IEEE Systems, Man, and
Cybernetics Conference (SMC'99)
October, 1999

A Mobile Agent-Based Framework for Active Networks

Ichiro Satoh

Department of Information Sciences, Ochanomizu University
2-1-1 Otsuka Bunkyo-ku Tokyo 112-8610, Japan
E-mail: ichiro@is.ocha.ac.jp

Abstract

This paper presents a framework for active networks. The framework is unique among other existing active networks in being based on mobile agents. It allows protocols to be implemented within mobile agents and dynamically deployed at end systems. Moreover, it can dispatch networked applications, including client/server programs built on TCP/IP, to remote computers. In this paper, we outline an implementation of the framework which is built on a new Java-based mobile agent system designed for active networks. In order to demonstrate the utility of the framework, we illustrate how several networked services to support human activities are dynamically introduced into a network that initially lacks them.

1 Introduction

The innovation of network services to support human activities is unrelenting and new network services are always being wanted and developed. However, the spread of these services has been much slow. This is because these services often need their own network protocols designed for the services, and the current process of changing and deploying network protocols are both lengthy and difficult. The process often requires standardization, which takes some years to be consented. Furthermore, once the new protocols have been accepted, their deployment is difficult, because there is not any automatic mechanism for dispatching programs for new protocols to remote computers. In addition, networked applications designed for these new protocols often have to be deployed at remote computers.

This paper addresses these problems. We present a new active network which allows new network protocols to be automatically deployed and operated at end systems. Our framework is characterized in being based on a mobile agent system, where mobile agents are autonomous programs that can travel from computer to computer in their control. The framework helps us to implement network protocols within mobile agents. By using the mobility of mobile agents, it can transfer not only the code of a protocol program but also its state, including various parameters which must be customized before establishing its communication sessions. Furthermore, since it can deploy networked applications supporting user interface in addition to network protocols at remote computers. Hence, it can provide a practical infrastructure to support new networked services for human activities, such as communication among humans, remote manipulation, and multimedia communication.

We constructed an implementation of the framework. It is built on a new mobile agent system, which is optimized to support active networks. The system is built on the Java virtual machine and mobile agents are written in the Java language. Although the framework is inherently independent of any underlying network, it offers several mechanisms and libraries for constructing application level protocols built on TCP/IP, because TCP/IP is widely used in present day network.

In the next section of this paper, we provide a background on mobile agents and active networks. In Section 3 we present our mobile agent system, called AgentSpace, which allows agents to be migrated to other computers. In Section 4, we present the architecture of our active network framework. In Section 5 we demonstrate two examples of the framework. Finally, conclusions and future work are presented in Section 6.

2 Background

In this section, we provide background on active networks and mobile agents.

2.1 Active Networks

There are two approaches to the realization of active networks (for example see [13]). The programmable packet approach replaces destination addresses in the packets of present day architectures by miniature programs interpreted at nodes on arrival. A typical example of the approach is SwitchWare [3]. It introduces active packets, which can carry programs consisting of both code and data. The approach enables the routing and behavior of each packet to be customized at its own control, but its performance is unfortunately unreasonable.

On the other hand, the active node approach allows new protocols to be dynamically deployed at intermediate and end nodes by using mobile code techniques. Typical examples of the approach are ANTS [14] and NetScript [16]. ANTS is an active network toolkit to support the deployment and cache of network protocols. Each packet has a reference to the forwarding routine to be used to process the packet at each active node. When a routine for processing specific packets does not reside at its arriving node, the routine can be transferred to the node before the packets can be processed. Netscript proposes a language which a dynamic dataflow language for building network software on a programmable network. It is designed for introducing new services for network management and control rather than data transfer purposes.

Most of existing attempts on active networks intend to dispatch miniature programs to remote hosts in order to change the way of communications, and thus cannot deal with any networked applications, for example server programs and client programs including user interfaces.

2.2 Mobile Agents

A mobile agent is an active program that acts on behalf of a user or another program under its own control. That is, an agent can choose when and to where it will migrate itself to another computer and continue its execution in the destination.

As recently as a few years ago, many mobile agent systems have been released. Telescript [15] is the first commercial implementation of the mobile agent paradigm, and AgentTcl [2] is a mobile agent based on an extended Tcl interpreter that executes the Tcl agents. Like ours, most of them have been implemented in the Java language, for example see Aglets [6], MOA [7], Odyssey [1], Voyager [8], and so on.

Mobile agents are very close to active networks. This is because a mobile agent may be regarded as a specific type of an active packet, and an agent platform in traditional networks can be regarded as a specific type of an active node. However, most of existing mobile agent systems are not unfortunately available as a framework for constructing active networks, because they lack any resource management and thus cannot always discover and acquire the resources which an agent needs to accomplish its task. Among them, there are a few attempts to incorporate the mobile agent technology with the active network technology (for example see [4]). For example, TINA [5] and Tempset intend to develop a future network architecture of broadband ATM. The purpose of existing attempts is to apply mobile agents to network management and control. On the other hand, our framework is designed for the deployment of new network protocols (and their applications) for data transfer purposes rather than network management and control.

2.3 Our Active Network Framework

The goal of our framework is to construct a practical infrastructure for active networks, and thus a mechanism is needed for effciently propagating program definitions to where they are needed. Therefore, the framework addresses the dynamic deployment of protocols at a coarse granularity, i.e., per communication session rather than per packet. The framework is also characterized in that it is based on a mobile agent technology, instead of any mobile code technologies. It allows new protocols and networked applications to be constructed within mobile agents. By the migration of these mobile agents, it can dynamically deploy and automatically operate the protocols and the applications at remote computers. Since each mobile agent can autonomously migrate itself to its destination under its own control, a policy of deploying programs for handling protocols is defined in the agents offering the protocols, instead of any underlying systems, including any active nodes.

An active network is often expected to be used in various networks, and thus it should be constructed independently of its underlying network. Therefore, the agent deployment mechanism of our framework can be abstracted from the underlying network and can be dynamically changed and adapted.

In addition, our framework provides a practical mechanism designed for the deployment of application level protocols on TCP/IP, including client/server programs, although it is inherently independent of any underlying protocol. This is because TCP/IP is available in most current computer networks, including the Internet, and various application level protocols on TCP/IP are widely used

and being developed unrelentingly.

3 The AgentSpace System

This section presents our mobile agent system, named AgentSpace, and its mobile agents. The system is implemented on top of the Java virtual machine, without any modifications to it. We chose the language because of allowing agents to be compiled and executed in platform-neutral byte code and the likely emergence of higher performance compilers and runtime systems. Also, the powerful expressiveness of the language, for example object orientation, libraries for communication , multithreading, and dynamic linking/loading allows us to easily construct programs for handling application-specific protocols and their applications within mobile agents.

Agent Program

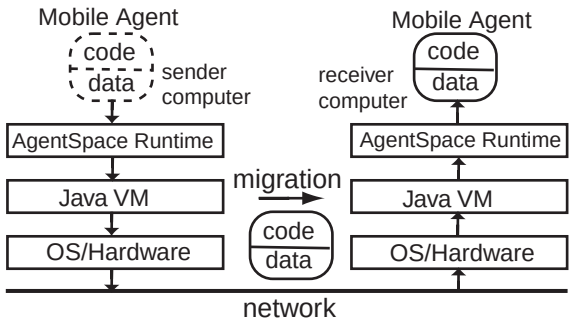

Figure 1: Architecture of AgentSpace

3.1 Mobile Agent Platform

The runtime system is introduced as a platform for mobile agents. It can create/destroy mobile agents, and send/receive mobile agents to/from a runtime system running on another computer. Also, it is characterized in being designed for a distribution mechanism of network protocols and applications.

Agent Migration

The runtime system permits the migration of not only the code but also the values of the instance variables included in the agent. Hence, after migrating the agent, the values are restored in the agent again, and then its execution starts from a given method. The current implementation of the system uses the Java object serialization package in order to marshal and unmarshal agents. The package does not support capturing the stack frames of threads. However, this limitation is not serious in the development of most distributed applications based on mobile agents, as discussed in [12].

Agent Cloning

The runtime system offers a mechanism to create a copy of an existing agent including all instance variables. The cloned agent has the same state as the original agent has, but its identity is different from that of the original one. If the original agent has a reference to resources, the runtime system protects the resources appropriately.

Agent Caching

The runtime system can load the code of each agent on demand and cache it in order to improve performance when particular protocols are sometimes used. The cache is managed in a least used order. When the state of an agent arrives at a remote node, the runtime system on the remote note checks a cache of codes. If the required code is not found at the cache, it sends a load request to the previous node or certain code base nodes. Furthermore, the runtime system ensures that mobile agents can be automatically and dynamically transferred to the nodes which are needed.

3.2 Mobile Agents

Every agent is an instance of a subclass of the base class for mobile agents, called `Agent`. The class defines fundamental callback methods invoked when the life cycles of a mobile agent changes, such as creation, suspension, marshaling, unmarshaling, and destroy, like event delegation event model in Aglets [6]. An agent can migrate itself to the destination specified as `dst` by invoking the `go(AgentURL dst)` method. An agent can create a copy of the agent at the location specified as `url` by means of the `duplicate(AgentURL dst)` method.

```
public abstract class Agent implements Serializable {
  // methods to hook the change of
  // its state in the life-time
  void create(); // after creation
  void destroy();// before termination
  void arrive(); // after accepted a child
  void leave(URL dst); // before dispatched to dst
  void clone();  // before duplication
  void parent(); // after duplication at the original
  void child();  // after duplication at the copy
  ....
  // APIs for Mobile Agents
  void go(URL dst) throws
    NoSuchLocationException ... { ...}
```

```
  URL duplicate(URL dst) throws
    NoSuchLocationException ... {...}
  void setTimeout(int time) // the life-span
  ....
}
```

List 1: `Agent` class

4 Active Network Framework

Our framework allows each protocol and its applications to be implemented in mobile agents and deployed to remote computers through the migration of agents supporting the protocol.

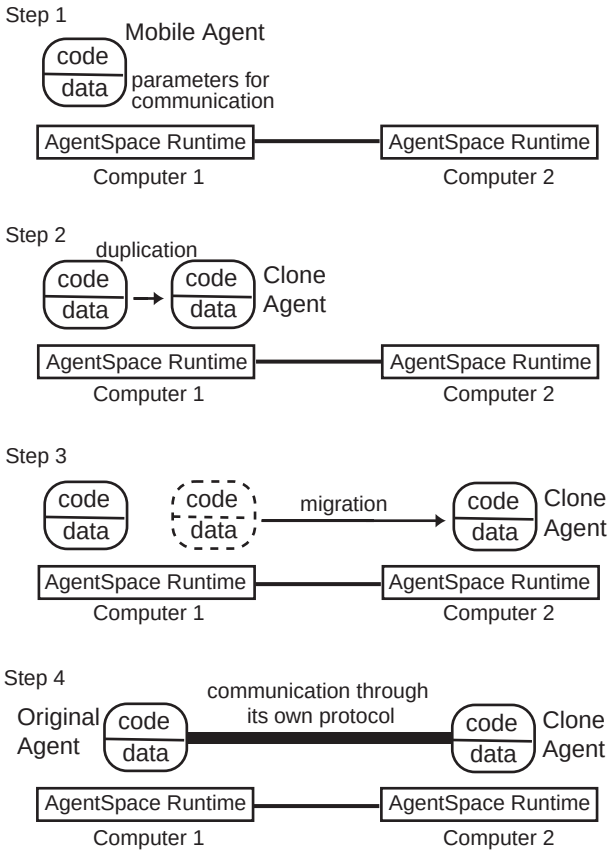

Figure 2: Protocol Deployment

4.1 Active Nodes

Our active network is managed in a decentralized manner and consists of an interconnected group of active nodes that are built on the AgentSpace runtime systems. The runtime system can pull and push the code of an agent and its state separately to remote nodes, and can automatically cache the codes of visiting agents in order to improve performance when particular protocols are often used.

4.2 Protocols

The goal of our framework is to simplify the development and deployment of programs for handling protocols and application programs designed for them. Also, it provides several mechanisms and libraries for constructing network protocols within mobile agents and migrating them to the nodes which must communicate through the protocols.

Next, we present how to deploy a protocol and its applications as follows:

(1) The runtime system loads a mobile agent which implements the protocol and its application.

(2) The user or other agents customize parameters whose values included in the the mobile agent.

(3) The runtime system creates a copy of the customized mobile agent and migrates the original agent and the clone agent to the nodes where the communication session will be established.

(4) When the agents include application programs for the protocol, the programs are autonomously operated and then communicate with each other through the protocol.

Note that new protocols can be dynamically injected into nodes that initially lack them. Also, the new protocols do not have to agree with other nodes, and thus a peculiar protocol can be used without any agreement among other nodes.

Moreover, the original agent and the clone agent can share the same values in their instance variables. This is important and practical. This is because a protocol often contain parameters which must be customized as the common values among entities associated with establishing a communication session, such as buffer size, expiration due to timeout, bandwidth. On the other hand, our framework allows these parameters to be stored as values in the instance variables of mobile agents which implement the protocol and its application. Therefore, the original agent and the clone agent can share the same customized values for the protocol and its applications without any communication between them. Consequently, we can improve the startup performance of a communication session.

This framework can also facilitate short-lived protocols because mobile agents are explicitly imposed on their survival periods. Such agents are automatically destroyed and are discarded from the caches of the nodes after elapsing their survival periods.

4.3 Protocols for Agent Migration

Our framework is unique among other existing mobile agent systems and active networks in that its agent migration can be abstracted from the underlying networks. Instead, our framework introduces special mobile agents, called transmitter agents, that can migrate other mobile agents around networks by means of through their favorite network protocols. We have already implemented several transmitter agents incorporated with TCP, UDP, and IrDA. Transmitter agents are mobile agents and can be dynamically allocated on hosts. Therefore, the runtime system can dynamically add and revise its agent migration mechanisms by migrating transmitter agents implementing a new agent migration protocol to the system. Also, these transmitter agents offer methods for creating agents at remote hosts, tracking the trails of moving agents, and caching agents, in addition to transferring agents.

4.4 Active Networks on TCP/IP

This framework is inherently independent of any underlying networks and protocols. On the other hand, TCP/IP is one of the most popular protocols in current computer networks, including the Internet. A lot of application level protocols on TCP/IP have been wanted and developed unrelentingly. The framework offers practical mechanisms and libraries for constructing application level protocols built on TCP and UDP, in order to simplify the deployment of the protocols and their applications. The libraries consist of builtin classes. For example, `TCPAgent` is such a built-in subclass. It allows us to easily deploy a server program for application level protocols built on the TCP socket and its client programs at remote nodes and automatically communicate between the original program and the clone program through the protocols.

TCP and UDP must to be identified by a port number as well as an IP address. However, a mobile agent may not be able to process their communication through TCP/IP, when the ports which the agent tries to acquire are already assigned to other communications. We introduce the concept of virtual port which is an additional port between socket and TCP port. A virtual port acts as a TCP port from the viewpoint of a socket. When an application binds a socket to a TCP port, the runtime creates a virtual port, and binds the socket to it accordingly. The TCP association is established between two such virtual ports. All code above the virtual port is the same as used in the current socket code such as the Socket class and the SocketServer class provided by the Java language.

5 Examples

In order to demonstrate how we intend our framework to be used, we present two protocols that introduce chat and smart mail services into a network that initially lacks them.[1] We chose these examples because they are typical services to support human activities in the Internet and need to be innovated.

5.1 Chat System

There have been chat systems, which allows people over networks to talk to one another in real-time, for example talk in Unix, IRC, and ICQ. However, there is no standardization for chat protocols that can be agreed among all the interested parties. Consequently, if a user wants to talk his/her friends through a chat protocol, their computers cannot often operate the protocol.

Therefore, we develop a new chat system based on mobile agents. The system consists of mobile agents corresponding to a chat server and clients, including programs for handling its chat protocol. It can dynamically migrate these chat server/client agents to the computers, which initially lacks any chat protocol and its applications but can execute the AgentSpace runtime, according to the deployment process of protocols presented in the previous section. The process is as follows: When the user loads a chat agent through the runtime and then the loaded agent shows a window to input the address of the chat partner. The agent asks the runtime system to create its clone at the same node. Next, the clone agent asks the runtime system to migrate itself to the node specified as the address. After the deployment of them, the original agent and the clone agent can autonomously start to communicate with each other through its own protocol, because they are active programs. Once communication is established, the two users can type simultaneously, with their output displayed in separate regions of the screen.

5.2 File Sharing System

One of typical method of sharing information among remote computers is file sharing, for example Network File System and World Wide Web, which allows to access files on remote hosts. There have been various file sharing protocols are suggested and are used. In order to share files on a remote computer through

[1] Note that we do not intend to present new and better solutions to these protocols.

a file sharing protocol which a client can deal with, a server designed for the protocol must be running in the computer. However, such a file server must be installed on the computer in manual and off-line.

Therefore, we develop a new approach to automatically install and operate a file sharing server at the computers whose files are wanted to be accessed. The approach introduces a file sharing agent which implements a file sharing protocol and its server within a mobile agent. By migrating the agent to a remote computer, we can share files on the computer. The current implementation of the file sharing agent is based on an extended protocol of HTTP server with several operations to write files and to operate file directory.

6 Conclusion

We developed a new framework for constructing active networks. This framework is built on a mobile agent system and thus can transfer not only the code of a program for handling protocols and their applications but also its state, including various parameters which must be customized before establishing each communication session. Therefore, It can dynamically deploy protocols and their applications to remote computers and automatically operate them by the migration of mobile agents to implement them. We believe that our framework can provide a practical infrastructure to support new networked and cybernetic applications for human activities, such as communication among humans, remote manipulation, and multimedia communication. An implementation of the framework and the examples presented in this paper is available from `http://islab.is.ocha.ac.jp/agent/index.html`.

Acknowledgements

We would like to thank to K. Tanahashi for her helping the implementation of the chat system.

References

[1] General Magic, Inc. Introduction to the Odyssey, http://www.genmagic.com/agents, 1997.

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

[3] C. A. Gunter, S. M. Nettles, and J. M. Smith, The SwitchWare Active Network Architecture, IEEE Network, special issue on Active and Programmable Networks, vol. 12, no. 3, 1998.

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

[5] S. Krause, and T. Megadantz, Mobile Serivce Agents Enabling Intelligence on Demand in Telecommunications, Proceedings of Addison-Wesley, 1998.

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

[7] D. S. Milojicic, W. LaForge, and D. Chauhan, Mobile Objects and Agents (MOA), USENIX Conference on Object Oriented Technologies and Systems, April 1998.

[8] ObjectSpace Inc, ObjectSpace Voyager Technical Overview, ObjectSpace, Inc. 1997.

[9] Object Management Group, The Common Object Request Broker: Architecture and Specification, Revision 2.0, OMG formal document 97-02-25, 1997.

[10] I. Satoh, AgentSpace, http://islab.is.ocha.ac.jp/agent/index.html, 1997.

[11] I. Satoh, Hierarchically Structured Mobile Agents and their Migration, to appear in ECOOP Workshop on Mobile Object Systems (MOS'99), June, 1999. also available in http://islab.is.ocha.ac.jp/download/satoh-mos99.pdf.

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

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

[14] D. J. Wetherall, J. V. Guttag, and D. L.Tennenhouse, ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols, in Proceedings of International Conference on Open Architectures and Network Programming, April 1998.

[15] J. E. White, Telescript Technology: Mobile Agents, General Magic, 1995.

[16] Y. Yemini, and S. da Silva, Towards Programmable Networks in Proceedings of FIP/IEEE International Workshop on Distributed Systems, October, 1996.