Network Processing of Mobile Agents, by Mobile Agents, for Mobile Agents

Ichiro Satoh

National Institute of Informatics /
Japan Science and Technology Corporation
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
Tel: +81-3-4212-2546 Fax: +81-3-3556-1916
E-mail: ichiro@nii.ac.jp

Abstract. This paper presents a framework for building network protocols for migrating mobile agents over a network. The framework allows network protocols for agent migration to be naturally implemented within mobile agents and to be constructed in a hierarchy as most data transmission protocols are. These protocols are given as mobile agents and they can transmit other mobile agents to remote hosts as first-class objects. Since they can be dynamically deployed at remote hosts by migrating the agents that carry them, these protocols can dynamically and flexibly customize network processing for agent migration according to the requirements of respective visiting agents and changes in the environments. A prototype implementation was built on a Java-based mobile agent system, and several practical protocols for agent migration were designed and implemented. The framework can make major contributions to mobile agent technology for telecommunication systems.

1 Introduction

Mobile agent technology is an emerging technology that makes it much easier to design, implement, and maintain telecommunication systems. The technology can be used in the development of various network applications. These applications often require application-specific network processing for migrating their agents over a network. For example, a typical application of the technology is network management, where an agent travels to multiple nodes in a network to observe and access the components locally. The itinerary of such a monitoring agent seriously affects the achievement and efficiency of its tasks. Moreover, a mobile agent for electronic commerce may have to to be transformed into an encrypted bit stream before it can transfer itself over a network. However, existing mobile agent systems assume particular network infrastructures and cannot dynamically adapt their own network processing to the requirements of visiting agents and to changes in their environment.

This paper addresses the dynamic customization of network processing for agent migration, rather than for data transmission. I describe a new framework for dynamically deploying and changing network protocols for agent migration. My framework is based on two key ideas. The first is to apply active network technology to a network infrastructure for mobile agents. The second is to construct network protocols for agent migration within the agents themselves. That is, my mobile agent-based protocols can transmit mobile agents as first-class objects to their destinations. Also, the protocols can be dynamically deployed by the migration of the agents that support these protocols. Therefore, my framework allows network processing for mobile agents to be adapted to the requirements of visiting agents and to changes in the environment. The framework can provide a useful testbed for implementing and evaluating different types of network processing for mobile agents.

In this paper I survey related works (Section 2), describe the design goals of my framework (Section 3), briefly review my mobile agent system, called *MobileSpaces* (Section 4), present several mobile agent-based protocols for agent migration (Section 5), show some real-world examples of the framework, and make some conclusions and describe research directions for developing new protocols.

2 Background

Many mobile agent systems have been developed over the last few years, for example, Aglets [10], Telescript [16], and Voyager [11]. To my knowledge, none can dynamically extend and adapt their network processing for agent migration to the characteristics of current networks and the requirements of respective visiting agents, although mobile agents must be used in heterogeneous and dynamic network environments, for example, in personal mobile communication, wireless networks, and active networks. This is because their agent migration protocols are statically embedded inside their systems.

A mobile agent, which visits multiple hosts to perform its task, must have an application specific itinerary. For example, a mobile agent may roam over more than one host without making any detours or may have to return to its home host after each hop instead of proceeding another destination. Also, a network-dependent itinerary is often needed for a mobile agent to travel to multiple hosts efficiently. However, it is difficult to determine such an itinerary at the time the agent is designed or instantiated because the network topology cannot always be known. Also, even if the itinerary of a mobile agent was optimized for a particular network to travel to multiple hosts efficiently, it might not be reused in another network. To overcome this problem, ADK [8] separates the travel itinerary of an agent from its behavior by building a mobile agent from a set of component categories: navigational components responsible for a travel itinerary and performer components responsible for executing one or more management tasks on each node. Aglets [10] introduces the notion of an itinerary pattern, which is similar to design patterns in software engineering, to shift the responsibility for navigation from an application-specific agent to a framework library described in [1].

Both approaches allow us to design the application-specific itinerary for an agent independent of the logical behavior of the agent, but the itinerary parts must be statically and manually embedded in the agent. Consequently, the agent cannot dynamically change its itinerary and cannot travel beyond its familiar networks.

There have been many attempts to apply mobile agent technology to the development of active networks [2, 4] because mobile agents can be considered a special case in mobile code technology, which is the basis of existing active network technologies. For example, the Grasshopper system offers an active network platform consisting of stationary and mobile agents as service entities for telecommunication. In contrast, the framework presented in this paper applies active network technology to mobile agent technology.

I described a portable and extensible mobile agent system, MobileSpaces, in my previous paper [12]. The system serves as the basis for the framework presented in this paper. It can dynamically adapt its functions and structures to changes in the environments. Also, I presented an architecture for building adaptive protocols in [14]. While in the previous papers I did not foucs on any approach to building application-specific protocols for agent migration, the goal of this paper is to design and implement a layered architecture for building and deploying configurable protocols for agent migration and present several protocols for agent migration.

3 Approach

The goal of the framework presented in this paper is to provide a self-configurable infrastructure for agents migrating over a network. This section outlines the overall architecture of the framework and describes the basic idea of network protocols based on the framework.

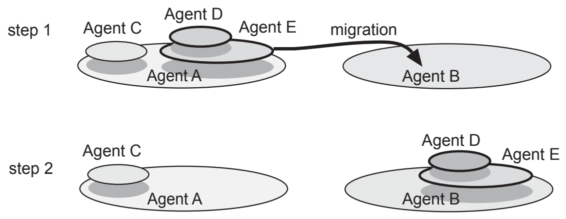

Fig. 1. Agent hierarchy and inter-agent migration.

3.1 Mobile Agents as First-class Objects

Mobile agents are autonomous programs that can travel between different computers. In the framework presented in this paper, mobile agents are computational entities like other mobile agents. When an agent migrates, not only the code of the agent but also its state can be transferred to the destination. The framework is built on a mobile agent system, called MobileSpaces, presented in [12]. The system is characterized by two novel concepts: **agent hierarchy** and **inter-agent migration**. The former means that one mobile agent can be contained within another mobile agent. That is, mobile agents are organized in a tree structure. The latter means that each mobile agent can migrate to other mobile agents as a whole, with all its inner agents, as long as the destination agent accepts it, as shown in Fig. 1. A container agent is responsible for automatically offering its own services and resources to its inner agents, and it can subordinate its inner agents. Therefore, an agent can transmit its inner agents to another location as first-class objects [5], in the sense that mobile agents can be passed to and returned from other mobile agents as values. As a result, network protocols for agent migration can be implemented within mobile agents.

3.2 Layered protocols for agent migration.

Most protocols for data transmission are often arranged in a hierarchy of layers. Each layer presents an interface to the layers above it and extends services provided by the layer below it. The hierarchical structure of mobile agents enables network protocols for agent migration to be organized hierarchically. That is, each agent hierarchy consisting of mobile agent-based protocols can be viewed as a protocol stack for agent migration, as shown in Fig. 2, and agent migration in an agent hierarchy is introduced as a basic mechanism for accessing services provided by the underlying layer. Mobile agent-based protocols in the bottom layer correspond to data-link layered protocols. They are responsible for establishing point-to-point channels for agent migration between neighboring computers. The middle layer corresponds to routing protocols for agent migration and provides a mechanism to transmit mobile agents beyond the channels between directly connected nodes. The framework enables routing protocols for agent migration to be performed by mobile agents.

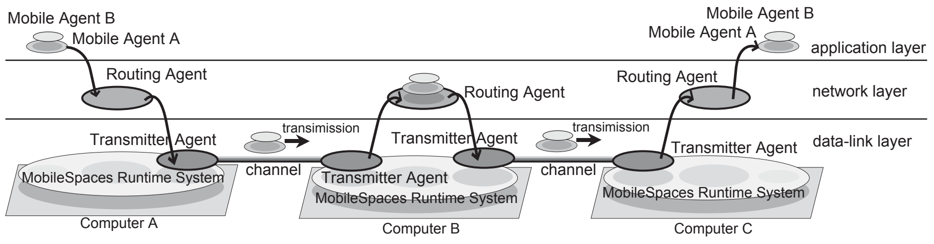

Fig. 2. Architecture of mobile agent-based protocols for agent migration.

4 MobileSpaces: An Extensible Mobile Agent System

This section briefly reviews MobileSpaces, which provides, in addition to mobile agent-based applications, an infrastructure for building and executing mobile agents for network processing. MobileSpaces is built on a Java virtual machine and mobile agents are given as Java objects. Its architecture is designed based on a micro-kernel architecture and consists of two parts: a core system and higher-level components. The former offers only minimal and common functions, independent of the underlying environment. The latter is a collection of higher-level components outside the core system that provide other functions, including agent migration over a network, which may depend on the surrounding environment.

4.1 Core System

Each core system is made as small as possible for portability. It has only three functions.

Agent Hierarchy Management: Each core system corresponds to the root node of an agent hierarchy, which is maintained as a tree structure in which each node contains a mobile agent and its attributes. Agent migration in an agent hierarchy is performed simply as a transformation of the tree structure of the hierarchy.

Agent Execution Management: Each agent can have more than one active thread under the control of the core system. The core system maintains the life-cycle state of agents. When the life-cycle state of an agent is changed, for example, at creation, termination, or migration, the core system issues certain events to invoke certain methods in the agent and its containing agents.

Agent Serialization and Security Management: The core system has a function for marshaling agents into bit streams and unmarshaling them later. The current implementation of the system uses a Java object serialization package for marshaling the states of agents, so agents are transmitted based on the notion of weak mobility [6]. The core system verifies whether a marshaled agent is valid or not to protect the system against invalid or malicious agents, by means of Java's security mechanism.

4.2 Mobile Agent Program

Each mobile agent consists of three parts: a body program, context objects, and inner agents as shown in Fig. 3. The body program is an instance of a subclass of abstract class `Agent`. This class defines fundamental callback methods invoked when the life-cycle of a mobile agent changes due to creation, suspension, marshaling, unmarshaling, destruction etc., like the delegation event model in Aglets [10]. It also provides a command for agent migration in an agent hierarchy, written as `go(AgentURL destination)`. When an agent performs the command, it migrates itself to the destination agent specified by the argument of the command in the same agent hierarchy. An inner agent cannot access any methods defined in its container agent, including the core system. Instead, each container is equipped with a context object that offers service methods in a subclass of the `Context` class, such as the `AppletContext` class of Java's Applet. These methods can be indirectly accessed by the inner agents of a container to get information about and interact with the environment, including the container, sibling agents, and the underlying computer system.

5 Mobile Agent-Based Protocols for Agent Migration

Since this framework can treat mobile agents as first-class objects, various types of network processing for mobile agents can be implemented as special mobile agents, called service agents, running on the core system of MobileSpaces. These service agents are hierarchically organized as a protocol stack.

- Each service agent is designed to provide its service to its inner mobile agents. Therefore, each service agent in a lower layer can be viewed as a service provider for agents in an upper layer. The movement of an agent to a service agent in a lower layer in the same agent hierarchy corresponds to the process of applying the network service of the service agent to the moving agent.

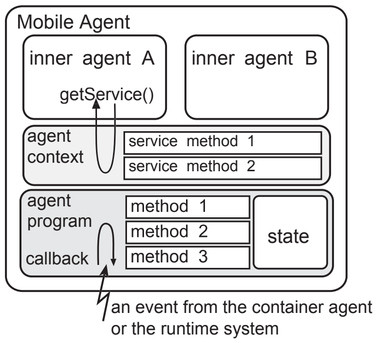

Fig. 3. Structure of a Hierarchical Mobile Agent.

- Each runtime system permits one service to be provided by one or more service agents. That is, different network protocols can be supported by different service agents. Moving agents or upper-layer protocols can dynamically select a suitable agent for their requirements and migrate their inner agents to the selected agent.
- Since service agents for performing protocols are still mobile, the protocols can be dynamically deployed at hosts by migrating the agents to the hosts.

Hereafter, I present several basic protocols for agent migration. Since these protocols are given as abstract classes in the Java language, we can easily define further application-specific protocols by extending these basic protocols.

5.1 Point-To-Point Channels for Agent Migration

Agent migration between neighboring hosts can be provided by mobile agents, called *transmitters*. They are responsible for establishing point-to-point channels for agent migration and can automatically exchange their inner agents through their common communication protocol. After an agent arrives at a transmitter agent from an upper layer, the arriving agent indicates its final destination. The transmitter suspends the arriving agent (including its inner agents), then serializes its state and codes. Next, it sends the serialized agent to a coexisting transmitter agent located at the destination. The transmitter agent at the destination receives the data, reconstructs the agent (including its inner agents), and migrates it to the destination or specified agents for offering upper-layer protocols.

5.2 Routing Mechanisms for Agent Migration

Application-specific mobile agents often need to travel to multiple hosts to perform their tasks. However, it is difficult to determine the itinerary at the time the agent is designed or instantiated. Therefore, I introduce two approaches to determining and managing the itinerary of agents. These approaches are based on transmitter agents running on hosts and correspond to different kinds of application-specific routing protocols.

Forwarder Agent: The first approach provides a function similar to that of an active node (also called a programmable node) in active network technology. I introduce a service provider, called a *forwarder* agent, for redirecting moving agents to new destinations. Each forwarder agent holds a table describing part of the structure of the network and can be dynamically deployed at a host. When receiving agents, it can propagate certain events to its visiting agents instructing them to do something during a given time period and then redirects the agents to their destinations through point-to-point channels established among multiple hosts as shown in Fig. 4. Each forwarder agent will repeat the entire process in the same way until its visiting agents arrive at their destinations.

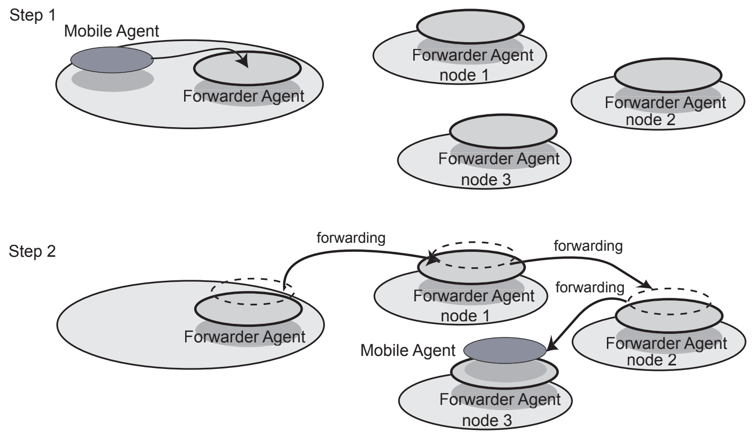

Fig. 4. Routing agents for forwarding the next hosts.

Navigator Agent: The second approach is similar to the notion of an active packet (also called a programmable capsule) in active network technology. Existing mobile agents can move from one host to another under their own control, as active packets can define their own routing. I propose a service provider, called a *navigator*, to convey inner agents over a network, as shown in Fig. 5. Each navigator agent is a container of other agents and travels with them in accordance with a list of hosts statically or algorithmically determined, or dynamically based on the agent's previous computations and the current environment. That is, a navigator agent can migrate itself to the next place as a whole, with all its inner agents. Upon its arrival at the place, the navigator propagates certain events to its inner agents. After the events have been processed by the inner agents, the navigator continues with its itinerary.

5.3 Protocol Distribution

Given a dynamic network infrastructure, a mechanism is needed for propagating mobile agents that support protocols to where they are needed. The current implementation of

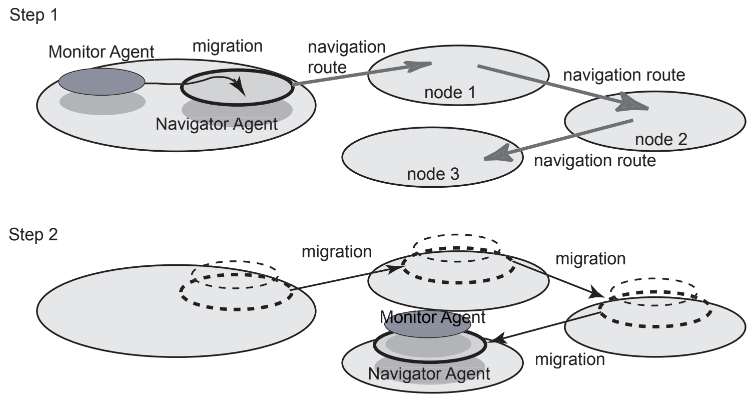

Fig. 5. Navigator agent with its inner agents for traveling among hosts.

this framework provides the following three mechanisms: (1) mobile agent-based protocols autonomously migrate to hosts at which the protocols may be needed and remain at the hosts in a decentralized manner; (2) mobile agent-based protocols are passively deployed at hosts that may require them by using forwarder agents prior to using the protocols as distributors of protocols; and (3) moving agents can carry mobile agent-based protocols inside themselves and deploy the protocols at hosts that the agents traverse. These mechanisms can improve performance in the common case of agent migration, i.e., a sequence of agents that follow the same path and require the same processing. All the mechanisms are managed by mobile agents, instead of by a runtime system. As a result, the deployment of transmitter agents must to be performed by other transmitter agents.

5.4 Current Status

The framework presented in this paper and its mobile agent-based protocols were implemented on MobileSpaces in the Java language. They can be run on any computer with a JDK 1.2-compatible Java runtime system. The framework provides several useful libraries for constructing network protocols within mobile agents. Several mobile agent-based protocols were developed, in addition to the protocols presented in the next section. They include agents for establishing channels through TCP, HTTP, and SMTP, forwarder and navigator agents for traveling among multiple hosts according to their own static routing tables and SNMP agents at each hosts. The current implementation of this framework was not built for performance. However, in order to compare two routing protocols, the forwarder agent protocol and the navigator agent protocol, I measured the per-hop latency in microseconds and the throughput of a single node in agents per second in a network consisting of eight PCs (Intel Pentium III-600 MHz with Windows 2000 and JDK 1.3) connected by 100-Mbps Ethernet via a switching hub. In both cases, I migrated a minimal-size agent that consisted of only common callback methods invoked at changes in its life-cycle state by the runtime system. The size of the moving agent was about 4 Kbytes (zip-compressed). For reference, I measured the time of migrating the agent in an agent hierarchy and between two hosts. The time of migrating the agent in an agent hierarchy was 5 ms, including the time of checking whether the visiting agent was permitted to enter the destination agent. In this experiment, agent migration between neighboring computers was performed by using simple TCP-based transmitter agents. The per-hop latency of migrating the agent between two computers was 34 ms per hop and the throughput was 10.8 agents per second. The latency is a sum of marshaling, compression, opening a TCP connection, transmission, acknowledgment, decompression, and security verification.

The per-hop latency of migrating the agent using a simple forwarder agent running on the hosts was 38 ms per hop and the throughput was 9.2 agents per second. The forwarder agent determines the host that its inner agents will visit at the next hop according to its own routing table. In contrast, the per-hop latency of migrating the agent using a simple navigator agent running on the computers was 42 ms per hop and the throughput was 8.3 agents per second. The navigator agent migrated itself and its inner agents to the hosts sequentially by incorporating itself into a transmitter agent.

In this preliminary experiment, the forwarder protocol was better than the navigator protocol, because the latter protocol had to migrate not only the target agent but also the protocol itself. Also, in both protocols when more than one agent was migrated on a network, the congestion of each computer was occasionally unbalanced, because these agent-based protocols are performed asynchronously. All the above results were measured in a trial without any performance optimization and are thus difficult to evaluate. However, the overhead of the mobile agent-based protocols in terms of the latency of each agent migration was reasonable for a high-level prototype of application-specific protocols for agent migration, rather than for data communication. The throughput of each agent migration was limited by the security mechanism of the MobileSpaces system rather than by the protocols. I believe that the current throughputs are fast enough for the deployment of mobile agent-based applications.

6 Examples

This section describes three practical examples of this framework to demonstrate how it can be used.

6.1 Network Management System

A typical application of mobile agents is as a monitoring system for network management. A discussion on the suitability of mobile agents in network management can be found in [3, 9]. A system for locally monitoring equipment located at hosts in more than one network was constructed. The system consists of a monitor agent and navigator agents. The monitor agent has no mechanism for its own itinerary and thus is not dependent on any network. In contrast, each navigator agent is optimized for navigating in each of the networks and is responsible for periodically traveling among hosts in its networks. When a monitoring agent is preparing to monitor a network, it enters a navigator agent designed for that network. The navigator then generates an efficient travel plan to visit certain hosts in the network. Next, it migrates itself and the monitoring agent to the hosts sequentially. When it arrives at each destination, it dispatches certain events to its inner agents.

6.2 Locating Mobile Agents

When an agent wants to interact with another agent, it must know the current location of the target agent. Therefore, a mechanism for tracking a moving agent is needed. An extension of the forwarder agent approach presented in the previous section offers such a mechanism, as shown in Fig. 6. Just before an agent moves into another agent, it creates and leaves a forwarder agent behind. The forwarder agent inherits the name of the moving agent and transfers its visiting agent to the new location of the moving agent. Therefore, when an agent wants to migrate to another agent that has moved elsewhere, it can migrate into the forwarder agent instead of the target agent. The forwarder agent then automatically transfers it to the current location of the target agent. Several schemes for effectively locating mobile agents have been explored in the field of process/object migration in distributed operating systems. Forwarder agents can easily support most of these schemes because they are programmable entities and can flexibly negotiate with each other through their own protocols.

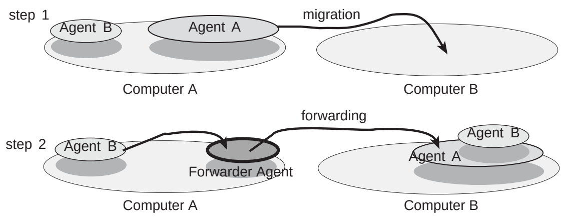

Fig. 6. Locating agents to locate moving agents.

6.3 Agent Migration in Mobile Computing

Mobile agent technology has the potential to mask disconnections in some cases. This is because once a mobile agent is completely transferred to a new location, the agent can continue its execution at the new location, even when the new location is disconnected from the source location. However, the technology cannot often solve network failures in the process of agent migration. That is, agents can be migrated from the source to the destination when all the links from the source to the destination are established at the same time. However, mobile computers do not have a permanent connection to a network and are often disconnected for long periods of time. When a mobile agent on a mobile computer wants to move to another mobile computer through a local-area network, both computers must be connected to the network at the same time.

To overcome this problem, *relay* agents are constructed by extending the forwarder agent approach to the notion of store-and-forward migration, as shown in Fig. 7. This notion is similar to the process of transmitting electronic mail by using SMTP. When an agent requests a relay agent on the source host to migrate to its destination, the relay agent makes an effort to transmit the moving agent to the destination through transmitter agents. If the destination is not reachable, the relay agent automatically stores the moving agent in its queue and then periodically tries to transmit the waiting agent to either the destination or a reachable intermediate host as close to the destination as possible. The relay agent to which the moving agent is transferred will repeat the process in the same way until the agent arrives at the destination. When the next host on the route to the destination is disconnected, the moving agent is stored in its current place until the host is reconnected. When a mobile computer is attached to a network, its relay agent multicasts a message to relay agents on other connected computers. After receiving a reply message from the relay agents at the destinations of agents stored in its queue, the relay agent tries to transfer those agents to their destinations.

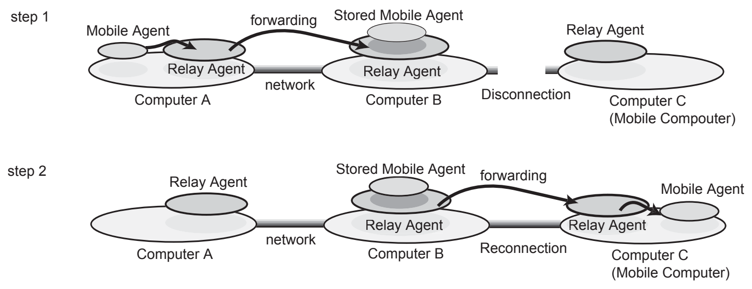

Fig. 7. Relay agent for tolerant network disconnection.

7 Conclusion

This paper described a framework for building a self-configurable infrastructure for agent migration. This framework provides a layered architecture for network protocols for migrating agents and allows these protocols to be naturally implemented within mobile agents. Therefore, network processing for mobile agents can be dynamically added to and removed from remote hosts by migrating corresponding agents according to the requirements of respective visiting agents and changes in the network environment. I developed several mobile agent-based protocols, for example, point-to-point channels among neighboring hosts, and application-specific routing protocols for migrating agents among multiple nodes. A prototype implementation of the framework built on a Java-based mobile agent system called MobileSpaces was carried out. The framework can greatly simplify the development of active network technology [15]. This is because mobile agents are introduced as the only constituent of this framework and thus algorithms and protocols for active networks can be constructed and reused through a single programmable abstraction for composition and refinement of mobile agents.

Finally, I would like to mention some future research directions. The performance of the current implementation is not yet satisfactory and thus further measurements and optimization are needed. I intend to focus on developing other protocols in addition to the examples presented in this paper. Also, my protocols are not always dependent on my framework and thus should be applied to other active network infrastructures.

References

1. Y. Aridor, and D.B. Lange, "Agent Design Patterns: Elements of Agent Application Design", in Proc. Second International Conference on Autonomous Agents (Agents '98), ACM Press, pp. 108-115. 1998.
2. C. Bäumer, and T. Magedanz, "The Grasshopper Mobile Agent Platform Enabling Short-Term Active Broadband Intelligent Network Implementation", in Proc. Working Conference on Active Networks, pp.109–116, LNCS Vol.1653, Springer, 1999.
3. A. Bieszczad, B. Pagurek, and T. White, "Mobile Agents for Network Management", IEEE Communications Surveys, Vol. 1, No. 1, Fourth Quarter 1998.
4. I. Busse, S. Covaci, and A. Leichsenring, Autonomy and Decentralization in Active Networks: A Case Study for Mobile Agents, Proceedings of Working Conference on Active Networks, pp.165–179, LNCS, Vol.1653, Springer, 1999.
5. D. P. Friedman, M. Wand, and C. T. Haynes, "Essentials of Programming Languages", MIT Press, 1992.
6. A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding Code Mobility", IEEE Transactions on Software Engineering, 24(5), 1998.
7. R. S. Gray, "Agent Tcl: A Transportable Agent System", CIKM Workshop on Intelligent Information Agents, 1995.
8. T. Gschwind, M. Feridun, and S. Pleisch, "ADK: Building Mobile Agents for Network and System Management from Reusable Components", in Proc. Symposium on Agent Systems and Applications / Symposium on Mobile Agents (ASA/MA'99), pp.13-21, IEEE Computer Society, 1999.
9. A. Karmouch, "Mobile Software Agents for Telecommunications", IEEE Communication Magazine, vol. 36 no. 7, 1998.
10. B. D. Lange and M. Oshima, "Programming and Deploying Java Mobile Agents with Aglets", Addison-Wesley, 1998.
11. ObjectSpace Inc, "ObjectSpace Voyager Technical Overview", ObjectSpace, Inc. 1997.
12. I. Satoh, "MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System", in Proc. International Conference on Distributed Computing Systems (ICDCS'2000), pp.161-168, IEEE Computer Society, April, 2000.
13. I. Satoh, "MobiDoc: A Framework for Building Mobile Compound Documents from Hierarchical Mobile Agents", in Proc. Symposium on Agent Systems and Applications / Symposium on Mobile Agents (ASA/MA'2000), LNCS Vol.1882, pp.113-125, Springer, 2000.
14. I. Satoh, "Adaptive Protocols for Agent Migration", in Proc. International Conference on Distributed Computing Systems (ICDCS'2001), IEEE Computer Society, pp.711-714, 2001.
15. D. L. Tennenhouse et al., "A Survey of Active Network Research", IEEE Communication Magazine, vol. 35, no. 1, 1997.
16. J. E. White, "Telescript Technology: Mobile Agents", General Magic, 1995.