agent due to the encryption methods used. So no untrusted third party can copy, modify, destroy or rob the agent. If one of the trusted host attacks the agent, the logging facilities of our system can trace back this breach of trust and similar to the host security case, call the responsible person to account.

In comparison with other concepts for securing an agent, like the usage of a digital signature for a static agent or the static parts of the agent, our system achieves an indirect security for the agent without any restrictions to its capabilities.

The trust service offers the possibility for the agent to encrypt its collected data with the public key of the trust service and to compare these data at a safe place inside the trust service. This method avoids a modification of already collected information and a manipulation of the agent by a host. So a fair competition of concurrent hosts is guaranteed and the agent can e.g. buy at the host with the most reasonable offer.

Electronic commerce enhancement for mobile agents

Our architecture offers an agent to cope with money transactions based on the concept of First Virtual. Additionally, the bank service can check the consistency of a money transaction by aid of the logging information. It can be validated that an agent involved in the business actually resides at the host from where the money transaction was announced to the bank. In case of an error or cheating of one party this can be detected an the money transaction will be refused. It is then up to the seller and the trust service to initiate further action.

Agent control and management

With the information about the agent and its itinerary at the trust service, suitable control and management

functions can be implemented. The owner of an agent can always locate it and give new instructions to it. Also a control of its lifetime and its goals can be achieved. The trust service can additionally offer to store a copy of the agent in the case that a host does not have a persistent storage or recovery mechanisms.

4 Realization with the Transaction Internet Protocol

The realization of our protocol is based on the recently published *Transaction Internet Protocol* (TIP) by Microsoft and Tandem [LEK97]. The *Internet Engineering Task Force* (IETF) currently examines to standardize TIP for the Internet. There exists also a publicly accessible reference implementation of TIP in JAVA [TIP97].

TIP includes a one and a two phase commit protocol and is based on TCP/IP. For every node involved in the transaction a TCP/IP connection will be established. Because TIP uses the end-to-end communication mechanisms of TCP/IP like time-outs, the implementation is rather simple and easy to extend for the special use for agent communication and migration. The main objective of TIP lays in the definition of state machines for all nodes involved in a transaction to achieve a common state.

TIP distinguishes two models, pushing and pulling a transaction, for propagating a transaction between two nodes. For the current realization we use the pushing mechanism, where first the transaction managers propagate a transaction identifier before the resource managers will be involved. At the hosts no multi-user access to a single resource takes place, so the realization of a transaction manager and a resource manager is quite simple. Only the corresponding interfaces must be realized. At the trust service and the bank service the usage of commercial transaction and resource managers might be adequate.

Messages in TIP are defined as plain ASCII strings. The design of our system demands that all communication, including the transaction coordination messages, must be encrypted. So we use an enhanced version of TIP, where the encryption mechanisms are layered between TIP and the TCP/IP stack. This modification is necessary because in TIP no security mechanisms are supported.

For all commands in TIP it is possible to include comments, which will normally be ignored by the state machine. We use these comments to transmit additional information, e.g. in the BEGIN command for the trust service we include the information about the agent ID and the location of the new host.

5 Conclusion and Future Work

In this paper we presented a concept how to use distributed transaction in a mobile agent system to achieve reliability for the agent transfer. Based on this concept we developed a security architecture for a mobile agent system which is highly secure against external attacks. The integrity and internal security is guaranteed by a trust service and by logging all relevant information. So in case of a fraud the offender can be ascertained and called to account. Additionally, we offer the agent to carry out money transactions based on an enhancement of the payment concept of First Virtual. Our architecture can be used on top of existing mobile agent systems, e.g. as an enhancement of the “GO-Statement“, and offers a high reliability because the implementation is based on standardized components.

We believe that every mobile agent-based application needs a special degree of security and fault tolerance. So we see as future work the integration of additional security mechanisms to incorporate application-specific security models into our architecture.

One direction will be the integration of the security models and policies of different agent languages like *Java* and *TCL* [OLW96]. Thus we can build various classes of security by the language and can assign them to the specific agent and its application. Another direction of enhancing our system is the integration of a *personal security assistance* [RaJa96], which observes an agent and permits critical commands only after a check by an expert. This decision is based on the degree of trust to the agent and the already executed commands.

So as the objective for the future we will extend our system to a toolbox with various features to build the right degree of security, fault tolerance and electronic commerce capabilities, specific for different agent-based applications.

We also investigate other distributed transaction processing systems like the CORBA-based *Object Transaction Service* (OTS) [OMG94] in a related project. For the mobile agent project we will examine the replacement of TIP with OTS and a complete integration of our system into CORBA.

Literature

[ArGo96] K. Arnold, J. Gosling: *The Java Programming Language*, Addison-Wesley, ISBN 0201-63455-4, 1996

[CGH+95] D. Chess, B. Grosof, C. Harrison, D. Levin, C. Parris, G. Tsudik: *Itinerant Agents for Mobile Computing*, IBM Research Report #RC 20010, IBM Research Division, 1995

[BeNe96] P. Bernstein, E. Newcomer: *Principles of Transaction Processing*, ISBN 1-55860-415-4, Morgan Kaufmann Publisher, 1996

[BoRo93] N. S. Borenstein, M. T. Rose: *MIME Extensions for Mail-Enabled Applications: application/Safe-Tcl and multipart/enabled-mail*, Distributed as part of the Safe-Tcl 1.2, November 1993, Working Draft

[Bor94] N. S. Borenstein: *EMail With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail*, ULPAA '94, Barcelona

[DEC96a] Digital Equipment Corporation*: Enterprise TP on the Internet*, http://www.software.digital.com/tpi/TPI_CO.HTMl, 1996

[DEC96b] Digital Equipment Corporation: *MILLICENT Digital's Microcommerce System* http://www.research.digital.com/SRC/millicent/, 1996

[Dig96] DigiCash Home Page: http://www.digicash.com/

[FGS96] W. Farmer, J. Guttman, V. Swarup: *Security for mobile agents: Issues and requirements*, In Proc. of the 19th National Information Systems Security Conference, Baltimore, MD, 1996

[FV96] First Virtual Home page: http://www.fv.com

[Gar94] S. Garfinkel: *PGP: Pretty Good Privacy*. ISBN 1-56592-098-8, O'Reilly & Associates, 1994

[GrRe93] J. Gray, A. Reuter: *Transaction Processing: Concepts and Techniques*, ISBN 1-55860-190-2, Morgan Kaufmann Publisher, 1993

[HCK95] C.D. Harrison, D.M. Chess, A. Kershenbaum: *Mobile Agents: Are they a good idea?*; IBM Research Report #RC 19887, IBM Research Division, 1995

[Hu95] W. Hu: *DCE Security Programming*, ISBN 1-56592-134-8, O'Reilly & Associates, 1995

[IBM96] IBM Research Hawthorne and Zürich: *Internet Keyed Payment Protocols*, http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/iKP.html, 1996

[Kal93] B. Kaliski: *Privacy Enhancement for Internet Mail: Part IV: Key Certification and Related Services*, RFC 1424, RSA, February 1993

[KaWh96] R. Kalakota, A. B. Whinston: *Frontiers of Electronic Commerce*, ISBN 0-201-84520-2, Addison-Wesley Publishing Company, Inc., 1996

[Lan96] D. B. Lange: *Agent Transfer Protocol* ATP/0.1 Draft, http://www.ibm.co.jp/trl/aglets/atp/atp.html, July 1996

[LDD95] A. Lingnau, O. Drobnik, P. Dömel: *An HTTP-based Infrastructure for Mobile Agents*, World Wide Web Journal - 4th Int. World Wide Web Conference Proceedings, Boston, December 1995

[LEK97] J. Lyon, K. Evans, J. Klein: *Transaction Internet Protocol*, Internet-Draft, http://204.203.124.10/pdc/docs/TIP.txt

[MaHa96] P. Mandl, B. Hackler: *Transaktionsorientierte Managementsysteme und Managementprotokolle*, Praxis der Informationsverarbeitung und Kommunikation, Vol. 19, No. 4, 1996

[MüSc92] M. Mühlhäuser, A. Schill*: Software Engineering für verteilte Anwendungen*, ISBN 3-540-55412-2, Springer Verlag, 1992

[MRK96] T. Magedanz, K. Rothermel, S. Krause: *Intelligent Agents: An Emerging technology for next generation telecommunication*, Proc. of INFOCOM'96, 1996

[NeTs94] B. C. Neuman and T. Ts'o, Kerberos: *An Authentication Service for Computer Networks*, IEEE Communications Magazine, Volume 32, Number 9, 1994

[OLW96] J. K. Ousterhout, J. Y. Levy, B. B. Welch*: The Safe-Tcl Security Model*, http://www.sunlabs.com/research/tcl/SafeTcl.ps, Draft, November 1996

[OMG94] Object Management Group: *Object Transaction Service*, 1994

[PeSt97] H. Peine and T. Stolpmann: *The Architecture of the Ara Platform for Mobile Agents*, Proc. of the First Int'l. Workshop on Mobile Agents MA'97 Berlin, LNCS No. 1219, Springer Verlag, 1997

[Pyth94] Safe-Python Homepage: http://minsky.med.virginia.edu/sdm7g/Projects/Python/SafePython.html, 1994

[RaJa96] A. Rasmusson, S. Janson. *Personal security assistance for secure Internet commerce*. In New Security Paradigms '96, ACM Press, Sept. 1996

[Riv92] R. Rivest: *The MD5 Message-Digest Algorithm*, RFC 1321, MIT, April 1992

[SBH96] M. Straßer, J. Baumann, F. Hohl: *Mole - A Java based Mobile Agent System*, Proc. of the ECOOP '96 Workshop on Mobile Object Systems, 1996

[Sch96] B. Schneier: *Applied Cryptography*, ISBN 0-471-11709-9, J. Wiley & Sons, Inc., 1996

[TIP97] Transaction Internet Protocol, Reference Implementation, http://204.203.124.10/pdc/docs/TIP.zip

[Vog96] F. Vogt: *Werkzeuge für die Transaktionsverarbeitung heute - morgen*; Proc. of the 19th European Congress Fair of Technical Communication - ONLINE'96, Congress VI, Hamburg, Feb. 1996

[Whi94] J. E. White: Telescript Technology: *The foundation for the electronic Marketplace*, White Paper. General Magic Inc., 1994