

Netlet – A New Architecture for Building Mobile Agent Systems

Ming Rutar and Shivakant Mishra
Department of Computer Science
University of Colorado, Campus Box 0430
Boulder, CO 80309-0430.
{rutar|mishras}@cs.colorado.edu

Abstract

With increasing popularity of the Internet and portable devices, the potential of mobile agent market has become more evident. However, despite the qualitative strength of mobile agent paradigm, such as its support for disconnected operation and mobility, mobile agent technology has not gained wide acceptance from commercial world. The main reason of this is the security issue. In this paper, we propose a new mobile agent system called Netlet. Netlet provides a more comprehensive security scheme. While it enables mobility of mobile agents, it also ensures the welfare of the host machine. Netlet preserves the essence of mobile agent technology and better meets security requirements of commercial applications.

1. Introduction

The explosive growth of the Internet and use of portable devices, such as cellular telephone, pager, personal digital assistant, laptops have imposed new computing needs that require new paradigms and new technologies. With features such as operation over a disconnected environment and support for mobility, mobile agent technology has a great potential in this scenario. However, the vision of mobile agent as the key technology for future electronic commerce applications can be realized only if all security issues are well understood and the corresponding mechanisms are in place [6].

This paper proposes a new mobile agent system – the Netlet. The key idea of Netlet is to tailor the current mobile agent approach to fit into traditional client-server computing paradigm, which has been well accepted and widely implemented in the commercial world. While Netlet provides agent mobility over the network, it restricts agent's accessibility on the host machine. An initial prototype of Netlet is currently being implemented on the recently released Microsoft .NET framework.

While in the last few years, many researchers have investigated the development of mobile agent systems, and several novel techniques addressing the design and implementation issues of mobile agent systems, the technology still hasn't gained wide acceptance in the commercial world. The main reason of this is the concern about different security issues that arise in a mobile agent system. There are three main classes of threats to security: disclosure of information, denial of service, and corruption of information. Four threat categories can be derived in a mobile agent system: 1) A hostile agent could attack an agent platform; 2) A hostile agent platform could attack an agent; 3) A hostile agent could attack another agent; 4) Other entities could attack the agent system. Theoretically, threats in categories 1 and 3 can be addressed by using the Java sandbox security model. However, unlike Java applets, mobile agents typically need to access local resources, and different agents typically have different resource requirements. This means managing sandbox configuration may impose a system-wide administration challenge. Threats in category 4 are not unique to a mobile agent system, and security techniques developed to protect other middleware systems can be used to address them. Threats in category 2 are extremely difficult to address completely. An agent owner must have some level of trust in a foreign agent host, before she can allow her agent to run on that host.

From commercial world point of view, it is the threats in categories 1 and 3 that make the acceptance of mobile agent paradigm less likely. The key issue here is that the techniques developed to address these threats typically require agent hosts to install a completely new middleware system, and may require application service providers to update their implementations to conform with the new middleware requirements.

Besides security consideration, lack of appropriate transactional support is another shortcoming of mobile agent systems. This is because a considerable number of today's commercial applications require a high degree of robustness. Since an agent may spawn new agents, may die, may migrate, precise synchronization among agents is difficult to achieve. But exact-once semantics for task

processing is an important feature for majority of net-based applications [6].

2. Netlet: Motivation

2.1 Mobile Agent Architecture

By definition, a software agent is a software entity that exists in a software and it must contain all of the following components: an agent model, a life-cycle model, a computational model, a security model, a communication model and finally a navigation model. A mobile agent is a software agent that moves from one computer to another either as directed by a user or autonomously. So, what is the software environment in which mobile agents live? A mobile agent environment is a software system, which is distributed over a network of heterogeneous computers, providing an environment in which mobile agents can execute. The mobile agent environment implements the majority of the models, which appear in the mobile agent definition. It may also provide: support services which relate to the mobile agent environment itself, support services pertaining to the environments on which the mobile agent environment is built, services to support access to other mobile agent systems, and finally support for openness when accessing non-agent-based software environments [2]. Figure 1 illustrates the basic mobile agent architecture.

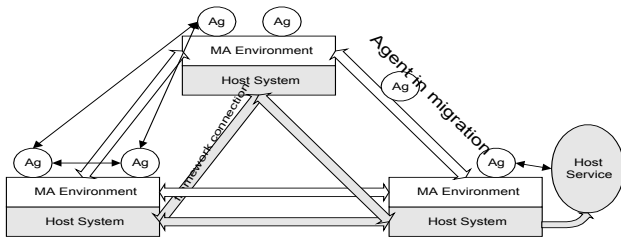


Figure 1: Basic mobile agent architecture

2.2 An analysis of current mobile agent computing

Client-server architecture has been well accepted in commercial world. Some major attractions of this architecture are security, configuration flexibility and manageability. Figure 2 illustrates client-server computing scenario where client can only accesses server application residing at a server machine through the interface provided by the server application. Other server resources, such as the Data Source, may be inaccessible, even invisible, to the client depends on the server security configuration. In many cases, the user of a client

application cannot directly access the server machine. The benefits of this approach are: 1) if a server machine hosts multiple server applications with different security scope and requirements, this architecture would not risk information disclosure because the client applications can only interface with its corresponding server applications. 2) As long as the interface remains the same, modification of server application and reconfiguration of data source will not effect client application, which normally requires more extensive administration because of its widely spreading over computers.

However, in current mobile agent architecture, once a client dispatches a mobile agent over to a server machine, the agent's accessibility to the server machine is defined by the mobile agent platform residing at the server machine. An agent may be able to access data source, or not be able to connect to the server application as illustrated in Figure 3.

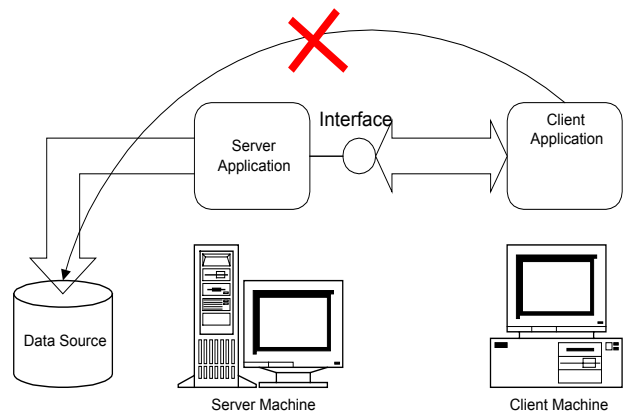


Figure 2: Client's access in client-server application

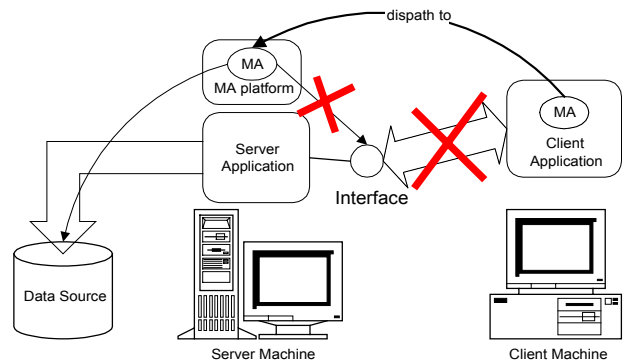


Figure3: Client dispatches a mobile agent over to the server machine.

If the platform constrains agent's access to the server application, it defeats the purpose of dispatching an agent over to the server, but doesn't endanger server's well been. However, the possibility of increasing agent's accessibility at host machine imposes serious security threats. Moreover, this possibility makes mobile agent an attraction for malicious platform attack because the platform could modify the agent's code and send the agent to probe security holes of some other sites. Although the sandbox security model is able to solve the problem, as we stated earlier, configuring sandbox may be challenge. For example, if the server machine hosts several server applications with different security requirements, setting up security could become an administrative nightmare, and a slight ignorance may lead to disastrous consequences.

The current mobile agent platform advocates a pro-agent environment and leaves security burden to the service providers. This approach lacks appeal to the industry, because the host machine is likely a heavy-weight server in terms of its processing power, amount of loaded software, and its importance in the enterprise. Changing configuration may distract normal operation.

Another issue with current mobile agent system is that the current platform opens its own communication channels (see Figure 1). In an industry, most companies install firewalls to protect their networks. To accommodate mobile agents, a company needs to open "holes" in the firewall if it is in the Internet environment. Such action may compromise company's security policy.

2.3 Netlet Approach

We strongly feel that putting the security and reliability burden on the service providers is inappropriate, because mobile agents consume their resources. We propose a mobile agent platform that ensures a mobile agent's access to a server machine to no more or no less than the agent's owner. Figure 4 illustrates Netlet approach.

Our solution is straightforward. The main aspects of Netlet are listed as following:

1. Netlet restricts mobile agent's accessibility to a server machine by using sandbox or similar mechanism. It applies agent security policy uniformly to all hosted agents.
2. A mobile agent gets object reference of its application server from the Netlet.

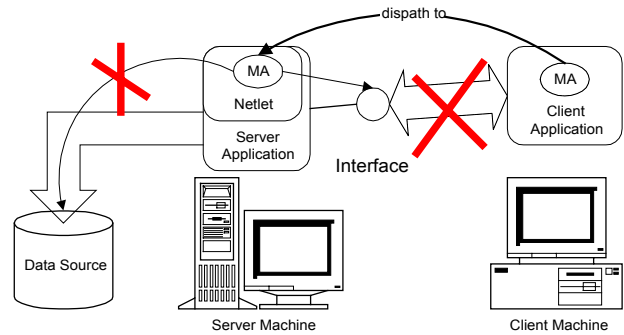


Figure4: A mobile agent has no more or less access to the server machine than its owner.

3. Netlet doesn't have its own communication facility. Rather it utilizes the communication capability of the underlying application through a set of interfaces.

Unlike traditional mobile agent platform, Netlet is not a standalone executable, rather a layer added to server application providing mobile agent hosting capability.

2.4 How do we address the security issue?

Other than interacting with one or more application server, as in a client/server model, a Netlet agent's access on a server machine is constrained by the Netlet security policy, which applies to all mobile agents hosted by the Netlet platform. If the policy is set properly, it is likely that a server machine doesn't need to take any additional security measures to accommodate mobile agents. The restriction imposed on Netlet agents makes Netlet agent unattractive for abuse. Let's look at the following scenario: a malicious host is interested in stealing some confidential information residing at host "A", so it modifies an innocent agent's code and sends the agent over to host "A". In the traditional mobile agent implementation, an agent may be able access confidential information and send it back to the malicious host as the malicious host had planned. However, in the Netlet environment, other than interacting with host "A"'s accessible interfaces, which do not disclose confidential data, an agent cannot do much more. Therefore, the motivation of attack agent has diminished.

However, some special-purpose agents, such as a software installation agent may require accesses to several resources that are not accessed by a typical agent. The Netlet solution in this scenario is configuring two separate Netlet platforms, each with its own security policies according to the security requirements of the two cases.

We believe that this solution is reasonable, because it is similar to differentiating the Internet and intranet web servers due to the different security requirements.

3. Netlet Architecture

The Netlet mobile agent platform has two layers: mobile agent execution environment layer and mobile agent platform layer. The communication plug-in is an adapter connecting the Netlet platform with application communication facility. Figure 5 illustrates the architecture. The components colored as cyan belongs to Netlet platform. The communication is a gray area because the Netlet defines the interfaces and may provide default implementation for common mechanism, such as TCP/IP.

The mobile agent execution environment layer consist agent execution context and agents. Because a context is dynamically created by platform and has the life cycle as the agent it hosting, the agent execution environment layer only exists when there is at least one agent living in the platform.

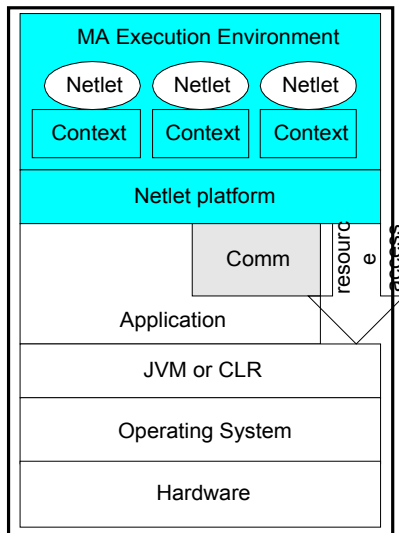


Figure 5: Netlet architecture

In our preliminary design, the Netlet platform contains the following facilities: Security Manager, Resource Manager, Access Control Manager and Message Routing Manager (see figure 6).

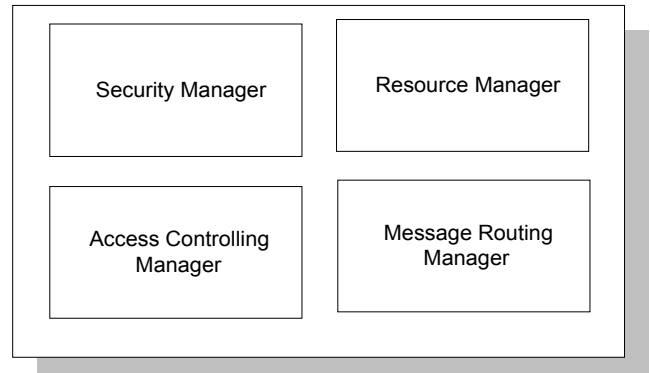


Figure 6: Platform components.

The Security Manager verifies eligibility of mobile agents. The Resource Manager allocates resource for agents, monitors agent resource consuming and ensures the health of the hosting application. The Message Routing Manager delivers and sends agent’s messages utilizing application’s message subsystem. The Access Controlling Manager enforces access rules defined by applications.

By not opening any communication channels on its own, the Netlet platform would give application server providers a sense of security control since all messages will be verified by layers of security enforcement facilities, which are already in place. It is especially important for messages passing through the firewall.

3.1 Netlet agent execution

Let us look at an example of an agent execution in Netlet.

- An agent arrives at a host.
- If policy requires Netlet platform to check the eligibility of the arriving agents, the security manager performs the verification.
- If the agent is not eligible to run on the host machine, it could either be dispatched to a destination of its choice or be terminated by the platform. The platform queries agent’s plan on the host. The reason for this action is that if the required service is no longer provided at this host, the host shouldn’t accept this agent. Also if the platform knows where the service has moved, it could dispatch the agent to the current location.
- If the agent is qualified to execute at the host, the platform creates a context for the agent.

- The agent’s direct access to the host is limited by the security policy.
- An agent doesn’t contact the platform directly. Rather it communicates through its context.
- Once an agent receives the reference of its server application, it interacts with the server in the same fashion as the remote client. If the application server requires additional security checks, such as login, the agent needs to provide the appropriate login information.

3.2 Agent Termination

An agent doesn’t always determine its life cycle at the host. An agent could die naturally as a result of being dispatched to the next stop or finishing the journey, or could get killed by the platform. If the host resources are running low, the platform needs to terminate agents to free resources. The platform will notify the selected agents, and agents can leave gracefully. If the platform detects abnormal behavior of an agent, such as consuming large amounts of CPU time, the platform would assume that the agent is either buggy or has malicious intentions, and would kill that agent.

4. Netlet Mobile Agent Service

While Netlet ensures the welfare of the host machines, it also preserves the essence of mobile agent technology - agent mobility. Figure 7 illustrates the logical mobile agent infrastructure over the network. The mobile agent infrastructure is not constrained by the boundaries of an autonomous system or the Internet. Any host with Netlet platform can become part of this infrastructure. In the other words, agent mobility is applied uniformly across the infrastructure.

While an agent can move around the infrastructure, it cannot always access a host machine it visits. This is because the host Netlet platform that enforces security according to some security policies, controls the agent’s access to the host. If the agent cannot use any of the services offered by the host server applications, the agent cannot live on that host.

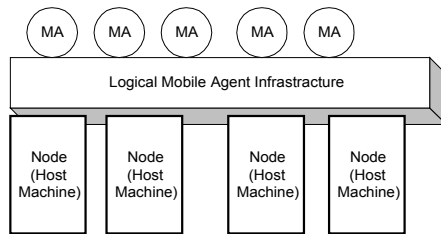


Figure 7: The Mobile Agent Infrastructure

As we pointed out earlier, the Netlet platform is not an executable itself. It is a layer added to an application. If we plug a Netlet platform into mobile agent application, then we have a mobile agent service (see Figure 8).

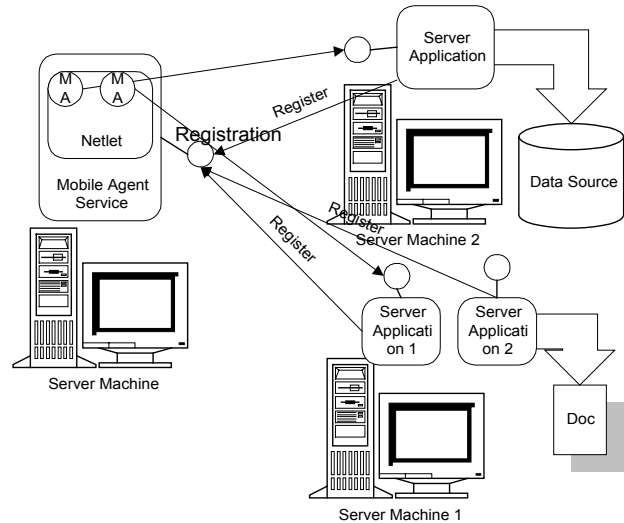


Figure 8: Mobile Agent Service

The mobile agent service is an attractive service for application providers, because by registering with the service, an application server can serve portable clients with very little code modification. The mobile agent service can have a GUI interface for easy configuration, such as setting security policies, resource allocation policies, and visual monitoring of mobile agent activities and overall host health. Another interesting feature of the service is providing QoS service, which could include member privilege, fault tolerance etc.

We like to emphasize the importance of fault tolerance. Fault tolerance is an expensive service in mobile agent computing paradigm [3]. However, a robust environment is the foundation for transactional operation. As we know, a considerable part of today’s commercial applications involve transactions. Without transaction capability, mobile agent applications will be limited to very simple applications such as weather report retriever or cheapest computer finder. In order for mobile agent technology to be considered for doing serious business, the mobile agent service needs to provide adequate fault tolerance.

Netlet platform has several advantages over current mobile agent platform:

- The platform could be easily integrated with existing commercial client/server applications, so

the applications can easily serve portable clients as well.

- Client's mobile code neither gains any new access, nor loses any access authorized by the server.
- The platform adapts to its application's network environment and doesn't require additional communication channel.

5. Netlet Implementation

The design goals of Netlet are stated as following:

- Safety and welfare of host machines are the primary concerns.
- No restrictions on agent mobility.
- The system must be configurable, scalable, flexible and extensible.
- Adapt to .NET programming paradigm and utilize .NET facilities as much as possible.
- Provide comprehensive and easy to use APIs for development.

5.1 Netlet Object Model

Figure 9 shows the object model of major Netlet components.

- Netlet is an abstract class. All Netlet agents must derive from this class.
- Context is execution environment for mobile agents created at runtime by Netlet platform.
- NetletPlatform is a .NET remote object, which is the key component of the system described in this paper.
- ChannelAdapter lies between netletPlatform and application native communication mechanism providing uniform communication interface to Netlet platform.
- AppAdapter is something similar to ChannelAdapter but for application.
- Message defines the format of message object.
- Principal contains agent owner's information.
- Plan is agent's itinerary.
- AccountInfo contain account information for an agent to access various application services.
- State is agent's execution state, which must be captured and moved with the agent if we implement strong migration.

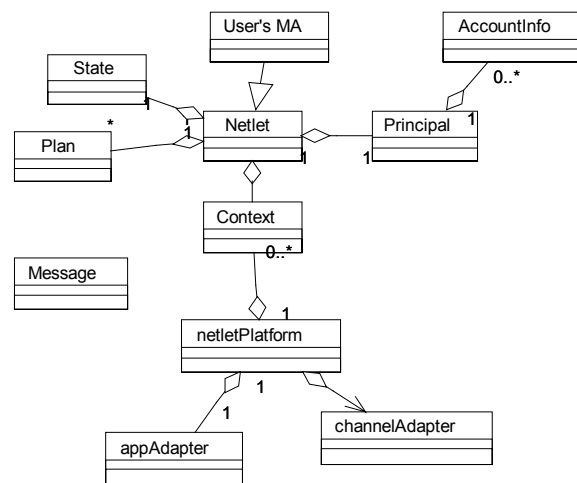


Figure 9: Netlet Object Model

5.2 Netlet Security Implementation

Figure 10 represents the complete model of security services when a mobile agent migrates to a new node. The following three steps, enforced by the security manager are executed:

1. Check the authentication, confidentiality, and integrity of the agent.
2. Bring in the appropriate application and according to the principal, generate the security policy under the role-based security.
3. Run the agent under the control of the policy.

6. Conclusion

New technologies are needed to address new challenges due to the fast growth of the Internet and explosive growth of portable devices. Due to its salient properties, mobile agent technology has promised a new distributed computing paradigm meeting these challenges. However, the important issue of security has prevented this technology from wider commercial acceptance. In this paper, we have proposed a new mobile agent platform that suits security environment of the industry and preserves the mobility of mobile agents. We have analyzed the problem with current mobile agent approach, described the Netlet architecture, explained why Netlet's approach would resolve the security issue, and designed a high-level object model. We believe that the Netlet approach, with its comprehensive agent access control and adaptive communication scheme, makes more sense in the commercial world.

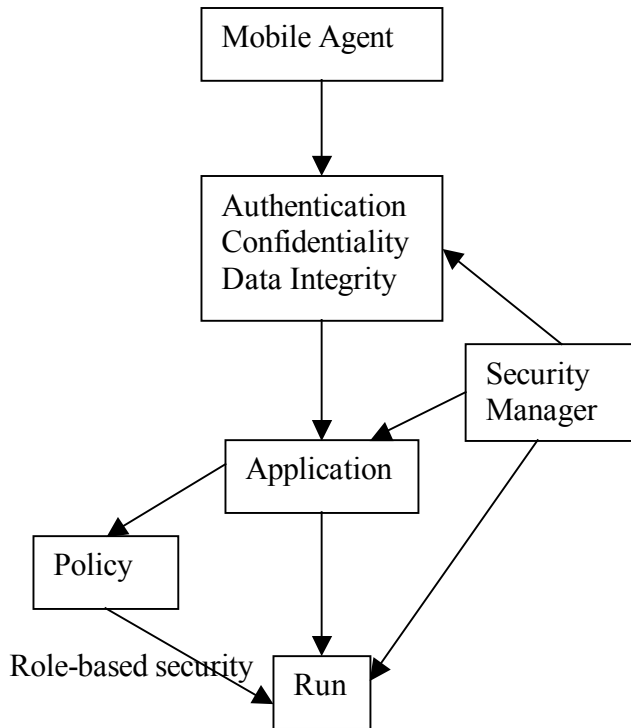


Figure 10: Agent execution steps.

References

- [1] D. Chess, C. Harrison, and A. Kershenbaum. "Mobile Agents: Are they a good Idea?". IBM Research Report RC 19887, 1995.
- [2] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers, R. Evans, "Software Agents: A Review," Technical report TCD-CS-1997-06, Trinity College Dublin, May 1997.
- [3] S. Mishra, Y Huang and H Kuntur. "Dependability Support in DaAgent Mobile Agent System." Proceedings of the ISCA 8th International Conference on Intelligent Systems, Denver, CO (June 1999).
- [4] R. Gray, D. Kotz, G. Cybenko, and D. Rus. "Mobile agents: Motivation and State of Art." (Jeffrey Bradshaw ed.). Handbook of Agent Technology, AAAI/MIT Press, 2000.
- [5] C. Thompson and J. Odell. "Agent Technology Glossary.". Online posting. www.objs.com/agility/tech-reports/9909-agent-glossary.html. September, 1999.
- [6] K. Rothermel, F Hohl, and N Radouniklis. "Mobile Agent Systems: What is Missing?". Proc. of IFIP WG 6.1 International Working Conference on Distributed

Applications and Interoperable Systems. Cottbus, Germany. 1997

- [7] D. Joseph, A. deLespinasse, J. Tauber, D. Gifford and M. Frans Kaashoek, Rover: "A toolkit for mobile information access." In the proceedings of the 15th ACM Symp. on Operating Systems Principles, December 1995.
- [8] L. Hurst, P. Cunningham and F Somers. "Mobile agents Smart messages." Proceedings of the 1st International Workshop on Mobile Agents, April 1997, Berlin, Germany.
- [9] J. Ritcher. "Applied Microsoft .NET Framework Programming". Wintellect. Microsoft Press. 2002.
- [10] S. Robinson, K Allen, O. Cornes, J. Glynn, Z. Greenvoss, B. Harvey, C. Nagel, M. Skinner and K. Watson. "Professional C#, 2nd Edition". Wrox Press Ltd. 2002.
- [11] Microsoft Development Network (MSDN). Online posting. <<http://msdn.microsoft.com>>.
- [12] IBM Aglet Woekbench. Online posting. <<http://www.tri.ibm.co.jp/aglet/>>.
- [13] Wayne Jansen, Tom karygiannis. "NIST Special Publication 800-19-Mobile Agent Security." Online post. <http://csrc.nist.gov/mobileagents/publication/sp800-19.pdf>
- [14] M. Sonntag and R. Hormanseder. "Mobile agent security based on payment." Institute for Information Proces.
- [15] Y. Kun, G. Xin, and L. Dayou. "Security in Mobile Agent System: Problems and Approaches." Operating Systems Review. 1/2000,21-28
- [16] N. Karnik. "Security in Mobile Agent Systems." Online posting. <<http://www.cs.umn.edu/Ajanta/defense/>>