Enhancing Mobile Agents with Electronic Commerce Capabilities

Hartmut Vogler, Marie-Luise Moschgath, Thomas Kunkelmann

Darmstadt University of Technology
Information Technology Transfer Office
Wilhelminenstr. 7
D-64283 Darmstadt, Germany
{vogler, malu, kunkel}@ito.tu-darmstadt.de

Abstract. *The paradigm of mobile agents offers a powerful and flexible opportunity to develop distributed applications on a high-level of abstraction. One of the most interesting tasks for mobile agents is an active search for relevant information in non-local domains on behalf of their users. A mobile agent will be sent out on behalf of an user to various host servers in the Internet and to find information. In the future this information might not be freely accessible, so the agent may have to pay for them. Thus the mobile agent has to be equipped with electronic commerce capabilities. This implies a fault tolerant and secure infrastructure for the mobile agent. In this paper we present a system which offers electronic commerce capabilities for mobile agents. These capabilities a part of an architecture which guarantees different security issues and fault tolerance for mobile agents. Security for the partners involved is handled by encryption methods based on a public key authentication mechanism and by secret key encryption of the communication. To achieve fault tolerance for the agent system, especially for the agent transfer to a new host, we use Distributed Transaction Processing (DTP). This 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.*

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

1 Introduction

Mobile agents [HCK95] offer a new possibility for the development of applications in distributed systems. The paradigm of mobile agents is based on various sections of computer science. The term *agent* was originally coined by the artificial intelligence community, where agents are defined as autonomous, reactive and pro-active entities which have a social ability [WoJe95]. From the operating system point of view, mobile agents are an evolution of codeshipping and object and process migration. In the research field of networking and distributed systems, mobile agents are an extension of client/server-computing [ZLU95] and of the World Wide Web capabilities using *Java* [ArGo96] as well as an emerging technology for the next generation of telecommunications [MRK96]. Encompassing these three areas of computer science, the paradigm of mobile agents has become one of the most interesting topics of the today’s research work.

Mobile agents are no longer a theoretical issue as different architectures for their implementations have been proposed. Research institutes as well as companies develop high-quality prototype systems for mobile agents, e.g. *ARA* [PeSt97], *Mole* [SBH96] or *Telescript* [Whi94]. Yet, these agent systems typically do not satisfy all requirements that mobile agents need for a full environment. However, these systems are open for further extensions to include additional characteristics.

One of the most interesting tasks for mobile agents is an active search for relevant information in non-local domains on behalf of their users. This includes retrieving, analyzing, manipulating and integrating information available form different sources. For example, a mobile agent will be sent out on behalf of an user to various servers to retrieve the latest version of a technical paper concerning “Electronic Commerce“. On its itinerary, a mobile agent can interact on an ask/receive paradigm with a static agent residing on the host that offers the information. Thus, the mobile agent can find out only the relevant information or data and avoid unnecessary network traffic. In this scenario of information gathering in the Internet, commercial aspects [KaWh96] has become more and more relevant, e.g. the mobile agent has to pay for information, or in other possible application scenarios will be paid for a service it offers. As a consequence, mobile agents have to be equipped with electronic commerce capabilities to be an autonomous entity in the electronic market place. This unfortunately demands a high level of security and fault tolerance for the mobile agent either is not robbed, or does get lost or duplicated on its itinerary.

In this paper we describe an enhancement of mobile agent systems for electronic commerce based on the concept of a bank service for the Internet introduced by *First Virtual* [FV96] with special extensions for mobile agents, which offer additional control within the payment. This enhancement is part of an architecture for mobile agent systems that offers fault tolerance as well as security and can be used on top of existing systems.

In section 2 we describe our extension for mobile agent systems with the embedded electronic commerce capabilities for the agent. The core of our system builds an instance called *trust service* and a special protocol for agent migration, which is a combination of data logging, encryption mechanisms and distributed transactions. In section 3 we discuss various realization and implementation aspects. We conclude our paper in section 4 with a summary and an outlook on our future work.

2 A Mobile Agent System with Electronic Commerce Capabilities

In a scenario where a mobile agent is equipped with electronic commerce capabilities it necessary to guarantee that an agent does not get lost or duplicated during its itinerary. Thus, the main objective for the design of our architecture was to achieve a reliable, fault tolerant and secure agent transfer which guarantees the *exactly-once-semantic* (only-once-type-2-semantic) [MüSc92].

The problem of a reliable and fault tolerant agent transfer is an application level topic. We can guarantee a reliable transfer on the transport protocol level by using TCP/IP, but after this transfer, an agent has to be initiated at the new host. This offers various fault situations, which can easily cause an inconsistent state for the whole agent system by the loss or duplication of an agent. Thus, other classes of fault semantics (may-be, at-most-once, at-least-once) are not acceptable for mobile agents. In our system we guarantee the exactly-once-semantic by ensuring the agent migration with distributed transactions [BeNe96] and the 2-phase-committ (2PC) protocol.

However, the usage of distributed transactions and the 2PC protocol contains some drawbacks such as the bad performance of the 2PC protocol due to its blocking characteristics. *Distributed transaction processing (DTP)* also introduces implementation overhead and additional software components. In analyzing these pros and cons of DTP in the context of mobile agents, we came to the conclusion that the benefit of distributed transactions should be used for additional valuable features of an agent system. Therefore we built a DTP-based enhancement of existing systems for host security as well as agent security, which additionally allows agent management and control. From the application point of view, the most interesting enhancements of our systems are the electronic commerce capabilities for mobile agents.

2.1 Trust Service

In analyzing the security problems in the mobile agent paradigm [FGS96], we came to the conclusion that it is not possible to achieve full security with complete functionality for the agent without trust between the agent and the host. By aid of a third party, which has information about all instances of the agent 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 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 extends these concepts to handle the specific problems of mobile agents and is based on common security concepts and services.

The trust service in our architecture is responsible for a special security protocol, which is explained in section 2.3, and for the logging of data about the agents and the host. 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.

We assume that every instance of the system, hosts as well as users of the agent system, are registered at the trust service. This registration must be associated with a legally binding contract in order to achieve a trusted starting point. To use PGP [Gar94] in our system we also assume that every participant in the agent system possesses a certified public key [Kal93].

The concept of a trusted host is an abstraction for an entire machine or only an agent environment on a host. Therefore, it is not possible to use only a host based authentication mechanism like the *Secure Socket Layer* (SSL) [FKK96].

2.2 Virtual Bank Service (VBS)

In conjunction with registration, a user of the agent system announces to the Virtual Bank Service (VBS) that he will participate with his agents in the electronic cash system as a buyer and/or as a seller. Therefore, the user needs a private e-mail account and a regular bank account. In Figure 1 we see the registration of a user at the VBS.

The user sends his application including his full name and his PIN choice, encrypted with the public key P_V of VBS via e-mail to the bank. The VBS confirms the application by returning a reference number, which is used in the next step for authentication. The reference number is encrypted with the public key P_U of the buyer and the private key S_V of the VBS. Now the user sends his sensitive financial data (credit card number or bank account), encrypted with his private key S_u and the public key P_V to VBS.

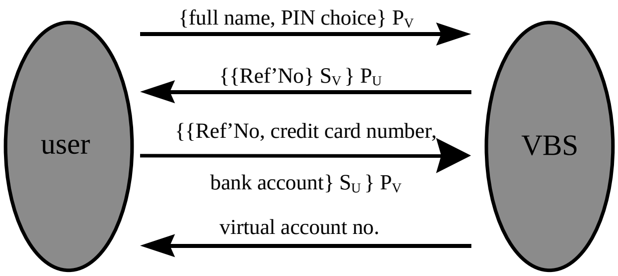
{full name, PIN choice} P_V
{{Ref'No} S_V } P_U
user
{{Ref'No, credit card number,
bank account} S_U } P_V
VBS
virtual account no.

Fig. 1. Registration at the VBS

The VBS establishes a virtual account and notifies the user about the virtual account number (VAN), generated from the user's PIN. All sales effected with this virtual account number have to be confirmed by the owner before the credit card will be charged. The VAN can be used for all agents a user announces to the system. Furthermore, the VAN can be used only in conjunction with a valid agent ID in our payment protocol, which we explain later.

This concept, introduced by First Virtual [FV96], with freely accessible data for the payment, seems to be the suitable mechanism for mobile agents because this concept needs no authentication of the participants and supposes no secure transmission of information. This concept abstains from encryption mechanisms and avoids technical hurdles (e.g. export restrictions of the USA for encryption mechanisms) and additional costs.

Other popular electronic cash systems are based either on secret information (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* [DEC96]. In either case the agent has to keep secret information which should not be visible to a third party. Even if an agent migrates on a chain of trusted hosts, it is against the intention of those electronic cash systems to make the payment data freely accessible to other instances than the transaction partners. So, only solutions with freely accessible data are suitable for mobile agents.

2.3 Life-Cycle of an agent

The following items describe the life-cycle of an agent in our architecture and its integration in an electronic market place.

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 agent-ID and the VAN of the agent owner, the agent is equipped to perform 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 as suggested in [PKL97]. To find a suitable new host the trader must possess agent-specific information concerning the resources, environment conditions and services offered at the host.

Agent transfer

After a successful negotiation between the target host and the agent, 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 agent system during transport. With encryption we achieve protection against modification during the transfer and authentication of the agent in an unreliable network.

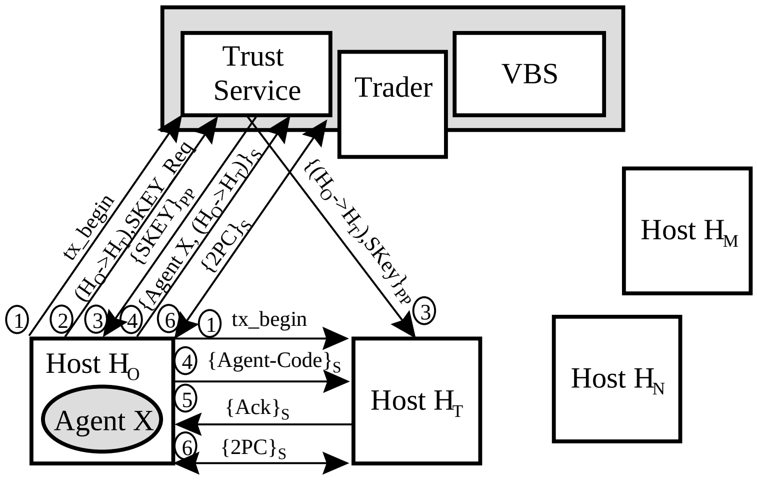

Fig. 2. Protocol steps of an agent transfer

Figure 2 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 following action and commands will be executed in a transactional context.
2. The originator host requests a *session key* (S) [Sch96] for secure communication and a 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 the use of various mechanisms or protocols for the agent transfer.
5. After decrypting the agent, 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

Once it is successfully transferred, the agent can process its tasks on the residing host, e.g. information retrieval on the local database. Therefore it can cooperate with a static agent on the host. In the future, the mobile agent may have to pay for information or, in other possible application scenarios, it will be paid for a service it offers. Figure 3 illustrates our protocol for the money transaction between a buyer agent and a seller agent.

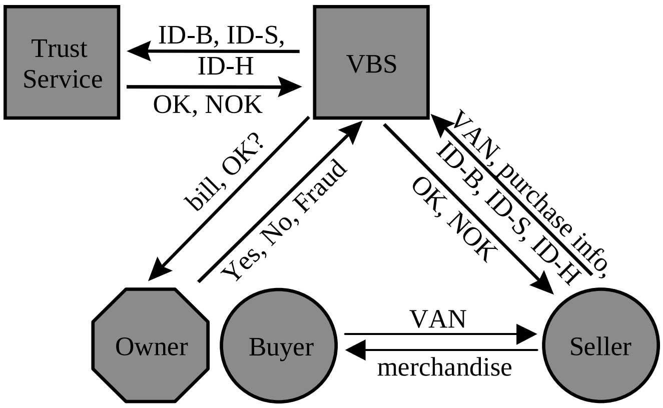

Fig. 3. Protocol for the money transaction

After a successful agreement is arranged between the local agent and the mobile agent, the buyer gives its virtual account number to the seller, which contacts the VBS to confirm the payment. Therefore the seller transmits the virtual account number together with information concerning the business, the identifiers of both parties (ID-B: identification of the buyer; ID-S: identification of the seller) and the host (ID-H: identification of the host) to the VBS. The VBS verifies by the aid of the trust service the consistency of the information (e.g. the mobile resides actually resides on this host), and acknowledges the validity to the seller. If the seller chooses the first payment sequence, he hands over the ordered article or information to the buyer after receiving an acknowledge. Later on, the VBS sends a bill (including the description and the price of the item, information about the seller, and the request for confirmation) to the owner of the agent. The owner now has the opportunity to confirm or to reject the sale or to shout ‘fraud’. Only in the first case does the seller receive the payment. If the seller selects the second payment sequence, it has to wait for an acknowledgment. Therefore the VBS sends forthwith a request to the owner of the agent, waits for its answer and informs the seller. Only if the seller receives ‘OK’ does it hand over the article. The first payment sequence is designed for sales with low amounts, the second for sales with high amounts.

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

This protocol for money transactions can be modified or enhanced for the special requirements of an application scenario. Therefore we propose a modification to keep the costs for a single transaction in an adequate range in a way that the bank does not check all money transactions for consistency. Instead, the bank only checks transactions with a high amount and take random samples for small business transactions.

To achieve a better control of the agent, the system can be enhanced with limits concerning the agent’s money transactions. With our architecture, the owner is able to define a maximal amount of money the agent can spend in one transaction or during the whole itinerary. These limits are stored and managed at the trust service.

Termination of the agent

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

An agent can also be eliminated anytime 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 cannot perform any further actions. Also the virtual account number in conjunction with the agent ID is invalidated.

With our system we can eliminate a leakage in existing agent systems which have to use special protocols for orphan detection and termination [Bau97]. Furthermore, our architecture offers the facility to define deadlines for mobile agents and to control them by the trust service.

3 Realization and Implementation Aspects

One of our design goals is to demonstrate the platform independence of our concepts. To do this, we evaluated the realization of our architecture using various platforms for DTP. We also examined performance aspects and how to scale our system for large distributed systems.

3.1 TIP based realization

In [VKM97a] we proposed a realization of the transactional migration in our system based on the recently published *Transaction Internet Protocol* (TIP) [LEK97] by Microsoft and Tandem. The *Internet Engineering Task Force* (IETF) is currently standardizing TIP for the Internet. There exists also a publicly accessible reference implementation of TIP in JAVA [TIP97]. TIP includes one- and two-phase commit protocols developed for the Internet. TIP is rather easy to handle but offers less routines for transaction management and control. To use TIP in our system we proposed an enhancement for encryption and for the exchange of our specific data like the agent ID. We layered this extension between the TIP protocol and the TCP/IP stack.

3.2 OTS-CORBA based realization

Another very promising approach is the realization of the transactional migration in our system using the Common Object Broker Request Architecture (CORBA) [OMG95] with its *Object Transaction Service* (OTS) [OMG94]. In [VKM97b] the assignment of the OTS components to the instances of our system is described. The whole communication for our protocol is based on object calls using CORBA, so only interface specifications are necessary.

3.3 Performance Aspects

One drawback of distributed transaction processing results in a loss of performance. However, for most applications there are no real-time requirements for the transport of a mobile agent, so this problem is less important for our protocol. If no error occurs, the loss of performance in relation to a normal agent transfer without transactions is not significant. Exact performance data will be available with the first benchmarks of different mobile agent systems.

If one of the hosts crashes or the network connection breaks down, the transmission of the agent will be blocked. After a time-out the transaction will be aborted and the agent on the old host remains valid, and after a reconnection a new migration of the agent can be initiated. The protocol is non-blocking with respect to the overall agent system; only the actions of a specific agent will be delayed by an error in its migration phase.

3.4 Scaling for large distributed systems

The design of our architecture is suitable for scaling in large distributed systems like the Internet. Our trust service is responsible for a specific *agent domain*, which can cooperate with other domains. This concept is similar to CORBA or OTS where all instances, servers as well as clients, pertain to one domain. It is important to see that for instance, an OTS domain is absolutely independent of a CORBA domain. If several CORBA domains can interoperate, this enables one OTS domain to span the various CORBA domains [KVT96].

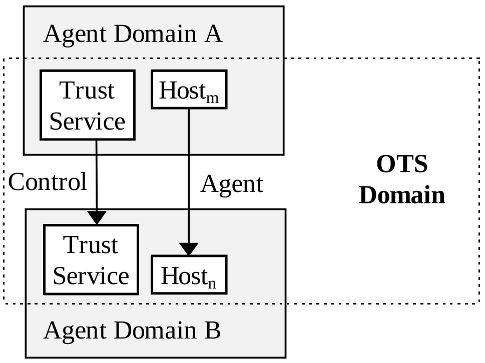

Fig. 4. Agent and OTS Domain

When an agent moves to a new agent domain, the control of this agent will be given to the new trust service. To preserve a consistence state of the agent system, the new trust service must to be involved in the agent transfer and the corresponding distributed transaction. Figure 4 illustrates the relationship between an agent and an OTS domain.

4 Conclusion and Future Work

We presented how mobile agents can be equipped with electronic commerce capabilities. These capabilities are part of an architecture for mobile agent security and fault tolerance which can be set up on top of existing mobile agent systems, e.g. as an enhancement of the “GO-Statement“. Our architecture is highly secure against external attacks and offers a high reliability because the implementation is based on standardized components.

As future work we see the development of mechanisms for agent control and management. With the data at the trust service about the agent and its itinerary we have the basis to implement suitable control and management functions. Another interesting topic for further work may be the evaluation of the specification for *Mobile Agent Facilities* (MAF) [MAF97] to conform our system to this standard.

Literature

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

[Bau97] J. Baumann: A Protocol for Orphan Detection and Termination in Mobile Agent Systems, Technical Report, TR-1997-09, University of Stuttgart, 1997

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

[DEC96] 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

[FKK96] A. O. Freier, P. Karlton, P.C. Kocher: *The SSL Protocol Version 3.0*, ftp://ietf.org/internet-draft/draft-ietf-tls-ssl-version3-00.txt, 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

[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 Securtity Programming*, ISBN 1-56592-134-8, O'Reilly & Accociates, 1995

[IBM96] IBM Research Hawthorne and Zürich: *Internet Keyed Payment Protocols (iKP)*, 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

[KVT96] T. Kunkelmann, H. Vogler, S. Thomas: *Interoperability of Distributed Transaction Processing Systems*, Proc. Int'l Workshop on Trends in Distributed Systems (TREDS'96), Aachen, Germany, Springer Verlag LNCS 1161, 1996

[LEK97] J. Lyon, K. Evans, J. Klein: *Transaction Internet Protocol*, Version 2.0, Internet-Draft, ftp://ds.internic.net/internet-drafts/draft-lyon-itp-nodes-04.txt, 1997

[MAF97] Joint Submission by General Magic Inc., GMD FOCUS, IBM Corp.: *Mobile Agent Facilities*, http://genmagic.com/agents/MAF/, 1997

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

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

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

[OMG95] Object Management Group: *The Object Request Broker: Architecture and Specification*, Revision 2.0, 1995

[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 International Workshop on Mobile Agents MA'97 Berlin, LNCS No. 1219, Springer Verlag, 1997

[PKL97] A. Park, A. Küpper, S. Leuker: *JAE: A Multi-Agent System with Internet Services Access*, Proc. of the 4th Int. Conference on Intelligence in Services and Networks IS&N'97, Cernobbio, Italy, Springer Verlag, LNCS 1238, 1997

[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

[VKM97a] H. Vogler, T. Kunkelmann, M.-L. Moschgath: *Distributed Transaction Processing as a Reliability Concept for Mobile Agents*, Proc. 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'97), Tunis, Tunisia, IEEE Computer Society 1997

[VKM97b] H. Vogler, T. Kunkelmann, M.-L. Moschgath: *An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transactions*, 1997 Int'l Conference on Parallel and Distributed Systems (ICPADS'97), Seoul, Korea, IEEE Computer Society 1997

[WoJe95] M. Woolbridge, N.R. Jennings: *Intelligent Agents: Theory and Practice*, Knowledge Engineering Review 10(2), 1995

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

[ZLU95] K. Zielinski, A. Laurentowski, A. Uszok: *Multi-Agent Model As an Extension to the Client/Server Processing Paradigm*, Proceedings of the Distributed Intelligent and Multi Agent Systems Workshop'95, November 1995