World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:2, No:8, 2008

Federal Open Agent System Platform

Hong-Bing Wang, Zhi-Hua Fan, and Chun-Dong She

Open Science Index, Electronics and Communication Engineering Vol:2, No:8, 2008 publications.waset.org/14168/pdf

***Abstract*—**Open Agent System platform based on High Level Architecture is firstly proposed to support the application involving heterogeneous agents. The basic idea is to develop different wrappers for different agent systems, which are wrapped as federates to join a federation. The platform is based on High Level Architecture and the advantages for this open standard are naturally inherited, such as system interoperability and reuse. Especially, the federal architecture allows different federates to be heterogeneous so as to support the integration of different agent systems. Furthermore, both implicit communication and explicit communication between agents can be supported. Then, as the wrapper RTI_JADE an example, the components are discussed. Finally, the performance of RTI_JADE is analyzed. The results show that RTI_JADE works very efficiently.

***Keywords*—**Open Agent System, High Level Architecture, Heterogeneous Agents, Wrapper.

I. Introduction

MULTI Agent System is emerging as an appealing paradigm for modeling and developing large, complex and distributed information systems. With the development of Multi Agent System, Open Agent System has attracted more and more researchers' attention due to the two factors: more application range and emphasis on supporting openness. However, research on Open Agent System is currently at the preliminary stage. Open Agent System belongs to the domain of Multi Agent System. Luck made the prediction about the development of agent systems [1]. Agent systems involving heterogeneous agents will be achieved in a foreseeable future and not before 2008.

Many agent platforms have been obtained, including JADE [2], JAMES [3], Aglet [4], JACK [5], RePast [6], SIM_AGENT [7] and Cougaar [8]. Based on these platforms, many different agent systems have been developed. Therefore, it is not advisable to develop from scratch in order to build an Open Agent System without considering the integration of these legacy agent systems. However, different types of agents are difficult to interoperate each other because of lacking in a standard running platform. Although two communication standards, FIPA ACL and KQML, partly solve the problem of interoperation at the semantic level, the running platform supporting heterogeneous agents still does not appear. Further, FIPA ACL and KQML are only for explicit communication and the standard for implicit communication is not considered in the existing studies.

To support interoperability between heterogeneous agents, Genesereth proposed a federal architecture [9]. Agents communicate not directly with each other but indirectly via a media called facilitator. The correctness of communication is ensured by the facilitator. The autonomy of agents is sacrificed to some extent. As for the Client/Server architecture, there are some distinguished advantages with the federal architecture, such as dynamic configuration and easy integration [10]. Additionally, agents in different federates can be heterogeneous by the shielding of facilitator only if all the agents obey the communication rules built in the facilitator. This architecture has been applied in the famous agent system ARCHON [11]. However, the facilitator is application-specific in ARCHON and so its generality is a great problem.

In order to make much wider use of the federal architecture, facilitator must accord to a general standard. In this paper, High Level Architecture (HLA) [12] is taken for this purpose.

HLA is a common standard for distributed modeling and simulation with the core of the federal architecture. Interoperability and reusability are two major goals for HLA. Interoperability is the ability of system components to exchange data and interpret the data in a consistent way. Reusability is facilitated by having components with commonly understood behavior and well defined interfaces. HLA is composed of three specifications including framework and rules, interface specification and object model template. HLA framework and rules outline the responsibilities of federate and federation to ensure a consistent implementation. HLA interface specification defines the standard services and interfaces to be used by the federates in order to support efficient information exchange. These interfaces are arranged into six basic service groups as follows: (1) Federation management services offer basic functions required to create and operate a federation; (2) Declaration management services support an efficient management of data exchange through the information provided by federates; (3) Object management services provide creation, deletion, identification and other services for the actual transfer of data, including the updating and reflecting of object class, and, the sending and receiving of interaction class; (4) Ownership management services support the dynamic transfer of ownership of HLA object-instance attributes during a federation execution; (5) Time management services support the synchronization of runtime federates; (6) Data distribution management services support the efficient routing of data among federates during the course of a

WANG Hong-Bing is with the General Software Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China (phone: 0816-62614140-8107; e-mail: wahobi@chinaacc.com).

FAN Zhi-Hua is with the Institute of Software, Chinese Academy of Sciences, Beijing 100080, China (e-mail: fan_zhihua@chinaacc.com).

SHE Chun-Dong is with the Institute of Software, Chinese Academy of Sciences, Beijing 100080, China (e-mail: shrcd@chinaacc.com).

Open Science Index, Electronics and Communication Engineering Vol:2, No:8, 2008 publications.waset.org/14168/pdf

federation execution. HLA object model template defines the format and syntax of object modes.

Although HLA is from the domain of simulation, it has been applied in a full range of areas successfully, including education, training, analysis, engineering, and entertainment. In fact, the six classes of services defined by HLA interface specification except time management services have a strong generality.

There have appeared some studies using agents in the HLA-based application. Dannie discussed the feasibility of autonomous objects used in the simulation based on HLA [13]. Andersson proposed several different methods in the design of agent simulation systems based on HLA [14]. Less investigated the distributed simulation of agent-based systems with HLA [15]. Minson distributed RePast simulations with HLA [16]. Wang studied agent communication in distributed simulations [17]. Logan discussed the application of distributed discrete event simulation techniques to the simulation of Multi Agent Systems [18]. However, all these studies commonly lack in the consideration of the fundamental role of federal architecture of HLA for Open Agent System.

Therefore, Open Agent System platform based on HLA is especially proposed. This paper is organized as follows: Section II proposes the platform. Section III discusses the development of the wrapper RTI_JADE. Section IV analyzes the performance of RTI_JADE. Section V concludes with our work and discusses some future research directions.

II. Open Agent System Platform

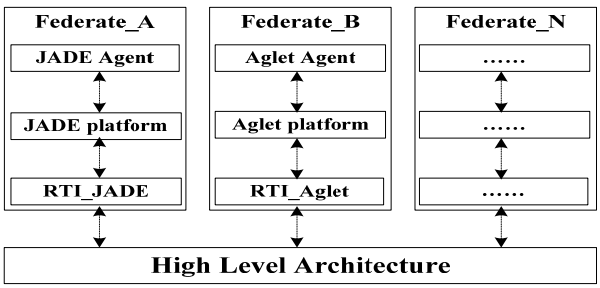


Fig. 1 Open Agent System Platform based on HLA

Open Agent System platform is presented in Fig. 1. The basic idea is to develop different wrappers for different agent systems, which are wrapped as federates to join a federation. The agents in the same federate must be homogeneous and the agents in different federates can be heterogeneous.

To develop different wrappers for different agent systems may require much effort. In fact, these wrappers have similar components. So the experience of developing a wrapper can easily be used in developing another wrapper. The wrapper RTI_JADE will be discussed in detail in Section III.

A. Entity Hierarchy

Generally, agent has its own control thread and federate is always an independent program. If agents are concretely integrated in the federate, update on the legacy agent systems may require much work. In this paper, the mapping method between agents and object class instances in the federate is used to reduce the coupling between agents and federates as much as possible. The entity hierarchy in the platform is shown in Fig. 2.

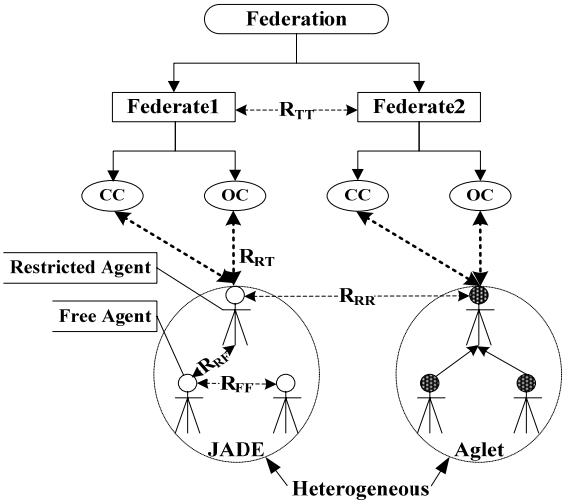


Fig. 2 Entity Hierarchy in the Platform

Not all agents in the legacy agent systems must be introduced into federates. Those agents, such as Directory Facilitator Agent and Resource Monitor Agent, working in the local domain, should not be modified in code because they have not any communication with remote agents. Therefore, agents are firstly divided into two types: restricted agents and free agents according to whether they have the need to communicate with remote agents. Only restricted agents are introduced into federates and thus these agents' autonomy is restricted by HLA. When a restricted agent is introduced into a federate, not the agent is concretely introduced into the work space of the federate, but a mapping object class instance of HLA, a snapshot storing the corresponding agent's attributes, is created in this federate.

Let $ES=AS \cup FS$ be the set of all entities. $AS=AS_R \cup AS_F$ is the set of all agents. AS_R is the set of restricted agents. AS_F is the set of free agents. FS is the set of federates. Five types of relationships are defined as follows:

(1) $R_{FF}=\{<Ag_i,Ag_j> / Ag_i \in AS_F, Ag_j \in AS_F\}$. Both Ag_i and Ag_j run in the legacy agent systems. They are both free and therefore have not the corresponding object class instances. The interaction relationship is maintained by the legacy agent systems and they interact via the communication channel provided by the agent platform.

(2) $R_{RF}=\{<Ag_i,Ag_j> / Ag_i \in AS_R, Ag_j \in AS_F\}$. Both Ag_i and Ag_j run in the legacy agent systems. Ag_i is restricted and it has the corresponding object class instance in the federate. Ag_j is free. The interaction relationship is also maintained by the legacy agent systems and they interact also via the communication channel provided by the agent platform.

(3) $R_{RR}=\{<Ag_i,Ag_j> / Ag_i \in AS_R, Ag_j \in AS_R\}$. Both Ag_i and Ag_j run in the legacy agent systems. They are both restricted and have the corresponding object class instances. The relationship

Open Science Index, Electronics and Communication Engineering Vol:2, No:8, 2008 publications.waset.org/14168/pdf

is maintained commonly by the wrapper and RTI. When two restricted agents are located in different federates, they interact indirectly via RTI.

(4) $R_{RT}=\{<Ag_i,F_j> / Ag_i \in AS_R, F_j \in FS\}$. Ag_i runs in the legacy agent systems. F_j runs as the form of federate. Ag_i is restricted and has the corresponding object class instance. The mapping relationship is maintained in the wrapper.

(5) $R_{TT}=\{<F_i,F_j> / F_i \in FS, F_j \in FS\}$. Both F_i and F_j run as the form of federate. The interaction relationship is maintained by RTI and they interact directly via the communication channel provided by RTI.

R_{RR} and R_{RT} are our focuses as they need to be maintained by our developed wrappers. In this paper, to satisfy the requirement of HLA that all exchange of data among federates will occur via RTI during the federation execution, the interaction between restricted agents in different federates is indirectly via RTI instead of directly via the communication channel provided by the agent platform.

B. Implicit Communication and Explicit Communication

In the above platform, HLA plays the important role of facilitator. Two types of communication between agents, implicit communication and explicit communication [19], are both supported on the basis of object management services of HLA, as shown in Fig. 3. Implicit communication generally refers to observation, i.e., the agent to obtain information takes the initiative to observe some variables in the sharing information space, while explicit communication means that the sender agent has the intention to interact with the receiver agent by the way of message passing. The important difference between them is that implicit communication has the persistent property and explicit communication has not. The variables to be observed in implicit communication have not been removed after one observation ends. But the message passing in explicit communication is instant.

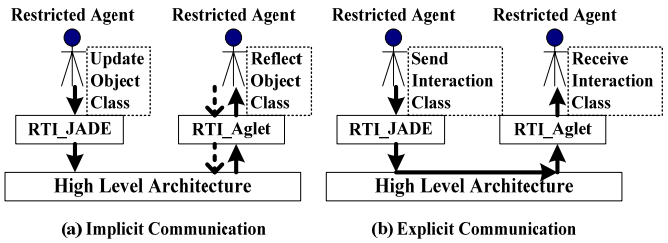


Fig. 3 Implicit Communication and Explicit Communication

Generally, two agents interact in the way of explicit communication. However, explicit communication is not suitable in some situations. For example, two warring agents in the battlefield simulation can not communicate explicitly and implicit communication is required. But it is obvious that this observation is a partial information observation and the modeling and reasoning about other agents are needed. This is out of our scope. In this paper, we assume that implicit communication is based on complete information.

In the internal RTI, implicit communication is based on the updating and reflecting of object class and explicit communication is based on the sending and receiving of interaction class. They both belong to the class of object management services of HLA and support group-cast.

III. RTI_JADE Wrapper

This section will take RTI_JADE as an example to discuss the development of wrappers. JADE 1.3 and pRTI1516 are preferably selected. JADE 1.3 is a well-known agent system platform supporting FIPA specification, a world-wide agreed agent standard. pRTI1516 has the best performance among the existing run-time interface softwares of HLA.

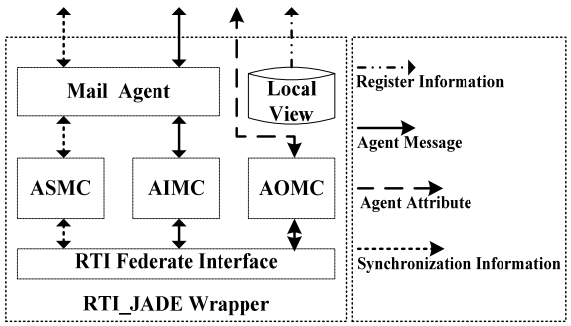


Fig. 4 RTI_JADE Wrapper

The wrapper RTI_JADE consists of three components: Agent Object Management Component (AOMC), Agent Interaction Management Component (AIMC) and Agent Synchronization Management Component (ASMC), as shown in Fig. 4. AOMC's functions are to maintain the mapping relationship R_{RT} between restricted agents and object class instances, and to realize implicit communication between restricted agents. This mapping relationship is stored in the local view of the wrapper. AIMC's function is to realize explicit communication between restricted agents. Both implicit communication and explicit communication between restricted agents involve in maintaining the relationship R_{RR}. ASMC is to synchronize all the local restricted agents in a federate when this federate is required to synchronize with other federates. Finally, a special agent, called mail agent, is introduced as the message passing inter-media for AIMC and ASMC.

In order to support implicit communication, individual agent needs two abilities: sensing and effecting its environment. Two interface objects, Sensor and Effector, are added [17]. A Sensor object enables an agent to observe its environment and an Effector object enables it to output its attributes into the environment. Both of them are implemented via the Object-to-Agent (O2A) communication channel provided by JADE platform. Besides, in order to support explicit communication, individual agent needs two buffers, InBuffer and OutBuffer, which respectively store the incoming messages and the outgoing messages.

In fact, many agent systems provide the two mechanisms of message-passing and environment-sensing. So the update

Open Science Index, Electronics and Communication Engineering Vol:2, No:8, 2008 publications.waset.org/14168/pdf

about restricted agents is very simple when the integration of heterogeneous agent systems is considered.

A. AOMC

One of AOMC's functions is to maintain the mapping relationship R_{RT} between restricted agents and object class instances. When an agent is introduced into a federate, its identifier is recorded in the local view and an object class instance is created in this federate. While an agent is removed from the federate, its identifier is deleted from the local view and the corresponding object class instance is deleted.

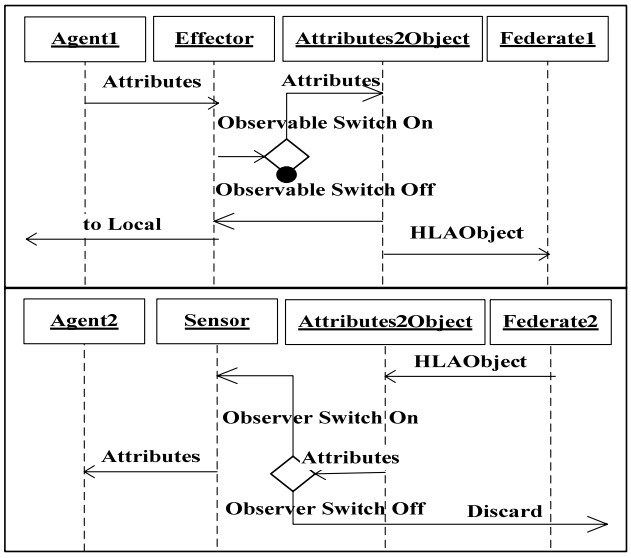


Fig. 5 Sequence Diagram of Implicit Communication

The more important function about AOMC is to realize implicit communication between restricted agents. The sequence diagram of implicit communication is shown in Fig. 5. In the internal RTI, it is based on the updating and reflecting of object class, whose routing is defined by data distributed management services of HLA. In RTI_JADE, the switches at two ends, observable and observer, are dynamically controllable, i.e., an agent can invalidate or validate its sensing or effecting ability at any time. Attributes2Object is to pack agent's attributes into object class instances, and to unpack object class instances into agent's attributes.

The combination between the switch mechanism at the agent level and the routing mechanism at the federate level forms a complete, flexible and hierarchical data filtering mechanism about implicit communication among restricted agents all over the federation. The relationship R_{RR} of implicit communication is maintained by this combinational mechanism.

B. AIMC

AIMC is to realize explicit communication between restricted agents. The sequence diagram of explicit communication is shown in Fig. 6. Here, we assume that: (1) All outgoing messages of a restricted agent are firstly passed to the local mailbox agent; (2) A mailbox agent can only send messages directly to local agents which are located in the same federate.

When a restricted agent sends a message, the message is firstly passed to the mail agent. The mail agent checks whether the destination agent of the message is in the local view or not. If so, the message is directly passed to the destination. If not, the message is passed to a transformer ACL2Interaction, which is to realize the transformation between ACLMessages and interaction classes of HLA. Then, the corresponding interaction class is received by another federate. The routing depends on data distributed management services of HLA. The federate transforms the HLA interaction class into ACLMessage and check whether the destination agent of the ACLMessage is located in this local federate. If so, the ACLMessage is passed to the destination by the mail agent in this federate. If not, the ACLMessage is discarded.

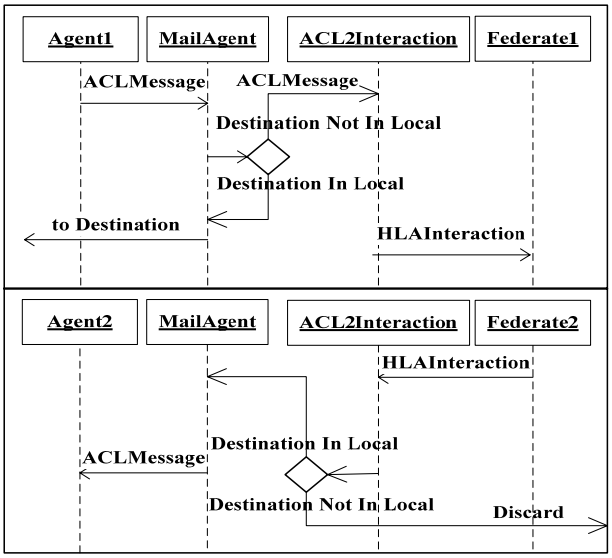


Fig. 6 Sequence Diagram of Explicit Communication

The combination between the local view mechanism and the routing mechanism also forms a complete, flexible and hierarchical data filtering mechanism about explicit communication between restricted agents all over the federation. The relationship R_{RR} of explicit communication is maintained by this combinational mechanism.

C. ASMC

ASMC is to synchronize all the local restricted agents in a federate. The sequence diagram of agent synchronization management is shown in Fig. 7. When RTI invoke the callback *announceSynchronizationPoint* to inform a joined federate of the existence of a new synchronization point, it shows that the request of synchronization has been issued. The federate firstly passes a *SynStart* message to the local mail agent. The mail agent then broadcasts an *AgentSyncStart* message to all the local restricted agents in the federate. When these agents have made ready for synchronization, they all send an

Open Science Index, Electronics and Communication Engineering Vol:2, No:8, 2008 publications.waset.org/14168/pdf

AgentSyncAchieve message to the mail agent. When the mail agent receives all the *AgentSyncAchieve* messages, it passes a *SyncAchieve* message to the federate. The federate then call the function *synchronizationPointAchieved* to inform RTI that it has achieved the registered synchronization.

The combination between the local synchronization mechanism between restricted agents in the federate and the global synchronization mechanism between federates in the federation forms a complete and hierarchic synchronization mechanism for all restricted agents all over the federation. This is very important for the simulation of heterogeneous agent systems.

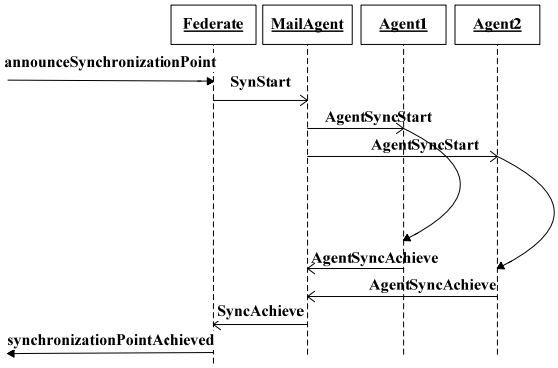


Fig. 7 Sequence Diagram of Agent Synchronization Management

IV. Performance Analysis

The overall communication delay, implicit or explicit, between restricted agents in the platform can be attributed to three aspects: (1) the performance of RTI; (2) the performance of JADE; (3) the performance of the wrapper RTI_JADE. Here, (1) and (3) are tightly related with the system platforms. (2) is our focus.

Our experiment proceeds with two restricted agents respectively located in two federate, PING federate and PONG federate. The average value every 100 times at the PING federate is taken as a sample point. The experiment environment is that two computers with the hardware configuration P4 2.6GHz/256M RAM are connected by the two 100M network cards.

A. Performance Analysis of Implicit Communication

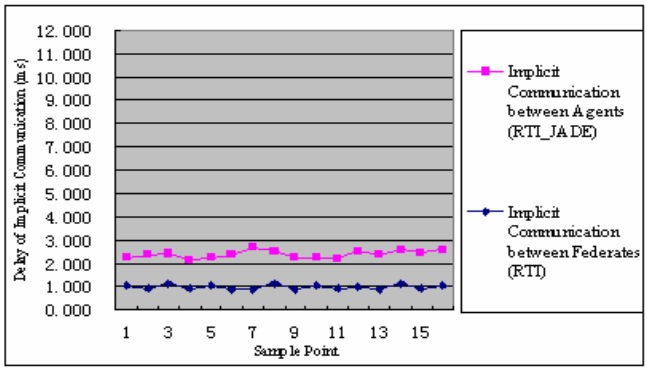


Fig. 8 Performance Comparision of Implicit Communication

Implicit communication is based on the updating and reflecting of object class in the internal RTI. The size of object class attributes is about 100 bytes. Delay of implicit communication, between agents via RTI_JADE and further RTI, and between federates via RTI, are compared in Fig. 8.

The delay of implicit communication between federates via RTI is about 0.95ms and the delay of implicit communication between agents via RTI_JADE and further RTI is about 2.4ms. The delay caused by the wrapper can be calculated as follows: (2.4-0.95)/2=0.725ms. This shows that the performance of RTI_JADE about implicit communication is very high. This is because the O2A channel of JADE works very efficiently.

B. Performance Analysis of Explicit Communication

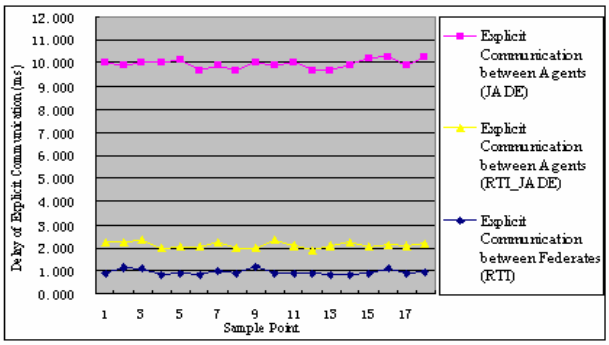


Fig. 9 Performance Comparison of Explicit Communication

Explicit communication is based on the sending and receiving of interaction class in the internal RTI. The size of interaction class is also about 100 bytes. Delay of explicit communication, between federates via RTI, between agents via RTI_JADE and further RTI, and between agents via JADE, is compared in Fig. 9.

The delay between federates via RTI is about 0.95ms. The delay between agents via RTI_JADE and further RTI is about 2.1ms. The delay caused by the wrapper can be calculated as follows: (2.1-0.95)/2=0.575ms. The interaction delay between agents directly by JADE is about 10ms. This shows that the performance of RTI_JADE about explicit communication is also high. This is because that restricted agents and the mail agent are both located in the same container of JADE and the high efficient event mechanism is used. While two agents located in two different containers of JADE take IIOP as their interaction way, the delay of explicit communication between agents via JADE is relatively higher. This shows RTI works much more efficiently than JADE with respect to explicit communication.

V. Conclusion

Open Agent System platform based on High Level Architecture is firstly proposed. The basic idea is to develop different wrappers for different agent systems, which are wrapped as federates to join a federation. As the platform is based on High Level Architecture, the advantages for the open standard are naturally inherited, such as system interoperability and reuse, especially, the federal architecture allows different federates to be heterogeneous so as to support the integration of

heterogeneous agent systems. Furthermore, both implicit communication and explicit communication between restricted agents can be supported based on the object management services of HLA. Then, as the wrapper RTI_JADE as an example, the development for three components, AOMC, AIMC and ASMC is discussed and the performance about RTI_JADE is comparatively analyzed. The results show that RTI_JADE works very efficiently.

There are a number of issues for future research. A paramount issue is the development of other wrappers. Another issue is the performance analysis of the local synchronization mechanism. The third issue is to integrate multiple standards for explicit communication, including FIPA ACL and KQML. Finally, in order to support transparent delegation, i.e., service matchmaking, often required by Open Agent System, a general service description language will be developed and further be integrated into the object model template of HLA.

REFERENCES

[1] M. Luck, P. McBurney and C. Preist. Agent Technology: Enabling Next Generation Computing. AgentLink. http://www.agentlink.org/roadmap.
[2] F. Bellifemine, A. Poggi and G. Rimassa. JADE – A FIPA-compliant agent framework. In: Proc. of PAAM-99. London, UK. 1999, pp. 97-108.
[3] A. M. Uhrmacher, P. Tyschler and D. Tyschler. Modeling mobile agents. In Proc. of the International Conference on Web-based Modeling and Simulation, part of the 1998 SCS Western Multiconference on Computer Simulation. San Diego , California. 1998, pp. 15–20.
[4] Aglet. Tokyo Research Laboratory IBM Corporation. http://www.trl.ibm.co. jp/aglets/
[5] P. Urlings, J. Tweedale, C. Sioutis, N. Ichalkaranje and L. Jain. Intelligent Agents as Cognitive Team Members. In Proc. of the 10th International Conference on Human-Computer Interaction. Crete. Greece. 2003, pp. 723-733.
[6] Nick Collier. RePast: An Extensible Framework for Agent Simulation. http://repast.sourceforge.net.
[7] A. Sloman and R. Poli. SIM AGENT: A toolkit for exploring agent designs. In M. Wooldridge, J. Mueller and M. Tambe, editors.: Intelligent Agents II: Agent Theories Architectures and Languages (ATAL-95). Springer–Verlag, 1996, pp. 392-407.
[8] Cougaar. Cougaar Open Source Software. http://www.cougaar.org.
[9] M. R. Genesereth and S. P. Ketchpel. Software agents. Communications of the ACM. 1994, vol. 37, no. 7, pp. 48-53.
[10] N.B. Wang, X.F. Xu, G. Wang and S.C. Deng. Designing Federation Multi-agent System Based on Ontology. Computer Engineering. 1999, vol. 25, no. 3, pp. 50-52
[11] N. R. Jennings, L. Varga, R. Aarnts, J. Fuchs and P. Skarek. Transforming Standalone Expert Systems into a Community of Cooperating Agents. Engineering Applications of AI. 1993, vol. 6, no. 4, pp. 317-331.
[12] DMSO: High Level Architecture Rules, Interface Specification, Object Model Template. Version 1.3. 1998. http://www.dmso.nil.
[13] E. Dannie, P. John and A. Stephan. Feasibility and Functionality of Autonomous Objects in the HLA. In Proc. of the 1997 Spring Simulation Interoperability Workshop. Orlando, FL. No: 97S-SIW-055, 1997, pp. 3-7.
[14] J. Andersson and S. Lof. HLA as Conceptual Basis for a Multi-Agent Environment. Technical Report 8th-CGF-033, Pitch Kunskapsutveckling AB, 1999.
[15] M. Lees, B. Logan, T. Oguara and G. Theodoropoulos. HLA_AGENT: Distributed Simulation of Agent-Based Systems with HLA. In Proc. of the International Conference on Computational Science (ICCS'04). Huntsville, Alabama USA. 2004, pp. 907-915.
[16] R. Minso and G. Theodoropoulos. Distributing Repast agent-based simulations with HLA. In Proc. of the 2004 European Simulation Interoperability Workshop, Edinburgh, Simulation Interoperability Standards Organization and Society for Computer Simulation International. 2004.
[17] F. Wang, S.J. Turner and L. Wang. Integrating Agents into HLA-based Distributed Virtual Environments. In Proc. of the Fourth Workshop on Agent-Based Simulation (ABS2003). Montepellier, France. 2003, pp. 9-14.
[18] B. Logan and G. Theodoropoulos. The Distributed Simulation of Multi-Agent Systems. In Proc. of the IEEE - Special Issue on Agent-Oriented Software Approaches in Distributed Modeling and Simulation. 2001, vol. 89, no. 2, pp.174-185.
[19] Michael Van Wie. Role Selection in Teams of Non-communicating Agents. Thesis for the Degree Doctor of Philosophy. University of Rochester. 2001.

Open Science Index, Electronics and Communication Engineering Vol:2, No:8, 2008 publications.waset.org/14168/pdf