A Scalable Contents Distribution Service using a Java Agent Application Framework

Kil-Hung Lee[1], Jae-Soo Kim[1], Yong-Hyeon Shin[1]

[1] Dept. of Computer Science and Engineering, Seoul National Univ. of Tech.,
172 Gongnung-2dong, Nowon-gu, Seoul, 139-743, Korea
{Khlee, Jskim, Yshin}@Snut.ac.kr

Abstract. This paper presents a Java agent application framework and shows its example through the implementation of an agent application service. This framework comprised of agent, agent system, agent master and agent manager components. Each component is connected to others via an agent interface, and messages are passed through these interface points. By using this framework, it is possible to deploy a new agent application service with simple and efficient operation. Herein, we implemented the agent application service network using a Java agent application framework. Specifically, the implemented network was a contents distribution network using an agent service, and the network was both controlled and managed with the agent framework interface. In our test, the network service decreased network traffic while preserving reasonable quality through the adequate deployment and management of agent components.

1 Introduction

With the development of the Internet, many new and hitherto unimagined application services have been introduced through this revolutionary medium. To facilitate the actual use and embodiment of such services, innovative new approaches have been studied and many are now in the process of development. As an example, the peer-to-peer (P2P) method has strong merits given its scalability, and has the potential to serve as an applicable means of tackling errors more easily. Or consider distributed computing technology through the grid framework, which may serve to bring us closer to the time when these exciting application services can be used. Likewise, mobile agent technologies that can conduct critical operations at precise locations are of significant importance in the overall software technology and development progress [1].

An agent is an autonomous programming object that can perform by itself those functions that have been entrusted to it by the software user. This operational method is an innovative new way of implementing the traditional distributed computing system and constitutes a new computing paradigm. An agent can be characterized as autonomous, responsive to given conditions, and self-mobile with cooperative and learning properties. Through proper utilization of the functions of an agent, which can be used to operate in specific places at precise times, we can achieve reductions in network traffic and contribute to increased operational efficiencies.

These agent technologies may be applicable to e-commerce and information retrieval technology, to mention just two potential services: *Applications to communication infrastructures* and *network management services* [2]-[4]. In [5], the agent technology was exploited to effectively monitor a large-scale network. Having reproduced itself and allocated various management functions by deployed distribution of these reproduced agents in the network, they reduced management traffic, shortened the response time, and provided the context wherein agents can safely operate even in periods of malfunction. In [6], a mobile agent was used to gather data through the Internet. After transferring to each server where the data is located, agents accumulate and collect the information gathered from their each visit. Such methods showed significant reductions in data traffic as well as much improvement in reducing response time when planned efficiently [7].

In this paper, a JAAF (Java Agent Application Framework) has been implemented to deploy the application service more effectively by using agents. The means to better define what agents are and their effective control and management, and new services utilizing these functionalities will be discussed. In Section 2, the standards and deployment status of the agent system will be examined. In Section 3, the agent application framework will be discussed. Thereafter, the implementation of a distribution network service using agents will be presented and the implemented service property further elaborated upon in Section 4. This paper will be closed with final remarks in Section 5.

2 Status of the Agent System

Since the early 1990s, software agents – also known as intelligent agents – have been the subject of a great deal of speculation and marketing hype. Agents can be classified into two major categories: *resident* and *mobile*. Resident agents are software agents that stay in the computer or system and perform their tasks there. For instance, many of the wizards in software programs are designed to carry out a very specific task while we are using the computer. Mobile agents are also software agents that can transport themselves across different system architectures and platforms.

The agency's core is the agent platform, a component model infrastructure that provides local services such as agent management, security, communication, persistence and naming. In the case of mobile agents, the agent platform also provides agent transport. Most agent systems provide additional services in the form of specialized agents that reside in some – possibly remote – agency. Some agent systems also include standard service agents, such as a broker, auctioneer, or community maker. These service agents augment the basic agent infrastructure. The agent platform and additional service agents can monitor and control message exchanges to detect any violation of the rules of engagement [8].

2.1 Agent Standards

As various kinds of agent technologies have been developed, OMG (Object Management Group) set up the standard for the MAF (Mobile Agent Facility) mobile technology [9]. This standard is based on the outer interface of CORBA (Common Object Request Broker Architecture) with regards to the mobile agent system. The agent standards for by OMG or FIPA (Foundation for Intelligent and Physical Agent) have been emphasizing the communication between agents and agent management as it relates to their creation and deletion. Developing ways to implement the necessary system needed for compatibility within the application environment, at the same time observing such standards, may take various forms depending on available options.

MAF is basically comprised of agents, place, agent system, and communication infrastructure. The agent system is a platform that supports the composition, interpretation, operation, and movement of agents. The place, which identifies logical location, is defined within this agent system and agents are located in specific places while operating the service by moving from place to place. The locations of agents include the place where agents are located and the address of the agent system.

A communication infrastructure is the basis for providing communication to agent systems, and provision of services such as RPC (Remote Procedure Call), naming services, and security are possible. However, since the naming service of CORBA does not need to be operated separately in order to support mobile agents, it alternately provides a naming service called MAFFinder for MAF exclusively and sends that interface to COBRA instead.

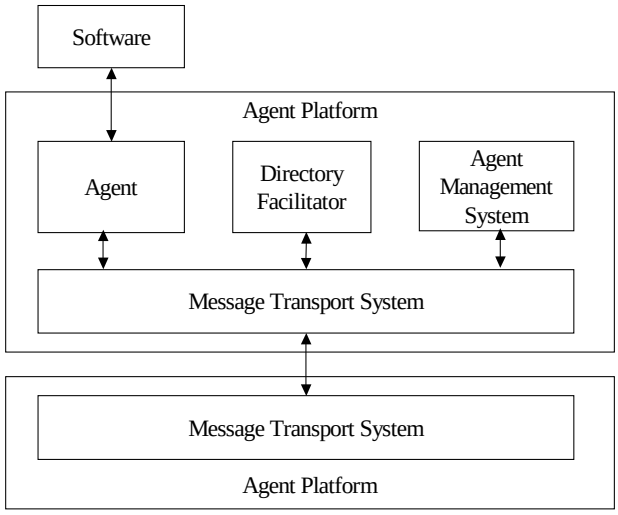

Fig. 1. Agent Management Reference Model

The purpose in setting up the FIPA standard was to help operations among applied agents by defining a common communication standard among agents in various com-

puting environments. FIPA has been preparing five standards such as application, abstract structure, agent management, agent communication and agent message transmission. The agent management, stipulated in the FIPA, is concerned with the life cycle management of agents such as creation, registration, designating location, movement, or termination [10].

As shown in Fig. 1, FIPA agent management reference model consists of a DF (Directory Facilitator), AMS (Agent Management System), ACC (Agent Communication Channel) or MTS (Message Transport System). The DF provides agents with a 'Yellow Page' function. The AMS and ACC back up the communications among agents. The DF, AMS, and ACC form the group that defines functions and can be operated by individual agents or by all of the different agents.

2.2 Examples of Java Agent Development Systems

Although mobile agents can be developed with virtually any kind of programming, the development of a platform running on an independent language to serve as the interpreter when considering different operation environments. Developers often use distributed objects, active objects, and scriptable components to implement agents. To build agents, Java, Java-based component technologies, XML, and HTTP have been used. These technologies are simple to use, ubiquitous, heterogeneous, and platform-independent. Among such languages, Java is particularly ideal for use in developing mobile agents. Some currently available toolkits using the Java mobile agent system include IBM's Aglet, JADE (Java Agent DEvelopment Framework), and FIPA-OS (Open Source), to name a few.

Aglet, developed by IBM's Japan laboratory, is a Java-based workbench and is a system that combines agents and the applet [11]. The development of agent applications using the Aglet class provides a convenient means of mobile agent implementation while preserving a significant portion of necessary characteristics and functions. These functions include an unique name assigning function for the agent, an asynchronous and synchronous communication method among agents, an agent class loader that can bring the Java byte code of agent through the network, and an agent context that can obtain environmental information independently without regard for the agents' operating system environment.

JADE is a framework implemented entirely using the Java language [12]. The goal of JADE is to simplify the development of multi-agent systems while ensuring standard compliance through a comprehensive set of system services and agents in compliance with the FIPA specifications: naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used. The JADE Agent Platform complies with FIPA specifications and includes all those mandatory components that manage the platform, that is the ACC, the AMS, and the DF. All agent communication is performed through message passing, where FIPA ACL (Agent Communication Language) is the language to represent messages.

FIPA-OS is a component-based toolkit that makes it possible for fast development of agents while observing the FIPA standards [13]. FIPA-OS supports most standard specifications, and is continuously being improved upon through the management of

an open source community project, and can also be regarded as the ideal choice for easy and fast development of agents that follow the FIPA specification. The Agent City Network consists of set of software systems (platforms) connected to the public Internet [14]. Each of these platforms hosts agent systems capable of communicating with the outside world using standard communication mechanisms. Each agent is in turn able to provide zero or more services which can be requested and accessed by other agents in the network. The network is completely open and anybody wishing to deploy a platform, agents or services is welcome to do so. All technologies used are based on consensual standards such as the FIPA standard or the emerging Semantic Web Ontology standard.

3 JAAF (Java Agent Application Framework)

Java Agent Application Framework is an agent development framework developed using Java language and consists of several components. Each component is a package of classes that has a role in agent application environment. Messages are exchanged between these components through the agent interface.

3.1 JAAF Components

The agent application framework component is composed of the agent, agent system, agent master system and agent manager. Fig. 2 shows the agent components.

Agent: An agent is a software component and a small program that is designed to fulfill certain service operations. The agent is a class implementing a service interface in Java, which defines the method necessary for both the operation and control of the service. The agent is sent to the agent system and performs certain tasks while also responding to various events. Agents can be divided into two types depending on the ways they are to be operated in JAAF: *RMIAgent* and *MSGAgent*. An *RMIAgent* is sent to its target place to conduct necessary operations and called its service function by the client. Fast deployment of services and the small one-time operation is easily implemented by using the *RMIAgent* service. A *MSGAgent* communicates with the client through the exchange of agent messages. The *MSGAgent* has the advantage of continuing one specific operation and sends messages when they are necessary without having to maintain the connection.

Agent System: An agent system provides a place for the agent and controls the agent service. Agents are sent to the agent system through the *Service* message. The class loader in the agent system is in charge of loading the agent and can be operated through the following steps. First, confirm whether the *Service* message includes the code. Then, immediately create the object from the code and start the service. If the message does not have a service code, load the class from the user using the additional information in the *Service* message, that is, the code server *URL*. After receiving the service code, the agent system creates the object and provides the operation environment using services such as start, stop, restart, and termination. During the

agent operation, the agent system delivers the receipt and transmission of messages and reports the agent state to the agent master.

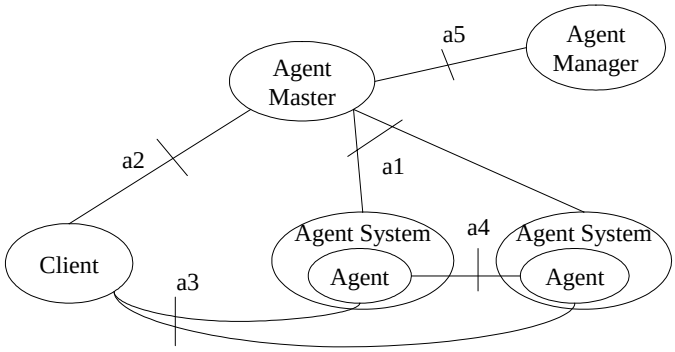

Fig. 2. Components and Interfaces of Java Agent Application Framework

Agent Master: An agent master controls the local agent system, intercedes with clients who are service users for the agent's service provided, and provides management interface to the agent manager. It connects between the client and the agent provider domain. It has available lists of the agent system and maintains the location and the status of the agent. It delivers the *Service* message to the appropriate agent system and returns the *Status* messages to the client. And, it relays the agent's *Inform* messages for cooperative operations between agents. The registry is the RMI (Remote Method Invocation) functional components, which is being used to expose the *RMIAgent* by itself to the client. The client can access the registry through the agent master and request the necessary service by obtaining the agent pointer. The registry is implemented as an internal function of the agent master.

Agent Manager: An agent manager can manage many agents with fewer complications by managing the agent master rather than directly controlling the agents. The agent master embodies the agent management information that is necessary to manage how agents behave. The agent manager can carry out monitoring activities by reading an MIB (Management Information Base) and service management functions by creating needed values. This scalable solution simplifies the agent management function efficiently.

Client: A client requests the service and gets the result and can offer the service codes through the agent master. The client can designate the agent system directly or help the agent master choose the appropriate agent to carry out the service. In the case of P2P applications, the agent system and the client can be combined and operated together.

3.2 JAAF Interfaces

In order to have effective control and management of the agent, we first need to define the interface. The definition of the agent interface is composed of the definitions

of the exchanged messages. The standards of MAF and FIPA are reflected in the definition of the interface and each interface is represented in Fig. 2. The agent master performs the DF function for FIPA and the agent system carries out the MAF and AMS functions for FIPA. Messages are defined to perform functions similar to those of standard messages for FIPA.

a1: This interface refers to an interface between the agent master and the agent system. A *Register* message is sent from the agent system to the agent master to signal the commencement of services. This message includes the parameter values such as the agent's system identifiers for the agent system registration and place. An *Unregister* message is sent from the agent system to inform the end of the agent service. The abnormal ending process can be detected when the agent system is not responding in the form of a *Status* message after receiving the message from the agent master.

a2: This interface refers to an interface between the client and the agent master. If a client sends a *Lookup* message together with the information necessary for the service request, the agent master resends the *Status* message with the agent's system information suitable for the user's demand. The agent service request, as demanded by a client, is initiated by the *Service* message. The *Service* message includes the service name, program information, and the agent system information to be operated. A client can check the condition of the service through the *Status* message.

a3: This interface refers to an interface between an agent and a client. Through this interface, service conditions and results can be exchanged via *Report* message. Usually, agents notify the service status information through the agent master. If the agent system accepts the command of the client via the *Command* message, the values of current and final results are delivered to the client.

a4: This interface exists between the agents. When the agents need to communicate, they can exchange the *Inform* message immediately through the a5 interface. Since the agents do not know where other agents are located and can transfer at any time, information exchange among the agents is generally conducted through the agent master.

a5: This interface refers to an interface between the agent manager and the agent master. The message passing through the interface is a SNMP (Simple Network Management Protocol) message. A SNMP message is composed of five messages including *Get*, *GetNext*, *Set*, *Response* and *Trap*. The information in the message is the MIB (Management Information Base) information necessary for controlling and managing the agents and the agent master.

3.3 JAAF Application Deployment

A client is an agent service user, and others are an agent service provider. The service provided by the agent application framework operates by the following steps. In the initial stage, the agent master starts the service and waits for the agent system to register. The agent system registers in the agent master by opening its service port. When registered, the agent system submits the service profile. This profile includes information regarding the system construction and the ability of agent, including communication method, the maximum number of threads or available memory size, and also notes whether network communication functions are provided and whether file or DB

service is offered. The client searches for an accessible agent system to use the available agent system and start a service by sending the *Service* message to the agent master. This search is done through the system profile and examines the location of agent the system, the level of service provided and the cost.

The client that requested the agent service can call out the necessary services and check the results by accessing its service through the registry and message service. Through the agent master, accessing for all agents as well as monitoring current operation locations and agents status is possible. The agent manager manages the agent and the agent system in the network through the managing MIB of the agent master. Service can be stopped and restarted by the client, agent master and agent manager. Information concerning the setting up the execution environments of the agent master is composed by the agent manager. The agent master is one of the points where it controls the agent in its own area. The agent manager can manage all of the various agents through the many agent masters within its allotted zone. This structure, through the thousands of agent systems and scores of agent masters, makes effective management possible. The agent MIB located in the lower part of the *iso.org.dod.internet.private.enterprise.snut.network.agent* of the complete MIB tree [15].

4 Agent Service Application using JAAF

By applying the agent application, a contents distribution network was implemented and monitored. In order to understand the network, definition of service components such as contents distribution server, contents distribution agent, contents distribution manager and client need to be examined. This structure is implemented in the dynamic fashion through the distribution agent and distribution manager.

4.1 Contents Distribution Networks

The contents distribution network is comprised of 3 domains: *service provider*, *service network* and *agent manager domain*. The service network is a tree structure and the original contents server is located at its root. Several agents are directly connected to the server and this constitutes the first level distribution network. On its lower location, other agents are connected and result in the gradual expansion of the network. A contents client is connected to the contents agent that is either the nearest from its location or favorable to its quality of service (QoS). The contents client receives the contents by attaching himself to the leaf of the connected agent tree network. Fig. 3 shows these components and the functions of the components in the contents service model are as follows.

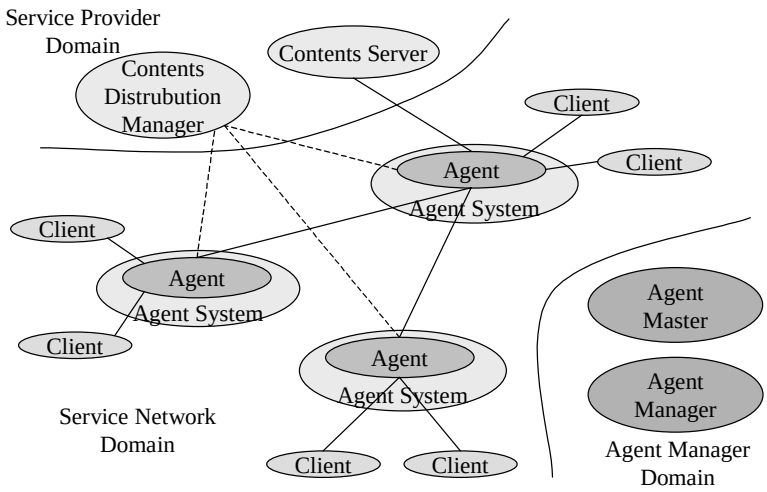

Fig. 3. The Components of the Contents Distribution Networks

Contents Server: The contents server is the root server that provides the information. The services provided by the contents server include broadcasting, video conferencing, real-time news services, and more. Methods of service provision can take various forms, including usage of stream, UDP datagram, and RTP service. In addition, depending on the types of services provided, there are various kinds of servers to be used, such as a push server where the client gathers information, and pull servers where the server provides information.

Contents Distribution Agent: A distribution agent takes the contents from the contents server and distributes them again to the client or other distribution agents. The distribution agents, depending on the contents types served, take different forms. Therefore, the agents locate themselves where the service providers want them to place and are served in a specific manner depending on the particular services and type of server. Like servers, agents also provide services with the contents client list and depending on the quality and forms of service, one or more threads are involved in the operations. An agent monitors the quality of service while at the same time providing services.

Contents Client: A client receives the contents from the agent distributor and sends it to users in the proper forms. In general, the communication handling and information handling portions constitute the client system.

Contents Distribution Manager: A contents distribution manager is the component that controls the data service between the server and the contents client. A contents distribution manager possesses all of the information regarding the server, distribution agent, and the client served, and manages the types of services provided by coordinating the distribution tree. In addition, the contents distribution manager is involved in the preparation of the list and the location of new agent through the service quality that is monitored and reported, and readjusts the distribution tree. The distribution manager, according to its location, locates the new agents into their

proper position within the distribution tree. When the contents client near the new agents requests a contents service, the distribution manager makes it possible to provide the service by connecting new agents to the proper agents. This system is the client component in the agent framework interface.

4.2 Characteristics of the Contents Distribution Networks

IP Multicast distributes group data efficiently. But forming a multicast router network is difficult to configure in the current Internet architecture [16]. As an alternative to IP multicast, an overlay multicast network has recently become a hot topic [17]. Contents distribution network is one implementation of such networks. Delivery bandwidth is required in backbone and leaf of the network. Specially, bandwidth reduction in the backbone network is very important. Contents distribution network saves bandwidth with the same amount of the multicast router network in a backbone area.

The transmission property of the delivery network requires some increased delay and jitter. But in the Intranet case, its delivery property is not significant and shows reasonable results. Transmission delays are measured from a few milliseconds to some tens of milliseconds depending on the condition of the network and host load. As the transmission hop of the tree is increased, delays are proportionally increased. Despite of this delay extension, the missing of the packet and average errors are not to be dramatically increased. Between autonomous systems through the Internet, the trans-mission delay property had more than 1 second in average delay and an average jitter of several hundreds to thousands of milliseconds.

Thus, the service provided by the contents distribution network that is composed of agents, reduces the required bandwidth dramatically. The delays and the jitter in the Intranet are not subject to significant changes. In the Internet environments, the service quality can be degraded in its nature. So, if the agent possesses a scheme that minimizes the jitter, the property of the service can be improved upon. Deployment of the agent at proper location increases the effect of bandwidth reduction and controls the service quality effectively.

5 Conclusions and Future Works

In this paper we present an agent framework and agent application service using the developed platform. The Java agent application framework is designed with the simplicity, accessibility, and manageability properties in mind. By using this framework, we can deploy new agent application services faster and more efficiently. In addition, each component's access can be improved upon by defining interfaces and messages among components. Agents can be controlled and managed effectively through the services provided by the agent master and the agent manager.

For providing the reliable and satisfactory service to clients, further study should be devoted in two areas; QoS guaranteeing scheme and strengthening security measures. Also, ensuring proper communications with other systems or languages in dif-

ferent types of platform settings would be needed. For this, defining and implementing the gateway functions for other service framework could be required.

References

1. Vu Anh Pham and Ahmed Karmouch, “Moble Software Agents: An Overview”, IEEE Communication Magazine., Vol. 36, No. 7, pp. 26-37, July 1998.
2. Alex L. G. Hayzelden and Rachel A. Bourne, Agent Technology for Communication Infrastructures, Weley, 2001.
3. R. Kawamura, R. Stadler, “Active Distributed Management for IP Networks”, IEEE Communications Magazine, Vol. 38, No. 4, pp. 114-120, April 2000.
4. Ichro Satoh, “Building Reusable Mobile Agents for Network Management”, IEEE Trans. on Systems, MAN, and Cybernetics-Part C: Applications and Reviews, Vol. 33, No. 3, pp. 350-357, Auguest 2003.
5. Antonio Liotta, George pavlou, and Graham Knight, “Exploiting Agent Mobility for Large-Scale Network Monitoring, IEEE Network”, Vol. 16, No. 3, pp. 7-15, May/June 2002.
6. Manuel G. and Torsten. B., “Internet Service Monitoring with Mobile Agents”, IEEE Network, Vol. 16, No. 3, pp. 22-29, May/June 2002.
7. Jin-Wook Baek and Heon-Young Yeom, “d-Agent: An Approach to Mobile Agent Planning for Distributed Information Retrieval”, IEEE Transaction on Consumer Electronics, Vol. 49, No. 1, pp.115-122, Feb. 2003.
8. Dinverno, M., and M. Luck, eds., “Understanding Agent Systems”, New York: Springer Verlag, 2001.
9. GMD FOKUS and IBM, Mobile Agent Facility Specification V1.0, Jan. 2000.
10. XC00023J, FIPA Agent Management Specification, 2002.12.6
11. Danny B. Lange and Mitsuru Oshima, Programming and Deploying Java Mobile Agent with Aglets, Addison Wesley, 1998.
12. F. Bellifemine, A. Poggi and G. Rimassa, “JADE – A FIPA-Compliant Agent Framework”, Proc. Fourth Int’l Conf. Practical Applications of Intelligent Agent and Multi-Agent Systems (PAAM ’99), pp. 97-108, April 1999.
13. http://fipa-os.sourceforge.net
14. http://www.agentcities.org
15. Kil-Hung Lee, “A Study of Agent Management Scheme”, Journal of Korea Computer Industry Education Society, Vol. 4, No. 3, pp. 191-198, March 2003.
16. J. Liebeherr, M. Nahas and W. Si, “Application-Layer Multicasting With Delaunay Triangulation Overlays”, IEEE JSAC, Vol. 20, No. 8, pp. 1472-1488, Oct 2002.
17. M. Castro, M.B. Jones, H. Wang and et el, “An evaluation of scalable application-level multicast built using peer-to-peer overlays”, IEEE INFOCOM 2003. Vol. 2, pp. 1510-1520, April 2003.