Dynamic Configuration of Agent Migration Protocols for the Internet

Ichiro Satoh
National Institute of Informatics / Japan Science and Technology Corporation
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
E-mail: ichiro@nii.ac.jp

Abstract

This paper presents a framework for building network protocols for migrating mobile agents over the Internet. The framework allows network protocols for agent migration to be naturally implemented within mobile agents and then dynamically deployed at network hosts by migrating the agents that perform the protocols. It is built on a hierarchical mobile agent system, called MobileSpaces, and several protocols for agent migration are designed and implemented based on standard protocols in the Internet, for example agent migration protocols through plain TCP, HTTP, SMTP or SSL, and application-specific routing protocols for efficiently migrating agents among multiple hosts. This paper describes the framework and its prototype implementation, which uses Java as both the implementation language and the protocol development language.

1 Introduction

Mobile agent technology is promoted as an emerging technology that will make it much easier to design, implement and maintain distributed systems. Although this technology has been expected to be used in the development of various networked applications in the Internet, there have few attempts to apply it to the Internet. The reason for this is a mismatch in network processing, in addition to the problem of protecting against malicious agents or malicious hosts. That is, most existing mobile agent systems are essentially designed to be used within an intranet, instead of on the Internet, and also they are often dependent on particular network protocols such as Java RMI and HTTP. In almost all intranet situations, however, there is a firewall that prevents users from establishing a direct socket connection to/from an external node, except for certain ports, and thus some of the systems cannot migrate any mobile agents through the firewall. Moreover, several applications often require application-specific network processing for migrating their agents over a network. For example, a mobile agent for electronic commerce needs to be transformed into an application-specific encrypted bit stream before transferring itself over a network, including the Internet. However, existing mobile agent systems cannot dynamically adapt their own network processing to the requirements of each visiting agent because their network processing is statically embedded inside them.

The goal of this paper is to present a mechanism for dynamically customizing network processing for agent migration in the Internet. We describe a framework for dynamically deploying and changing network protocols for agent migration according to the requirements of each visiting agent and the external environment. This framework introduces the notion of network processing of mobile agents, by mobile agents, for mobile agents. The notion allows network processing for mobile agents to be implemented as a set of mobile agents. That is, our mobile agent-based protocols can transmit mobile agents as first-class objects to their destinations. Also, the dynamic deployment of the mobile agent-based protocols can be naturally and easily performed by the migration of the agents that support these protocols. Therefore, our framework allows network processing for mobile agents to be adapted to the requirements of visiting agents and to changes in the environment.

This paper is organized as follows. Section 2 explains our framework of customizable network processing for agent migration. Section 3 briefly reviews our mobile agent system, *MobileSpaces*, and Section 4 presents several mobile agent-based protocols running on the system. Section 5 shows the usability of our framework based on two real-world examples and Section 6 surveys related work. Section 7 describes the current status of the framework and Section 8 provides a summary and then touches on future issues.

2 Architecture

The framework presented in this paper provides a self-configuring infrastructure for mobile agents. It can deploy and configure network protocols for agent migration according to the requirements of visiting agents and to

changes in the network environment. This section outlines the overall architecture of the framework and describes the basic idea of adaptive protocols based on the framework.

2.1 Configurable Network Processing

Our framework is built on a mobile agent system, called MobileSpaces, presented in our previous paper [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-like structure. The latter means that each mobile agent can migrate to other mobile agents as a whole, with all its inner agents. Each agent can freely move into any agent in the same agent hierarchy except into itself or its inner agents, as long as the destination agent accepts it. A container agent is responsible for automatically offering its own services and resources to its inner agents and can subordinate its inner agents. Therefore, an agent can directly instruct its inner agents to move to another location.

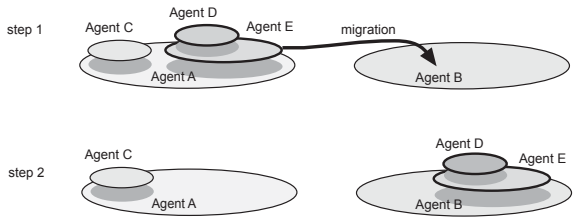

Figure 1. Agent hierarchy and inter-agent migration.

Our mobile agents can transmit other mobile agents as first-class objects [3], in the sense that mobile agents can be passed to and returned from other mobile agents as values by their container agents. Such container agents are still mobile and thus can transport their inner agents to other agents. As a result, our framework allows various operations for mobile agents, including network processing for agent migration, to be naturally constructed and performed by other mobile agents. That is, network protocols for agent migration are implemented within mobile agents. Accordingly, mobile agents are introduced as the only constituent of our network processing for mobile agents. Moreover, such protocols can be dynamically and easily changed by migrating agents that implement the protocols to destinations and immediate nodes. Therefore, our framework offers a self-configurable infrastructure of network processing for mobile agents. Furthermore, they can be constructed and reused through a single programmable abstraction for composition and refinement of mobile agents. As a result, our framework can greatly simplify the development of active networks.

2.2 Agent Migration Protocols for the Internet

Current 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. Our hierarchical structure for mobile agents enables network protocols for agent migration to be organized hierarchically. For example, network processing agents at the bottom layer of the source host transmit a mobile agent to the same kind of network processing agents running at its destination host through their favorite data transmission protocols. Encryption and authentication mechanisms for agent migration are constructed at a layer independent of network processing running at other layers. Also, agent migration in a hierarchy is introduced as a basic mechanism for accessing services provided by the underlying layer. When an agent wants a service, it can access the service by migrating itself into one of the agents providing that service.

3 MobileSpaces: A Runtime System for Hierarchical Mobile Agents

Here we briefly review the MobileSpaces system[1], which provides an infrastructure for building and executing mobile agents for network processing in addition to mobile agent based-applications. The MobileSpaces system supports mobile agents that obey the notions of agent hierarchy and inter-agent migration presented in the previous section.

Our mobile agent system can dynamically adapt itself to changes in the network environment because it is constructed based on a micro-kernel architecture, like several operating systems. That is, the system consists of two parts: a core system and subcomponents. The former offers only the minimal and common functions, independent of the underlying environment. It is thus independent of the network infrastructure.[2] The latter is a collection of subcomponents outside the core system that provide other functions, for example, agent migration between different computers and persistence of secondary storage, which may depend on the surrounding environment. These subcomponents, including network processing for agent migration, are implemented as mobile agents so that they can be dynamically added to and removed from the system by migrating and replacing the corresponding agents.

[1] Details of the MobileSpaces mobile agent system can be found in our previous paper [12].

[2] The current implementation of the core system has a built-in mechanism for transmitting agents over the network by using an extension of the HTTP protocol running on TCP/IP.

3.1 Core System

Each core system runs on a computer and is made as small as possible. It offers only three facilities.

Agent Hierarchy Management: Each runtime system has an agent hierarchy, which is maintained as a tree-like structure in which each node contains a mobile agent and its attributes, and corresponds to the root node of its own agent hierarchy. Agent migration in an agent hierarchy is performed simply as a transformation of the tree structure of the hierarchy. A container agent is introduced as a service provider for its inner agents. Each agent offers a collection of service methods that can be accessed by its inner agents. Each agent is active but subordinate to its container agent. That is, a container agent can instruct its inner agents to move to other agents or computers, and it can marshal and terminate them.

Agent Execution Management: Each mobile agent can have more than one active thread under the control of the core system. The core system maintains the life-cycle state of mobile agents. When the life-cycle state of an agent is changed, for example creation, termination, or migration, the core system issues events to invoke certain methods in the agent and its containing agents. The core system can explicitly limit the length of an agent's visit and the number of staying agents. When the time limit of a staying agent expires, it can automatically terminate the agent. This limitation offers a mechanism for caching network protocols.

Agent Verification Management: The current implementation of the system uses the Java object serialization package for marshaling the states of agents, so agents are transmitted based on the notion of weak mobility [4]. 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. Enriched security mechanisms for agent mechanisms can be implemented by mobile agents outside the core system. Also, the core system provides a facility for marshaling agents into bit streams and unmarshaling them later.

3.2 Mobile Agent Program

Our mobile agents are programmable entities like other mobile agents. Each agent consists of three parts: a body program, context objects, and inner agents. The body program is an instance of a subclass of abstract class `Agent`.[3] 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 [9]. 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 as the argument of the command. An inner agent cannot access any methods defined in its container agent. Instead, each container is equipped with a context object that offers service methods in a subclass of the `Context` class, like the `AppletContext` class of Java's Applets. These methods can be indirectly accessed by its inner agents to get information about and interact with the environment, such as with their container, their sibling agents, and the underlying computer system. Each inner agent can invoke the public methods defined in the context of its container via several built-in application programming interfaces.

Each agent is associated with a resource limit that functions as a generalized Time-To-Live field. This limit is carried with the agent and is decremented by nodes as resources are consumed when the agent arrives at a new place. Nodes can discard agents when their limit reaches zero. To restrict the total resource bounds, when one agent creates another inside the network, the resources allocated to each created agent must be less than those of the creating agent.

[3] Examples of mobile agent programs are given in the Appendix.

4 Agent Migration Protocols in the Internet

Since the MobileSpaces core system supports only functions independent of the underlying environment, other functions, including agent migration between different computers, must be provided by mobile agents outside the core system. The system introduces each mobile agent as a service provider, because it is designed to provide service to its inner agents. When a mobile agent is preparing for a trip over a network, the agent migrates itself into a mobile agent that provides appropriate network processing in the same agent hierarchy and then the agent automatically transfers the visiting agent (or migrates itself) to its destination, or delegates other mobile agents in the same agent hierarchy.

4.1 Point-To-Point Channels for Agent Migration

Our framework enables point-to-point agent migration to be provided by mobile agents, called *transmitters*, instead of by the core system. Transmitter agents correspond to a data-link layer or a network layer and are responsible for establishing point-to-point channels for agent migration between the source host and destination host through a (single hop or multiple hops) data transmission infrastructure, such as TCP/IP, as shown in Figure 2. They abstract away the variety in the underlying network infrastructure and exchange their inner agents with coexisting agents running

at remote computers through their favorite communication protocols. Furthermore, transmitter agents are implemented as mobile agents so that they can be dynamically added to and removed from the system by migrating and replacing the corresponding agents, to keep up with the changes in the network environment. After an agent arrives at a transmitter agent from the upper layer, the arriving agent indicates its final destination. The transmitter suspends the arriving agent (including its inner agents), then requests the core system to serialize the state and code of the arriving agent. Next, it sends the serialized agent to a coexisting transmitter agent located at the destination. The transmitter agent at the destination receives the data and then reconstructs the agent (including its inner agents) and migrates it to the destination or to specified agents for offering upper-layer protocols.

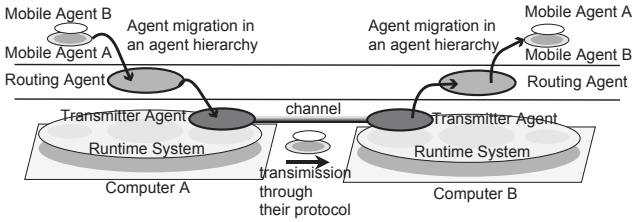

Figure 2. Transmitter mobile agents for establishing channels between nodes.

Since each runtime system can be equipped with more than one transmitter agent, upper-layer protocols can dynamically select a suitable agent in their requirements and migrate their inner agents to the selected transmitter agents. We have already implemented several transmitter agents based on data communication protocols widely used in the Internet, such as TCP, HTTP, and SMTP. Authentication services normally available in secure communications infrastructure include this functionality. Therefore, we implemented transmitter agents, which can exchange agents with each other through Secure Socket Layer (SSL), one of the most popular secure communication protocols in the Internet. We provide a virtual class in Java that can be specialized to create transmitter agents for various protocols. Therefore, we can easily implement point-to-point channels based on other secure communication protocols for data transmission.

4.2 Application-Specific Routing for Agent Migration

A mobile agent often must visit multiple hosts to perform its task and thus is required to make an application-specific and network-dependent itinerary. On the other hand, channels between transmitter agents support point-to-point agent migration. Therefore, we need mechanisms to migrate an application-specific mobile agent among multiple hosts so it can perform its tasks. However, it is difficult to determine the itinerary at the time the agent is designed or instantiated. In addition, even if an agent was optimized for a particular network, it might not be reused in another one. Therefore, we introduce two approaches for determining and managing the itinerary of agents, which are built on transmitter agents running on hosts.

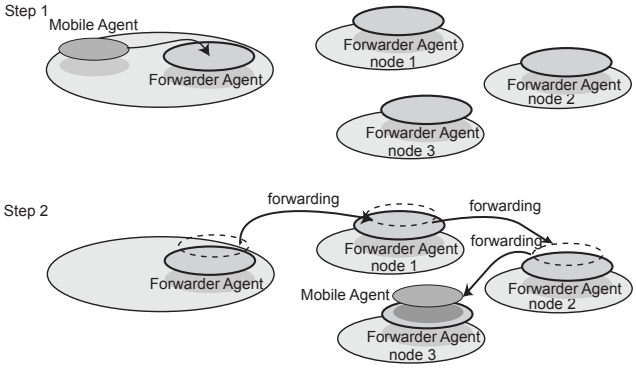

Figure 3. Routing agents for forwarding another agent to the next nodes.

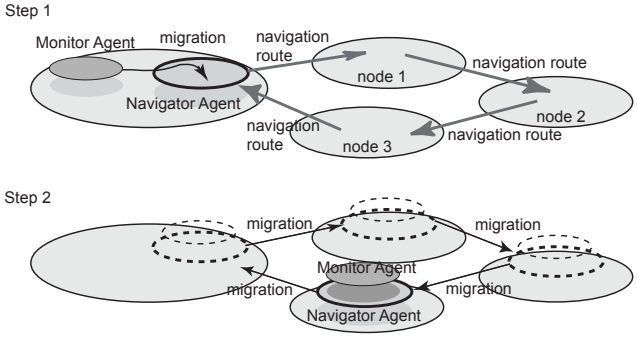

Figure 4. Navigator agent for traveling among nodes with its inner agent.

- The first approach provides a function similar to that of routers. We introduce a service provider, called a *forwarder* agent, for redirecting moving agents to new destinations. Each forwarder agent stays at a host. When receiving agents, it redirects the agents to their destinations through point-to-point channels established among multiple nodes as shown in Figure 3. Each forwarder agent holds a table describing part of

the structure of the network and can be dynamically deployed at nodes and coordinate with other forwarder agents to redirect moving agents to their destinations. However, if the destinations are not reachable, it tries to transfer the agents to another forwarder agents running on intermediate nodes as near the destinations as possible. Each forwarder agent will repeat the entire process in the same way until the agent arrives at the destination.

- 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 node to another under their own control, just as active packets can define their own routing. We propose a service provider, called a *navigator*, for conveying inner agents over a network, as shown in Figure 4. Each navigator agent is a container of other agents and travels with them in accordance with a list of nodes 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. Each navigator has a routing mechanism for managing a routing table consisting of nodes the navigator agent needs to visit. It maintains a list of nodes to be visited and provides methods to dynamically add and remove elements from this list.

The interaction between a forwarder or navigator agent and its inner agents is based on event-based communication. Upon receiving agents, a forwarder propagates certain events to its visiting agents instructing them to do something during a given time period. After the events have been processed by the inner agents, the forwarder navigator transmits another forwarder agent running at the next host. Upon arriving at a place, a navigator propagates certain events to its inner agents. After the events have been processed by the inner agents, the forwarder or navigator continues on its itinerary. Since the two approaches can hide the description of an agent's itinerary from its behavior, mobile agents become independent of the structure of networks and the modularity and reusability of application-specific mobile agents are promoted.

4.3 Configuration of Agent Migration Protocol

It is often argued that the advantage of agent migration lies in the reduction of communication costs in distributed computing settings. Although this argument is understandable, our framework can make use of agent migration as a unified mechanism for accessing, changing, or deploying network services for mobile agents. That is, when an agent wants a service, it can access the service by migrating itself to the agent which provides the service. Agent migration in the agent hierarchy is accomplished with the `go()` command and each mobile is referenced in the form of uniform resource locators (URLs). Our URL consists of two parts: scheme and address. The former is used as the reference of a service provider agent rather than the protocol name. The latter specifies a hierarchical mobile agent located at the same agent hierarchy or different computers. Suppose that an agent performs `go(new AgentURL( "TCP-TRANSMITTER: //some.where.com/ agent1/agent2"))` where `TCP-TRANSMITTER` denotes a transmitter agent which migrates its inner agents at TCP-based communication. When an agent performs the above command, the agent enters the transmitter agent to request that it be migrated into an agent which is a child agent, named `agent2`, of the `agent1` agent included in the base agent running on the host addressed as `some.where.com`.

Our framework can simultaneously support a variety of network protocols given by different mobile agents. Hence, a mobile agent can be provided its required service by one of the most suitable agents which can deliver the service in the current execution environment. Therefore, agent migration can be viewed as a meta mechanism for changing network protocols. Usually, each mobile agent itself selects one of the service provider agents. To find an appropriate service provider for network processing, an agent trying to move to other nodes is required to know the environmental information exactly, but it is difficult for a mobile agent to access the environments. Therefore, we introduce a simple and practical mechanism to detect suitable service providers by using environment variables maintained by the operating system and external programs.

The syntax of our URLs can contain the form `$(variable)`, where `variable` denotes a variable name whose value is a string. These variables are basically maintained in agents. Also, they can be associated with environment variables provided by the shell program or the operating system. The current implementation of our system assumes that the operating system and external programs can reflect environmental changes in the values of their environment variables. For example, suppose `go(new AgentURL( "$(TRANSMITTER): //some.where.com/ agent1/agent2))` where `$(TRANSMITTER)` is a variable for specifying a transmitter agent recommended by the operating system or external programs. For When the value of the variable is `/HTTP-TRANSMITTER`, which specifies a TCP-based transmitter agent located at the root of the agent hierarchy, the above URL is interpreted as `/HTTP-TRANSMITTER:` //some.where.com `/agent1/agent2`. When the network environment is changed, the operating system or external programs can detect the change and update the value

of the TRANSMITTER environment variable so that the value refers to one of the most suitable agents in the current network environment.

4.4 Protocol Distribution

Given a dynamic network infrastructure, a mechanism is needed for propagating mobile agents for supporting protocols to where they are needed. The current implementation of our framework provides the following three mechanisms: (1) mobile agent-based protocols autonomously migrate to nodes at which the protocols may be needed and remain there in a decentralized manner; (2) mobile agent-based protocols are passively deployed at nodes 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 nodes that the agents traverse. This mechanism can improve performance in the expected 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 the runtime system and thus can be customized easily.

Our framework basically uses a scheme in which agents for processing network protocols are downloaded before they are needed, or simultaneously migrated with the agents that need to be processed by the protocols. However, in an effort to increase flexibility, we offer an on-demand approach: such agents can be downloaded on demand and cached for future use, as in several existing active networks. Mobile agent-based protocols identify their types and the protocols to which they belong as they travel. When a mobile agent arrives at a node, the cache of agents is checked. If the required protocol agent is not present, a request to fetch the missing agents is sent to the node from which the agent arrived. The transmission of the arriving agent is suspended, awaiting the protocol, for a finite time. The deployment of transmitter agents needs to be performed by other transmitter agents. For practical reasons, our current implementation provides a built-in transmitter agent, which can deploy other agents to specified nodes through extended HTTP running on TCP/IP communication. It offers the bootstrapping capability needed to install other protocols.

5 Examples

This section presents some practical examples of the framework presented in this paper that demonstrate how it can be exploited.

5.1 Authentication for Agent Migration

Before migrating an agent to its destination host, the moving agent, the source host, and the destination host should to be authenticated. Our current implementation provides a Kerberos-based authentication mechanism for agent migration. Kerberos [18] provides secure authentication services, provided the Kerberos server itself is trusted. It authenticates users without exposing their passwords on the network and generates secret encryption keys that can be selectively shared between mutually suspicious parties. It also allows roaming mobile agents to authenticate themselves in foreign domains where they are unknown, thus enhancing the scale of mobility. Our methods have also been devised to use Kerberos for authorization control and accounting before establishing a TCP connection to transmit mobile agents from the source host to the destination host. This system consists of authentication servers and a ticket-granting server. Each authentication server verifies users during login, and the ticket-granting server issues proof of identifier tickets. We assume that transmitter agents are allocated at the source host and the destination host. A transmitter agent at the source host requests a session key to an authentication server and the ticket-granting server. After receiving the key from the ticket-granting server, the transmitter agent migrates its inner agents to the destination. The current implementation is susceptible to off-line password guessing attacks and to replay attacks for a limited time window.

5.2 Monitoring Network Nodes

A typical application of mobile agents is as a monitoring system for network management, and a discussion of the suitability of mobile agents in network management can be found in [2, 8]. Using navigator agents presented in the previous section, we can easily construct a system for monitoring a set of equipment located at nodes in a network and reacting to certain behavioral patterns.

A monitoring agent collects the network traffic load by accessing SNMP data from the MIB. However, it has no mechanism for its own itinerary and thus is not dependent on a particular network. On the other hand, a navigator agent is responsible for periodically traveling among nodes in a network. It can be designed for navigating in a particular network, and it can guide monitoring agents inside itself through its itinerary over the network.

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 nodes in the network. Next, it migrates itself and the monitoring agent to the nodes sequentially. When it arrives at each destination, it dispatches certain events to its inner

agents at specified timings. A navigator agent can handle exceptions such as inactive hosts on behalf of monitoring agents while trying to migrate itself and its inner agents to new destinations. When the agent has to travel over a network more than one time, it can refer the result of its previous itinerary such as reachable nodes and arrival timings in the next itinerary.

5.3 Locating Mobile Agents

When an agent wants to interact with another agent, it must know the current location of the target agent. Therefore, we need a mechanism for tracking a moving agent. An extension of our forwarder agent offers such a mechanism as shown in Figure 5. 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 in return for the target agent. The forwarder agent then automatically transfers it to the current location of the target agent. Several schemes for efficiently locating mobile agents have been explored in the field of process/object migration in distributed operating systems. Our forwarder agents can easily support most of these schemes because they are programmable entities and can flexibly negotiate with each other through data transmission protocols such as TCP/IP.

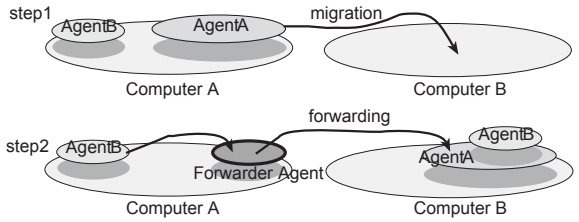

Figure 5. Forwarder agents for locating moving agents.

6 Current Status

Our adaptive protocols have been implemented with the Java language and tested in the MobileSpaces mobile agent system, which is built on the Java virtual machine. The core system is constructed independently of the underlying system and can run on any computer with JDK 1.2-compatible Java runtime.

Even though our implementation was not built for performance, we have conducted a basic experiment on agent migration. The cost of an agent migration in an agent hierarchy was measured to be 5 ms, including the cost to check whether the visiting agent was permitted to enter the destination agent or not. The cost of agent migration supported by transmitter agents allocated on two computers was measured to be 25 ms. A transmitter agent can communicate with another one by using an application-level protocol for agent transmission whose mechanism is modeled on that of the HTTP protocol over TCP/IP communication. On the sender side, a transmitter agent serializes and transfers the codes and state of an agent (including its inner agents) to the transmitter on the receiver side and waits for an acknowledgment message. The second result is the sum of the marshaling, compression, opening TCP connection, transmission, acknowledgment, decompression, security and consistency verifications, and unmarshaling. The moving agent is a simple navigator agent and consists of basic callback methods and contains two child agents. Its data size is about 3 Kbytes (zip-compressed).

7 Related Work

Mobile agents are active programs, which can travel between locations, i.e., agent platforms or runtime systems running on different computers. Many mobile agent systems have been released over the last few years, for example, Aglets [9], Mole [17], Telescript [20], and Voyager [11]. To our knowledge, none can extend and adapt their functions to the requirements of their visiting agents and changes in the environment while running, whereas ours can. Although mobile agents need to be used in heterogeneous environments, for example, mobile computers, information appliances, and wireless networks, existing systems explicitly and implicitly assume a particular network infrastructure. To overcome this problem, the agent migration mechanism of Aglets is implemented based on a hierarchical approach and thus the mechanism itself can be independent of the underlying network infrastructure, but cannot dynamically adapt to changes in the network environments, unlike ours. Among them, the MobileSpaces system presented in our previous [12] can dynamically adapt itself to changes in its execution environment. However, the system itself is just an infrastructure and thus does not support any agent migration protocols for the Internet. Moreover, we proposed a layered architecture for building mobile agent-based protocols for migrating mobile agents in previous studies [14, 15]. That architecture serves as the basis for the framework presented in this paper, but its target is to offer application-specific routing mechanisms, instead of any protocols for transmitting agents over the Internet. On the other hand, this paper proposes a self-configurable architecture for building and deploying agent migration protocols for the Internet.

The framework presented in this paper changes network processing by incorporating it with active network technology [19]. There have been many attempts to apply mobile agent technology to the development of active networks [1, 8], since mobile agents can be regarded as a special case of mobile code technology, which is the basis of most existing active network technologies. In contrast, we apply active network technology to mobile agent technology. Moreover, there have been many reported attempts to customize processing in the literature of meta-level and self-reflective architecture, instead of mobile agent technology. However, their customization mechanisms are often so complex that it is difficult to construct them and make them accessible and secure. We need to construct a simple and natural approach to configuring and adapting network processing for agent migration.

8 Conclusion

We have presented a framework for building a self-configuring infrastructure for agent migration over the Internet. The framework provides a layered architecture for adaptive protocols for mobile agents. These network protocols for agent migration can be naturally implemented within mobile agents and thus can be dynamically added to and removed from the system by migrating the corresponding agents, according to the requirements of visiting agents and changes in the environment. We presented several mobile agent-based protocols based on standard protocols in the Internet. Our prototype implementation built on a Java-based mobile agent system, called MobileSpaces, allowed us to experiment with the construction and deployment of these protocols. This experience strongly suggests that the use of active network technologies in mobile agents holds considerable promise and that our framework can dynamically and flexibly customize network processing for agent migration, without any limitation on the reusability of application-specific agents.

Finally, we would like to mention further issues. Our early performance measurements indicate that the performance of our adaptive protocols is reasonable for a high-level prototype and fast enough for experimenting with application-specific protocols. However, the performance of the current implementation is not yet satisfactory. We plan to improve the performance. We are interested in developing various agent migration protocols for the Internet, in addition to the examples presented in this paper. We also constructed a mobile agent-based approach for building and testing applications for mobile computing [16] and are interested in applying this framework to such a software development methodology for mobile computing. In addition to, our adaptive protocols are not always dependent on our framework and thus should be applied to other active network infrastructure.

References

[1] C. Baumer, and T. Magedanz, “The Grasshopper Mobile Agent Platform Enabling Short-Term Active Broadband Intelligent Network Implementation”, Proceedings of Intenral Working Conference on Active Networks, pp.109–116, LNCS, Vol.1653, Springer, 1999.

[2] A. Bieszczad, B. Pagurek, and T. White, “Mobile Agents for Network Management. IEEE Communications Surveys”, Vol. 1, No. 1, Fourth Quarter 1998.

[3] D. P. Friedman, M. Wand, and C. T. Haynes, “Essentials of Programming Languages”, MIT Press, 1992.

[4] A. Fuggetta, G. P. Picco, and G. Vigna, “Understanding Code Mobility”, IEEE Transactions on Software Engineering, 24(5), 1998.

[5] R. S. Gray, “Agent Tcl: A Transportable Agent System”, CIKM Workshop on Intelligent Information Agents, 1995.

[6] T. Gschwind, M. Feridun, and S. Pleisch, “ADK: Building Mobile Agents for Network and System Management from Resuable Components”, Technical University of Vienna, TUV-1841-99-10, 1999.

[7] C. Hedrick, “Routing Information Protocol”, RFC 1058, June 1988.

[8] A. Karmouch, “Mobile Software Agents for Telecommunications”, IEEE Communication Magazine, vol. 36 no. 7, 1998.

[9] B. D. Lange and M. Oshima, “Programming and Deploying Java Mobile Agents with Aglets”, Addison-Wesley, 1998.

[10] D. S. Milojicic, W. LaForge, and D. Chauhan, “Mobile Objects and Agents (MOA)”, Proceedings of USENIX Conference on Object Oriented Technologies and Systems, April 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”, Proceedings of 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”, Proceedings of 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”, Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS’2001), pp.711-714, IEEE Computer Society, April, 2001.

[15] I. Satoh, “Network Processing of Mobile Agents, by Mobile Agents, for Mobile Agents”, Proceedings of Workshop on Mobile Agents for Telecommunication Applications (MATA’2001), LNCS, Vol.2164, pp.81-92, Springer, August, 2001.

[16] I. Satoh, “Flying Emulator: Rapid Building and Testing of Networked Applications for Mobile Computers”, to appear in Proceedings of Conference on Mobile Agents (MA’2001), LNCS, Springer, December, 2001.

[17] M. Strasser and J. Baumann, and F. Hole, “Mole: A Java Based Mobile Agent System”, Proceedings of ECOOP Workshop on Mobile Objects, 1996.

[18] J. G. Steiner, B. Clifford Neuman, and J.I. Schiller, “Kerberos: An Authentication Service for Open Network Systems”, Proceedings of the Winter 1988 Usenix Conference. pp.191–201, February, 1988.

[19] D. L. Tennenhouse et al., “A Survey of Active Network Research”, IEEE Communication Magazine, vol. 35, no. 1, 1997.

[20] J. E. White, “Telescript Technology: Mobile Agents”, General Magic, 1995.