Agent Communication in Distributed Simulations

Fang Wang, Stephen John Turner, and Lihua Wang

Parallel & Distributed Computing Centre, School of Computer Engineering,
Nanyang Technological University, 639798 Singapore
{pg01538896, ASSJTurner, pg00960276}@ntu.edu.sg

Abstract. Multi-Agent Systems (MASs) provide a valuable tool for handling increasing software complexity and supporting rapid and accurate decision making. This paper focuses on how an MAS can be applied to represent certain entities in a distributed simulation or virtual environment. We address the communication issue which is very important for an MAS in simulation. The causality problem is also investigated, and conditions for ensuring consistency are identified. A general architecture is described and appropriate synchronization mechanisms given. A prototype system has been implemented, and some experimental results are presented.

Keywords: multi-agent systems, distributed simulation, communication, causality, synchronization

1 Introduction

Today's software applications are mainly characterized by their component-based structures which are usually heterogeneous and distributed. Agent technology provides a method for handling increasing software complexity and supporting rapid and accurate decision making. A number of different approaches have emerged as candidates for the agent architecture, and at the same time, dozens of environments for modelling, testing and finally implementing agent-based systems have been developed [1], among which JADE [2] is a well-known MAS (multi-agent system) development kit supporting the FIPA specifications, a world-wide agreed agent standard.

Since the concept of agent has become widely used during the last few years, it has already become involved with simulations. Commonly, there are three main research areas in the cross field of combining agents with simulations, namely, (1) to simulate an agent-based system in order to learn more about its behavior or investigate the implications of alternative architectures, (2) to use agents as entities in a simulation or virtual environment, (3) to utilize agents as a way of controlling simulations and providing services. Actually most of the current research in using parallel and distributed simulation techniques in multi-agent and multi-agent based simulation is addressed to the first area [3–5]. Our work is concerned with the second area, namely to develop autonomous agents for representing entities in distributed simulations.

Distributed simulation enables participants located in different geographical locations to share a common virtual world. The HLA (High Level Architecture) [6] is an industry (IEEE-1516) standard for modelling and simulation. It is increasingly being

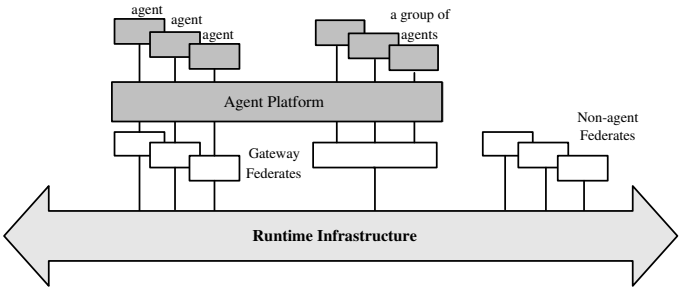

Fig. 1. An architecture for an MAS in an HLA-based simulation

used in various simulation areas, including education, training, analysis, engineering, entertainment and games. In the HLA, a distributed simulation is called a *federation*, and each individual simulator is referred as a *federate*, one point of attachment to the Runtime Infrastructure (RTI). A federate can be a computer simulation, it can also be a physical device, a passive data viewer or an interface to a human participant.

The novelty of our project comes from the way in which autonomous agents are integrated into a distributed simulation, with particular attention being given to the efficient use of HLA services. There are many applications such as battlefield simulations, interactive games, etc. that provide the motivation for this integration. Other work on integrating agent toolkits with the RTI includes SIM_AGENT [7].

In this paper, we mainly address the communication issue which is very important for an MAS in simulation, and investigate the causality problem. The rest of the paper is organized in this way: Section 2 presents an approach to integrating agents into HLA-based simulations. Causality and message ordering problems are discussed in Section 3. Section 4 illustrates how distributed agents can be enabled to communicate via the HLA. Section 5 introduces a prototype system, and Section 6 gives some experimental results. Finally, conclusions are made in Section 7.

2 Integration of Agents into Distributed Simulations

Multi-Agent Systems (MASs) have both the traditional advantages of distributed and concurrent problem solving, and the additional advantage of sophisticated patterns of interactions [8]. To integrate agents into an HLA simulation, different approaches have been made toward building up the overall architecture [3]. For a common and widely adoptable architecture, a fundamental concern is indeed an appropriate middleware constructed between the agents and the RTI.

As shown in Figure 1, we have proposed an architecture to build up a prototype system. A *gateway federate* is developed to take charge of agents. The federates are built upon the RTI and they can access the RTI interactively. The JADE platform has been selected to support the agents and their communication using ACL (Agent Communication Language). This standard agent platform also contains some agents including the Agent Management Service (AMS), Directory Facilitator (DF) and Remote Moni-

toring Agent (RMA) that support general agent-specific services for all the other agent entities.

In the architecture, every agent is attached to a gateway federate. We can have several agents in a group attached to the same gateway federate or a single agent attached to a separate gateway federate. At the same time, different federates can also be put in the same machine or different machines. In order to enable agents to interact with their environment, we develop two interfaces to play the role of interaction channels between the federate and the agent, namely, a *sensor* object and an *effector* object. A *sensor* enables an agent to perceive the environment within which it is embedded and an *effector* enables it to act upon the environment as a response. Both of them are transferred via the Object-to-Agent (O2A) communication channel provided by the agent toolkit [2]. With these facilities, a completed interaction mechanism for an embedded agent comes into being, and a three-phased (*sense-deliberate-react*) agent architecture can be achieved.

In the HLA simulation federation, all federates including agent gateway federates and the environment federate use the RTI infrastructure to interact, and at the same time, to keep the consistency of the whole virtual environment. One of the HLA rules (rule 3) states, "During a federation execution, all exchange of FOM [1] data among federates shall occur via the RTI." Thus to satisfy the requirements of the HLA and to guarantee the consistency of an application when it is distributed, we must make the communication between agents in different federates indirect via the RTI instead of direct via the communication channel provided by the agent platform. This will be further discussed in Section 4.

As we see, a good agent-specific services *middleware* is obtained combining gateway federates and the agent platform together. Moreover, with the HLA standard RTI infrastructure in place at the lower level, a similar standard agent platform is adopted at the higher level. In this way, a general architecture is obtained, and the overall architecture has great scalability to be applied in many other simulations.

3 Causality and Message Ordering

As we know, a basic requirement of a distributed simulation is to ensure that the individual simulators (federates) perceive a common view of the virtual environment. For example, it is important that any pair of participants should perceive a set of messages/events in the same sequence. For this purpose, a time management system is required.

Considering a collection of simulators interconnected via a network, it is not difficult to give a scenario leading to a problem of consistency violation without an effective time management system. Suppose that a treasure detector agent a discovers a gold bar, and sends a message to other agents including a picker named b and an observer named o, causing b to pick up this gold bar as a consequence. In the distributed simulation, the messages about the gold bar are as follows: the simulator of a generates a message

[1] The FOM (Federation Object Model) defines the data to be exchanged between federates during a simulation.

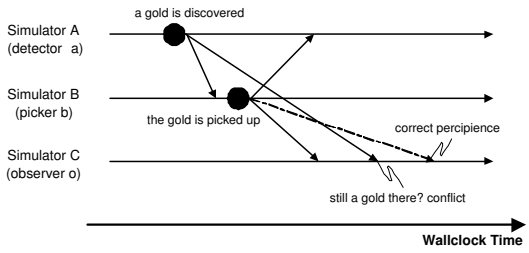

Fig. 2. A scenario of distributed simulation leading to a violation of causality.

indicating where the gold bar is. This message is sent to all the other participant simulators including b and o. Upon receiving this message, the gold bar will then be picked up in the simulator of the picker b. Another agent o receives both the messages about the finding from the detector a and the picking up from the picker b. The message about the finding may possibly arrive later than the message about the picking up action, delayed in the network. This causes the observer o to perceive the picking up action first before he knows some gold bar has been discovered. It must be very strange for o that he gets to know a gold bar's location when it has actually been taken away. So is the gold bar still there? This gives rise to a conflict. Thus to keep the causality of the system, messages should arrive in a correct order, that is, the message of discovering a gold bar must arrive before the message of picking up the gold bar. Figure 2 illustrates a scenario about this problem.

Actually this kind of problem is very common in simulations especially when they are distributed. Causality can be always kept naturally in the physical world, while in the distributed simulation, it might be affected by many factors such as the latency a message encounters as it is transmitted through the network. Accordingly, a time management system is required to control a distributed simulation avoiding the violations of causality. It can also ensure that repeated executions of the simulation with the same initial state and external inputs produce entirely the same result. The HLA has a time-stamp order message delivery service [9]. All federates in a federation should explicitly request an advance in their local virtual times. A time advancement request is granted only if no events containing a smaller time-stamp will later arrive at this federate. Furthermore, in a time-stepped simulation, the local virtual time vt advances with a fixed time increment Δvt. Generally, in this case, we need some additional conditions from the agent's point of view, such as:

- *An agent should have received all the messages of the last time step before it can process them and advance to another time step.*
- *All the messages that are sent at the same time step should also arrive at the same time step.*

In this way, all messages sent at time step vt will be received at time step $vt + \Delta vt$ as the global time advances, thus the system's consistency can be well kept between different federates and agents.

4 How Agents Communicate using HLA

For delivering ACL messages, the simplest way is to send them directly to those specified agents listed in the receiver slot, with the facility of the agent toolkit. This can be achieved easily no matter whether those agents are in the same physical machine or not. But according to the rule in Section 2 and the discussions in Section 3, this way may cause some consistency violation and is obviously inappropriate.

An alternative way is to send ACL messages indirectly via the RTI utilizing some mechanism. In this case, each ACL message can be represented by an interaction sent between federates, and every gateway federate needs a mailbox (MB) agent as an accessory to take charge of the ACL message transmissions. Moreover, two constraints need to be made on agent interactions, they are:

1. *all outgoing ACL messages of an agent are passed to the local mailbox agent to be processed,* and
2. *a mailbox agent can only send ACL messages directly to local agents that are located in the same federate.*

Those agents in different federates use mailbox agents as the communication middleman and they do not interact with each other directly. These constraints ensures that all exchange of FOM data among federates occurs via the RTI.

4.1 The Mailbox Agent

The mailbox agent has two interfaces connected with its federate. One is *mailboxIN* that contains the received interactions from remote gateway federates, and the other is *mailboxOUT* that contains the interactions to be sent to remote gateway federates. It is noticeable that all the incoming messages collected by a mailbox agent from local agents can be divided into three categories to be processed (Table 1). Here the term *local* indicates that the receivers are in the same gateway federate and the term *remote* indicates the opposite. It is possible that an ACL message is sent to both local and remote agents.

Table 1. Behavior of Mailbox Agent

Receivers	Category	Sent to agent	Sent as interactions
local agents	$isLocal$	yes	no
remote agents	$isRemote$	no	yes
local and remote agents	$isLocal$ & $isRemote$	yes	yes

In summary, when an agent need to send ACL messages to other agents, a complete transmission procedure consists of following steps:

1. The agent passes the messages to the local mailbox agent (MB).
2. The MB collects all the messages from local agents.
   (a) If the messages are sent to some local agents, the MB sends them accordingly.

(b) Otherwise,
   i. the MB packs the ACL messages into the *mailboxOUT*, and passes it to the gateway federate.
   ii. The gateway federate reads the *mailboxOUT*, and encodes each ACL message into a byte sequence which is then sent to all the other federates as a time-stamped interaction.
   iii. Another gateway federate receives these interactions, decodes them into ACL messages and packs these messages again into the *mailboxIN* preparing for the subsequent transmission to its MB.
   iv. The MB receives the the *mailboxIN*, and unpacks it into the original ACL messages. The MB then delivers these messages to the local agents that the receiver lists may include.

3. Finally all the destination agents receive the ACL messages.

In this approach, we ensure a federation does not exchange data representing state changes of shared object instances or interactions outside of the RTI service, thus the consistency of the distributed application is not violated.

4.2 Synchronization between Federates and Agents

In our prototype, the RTI plays the role of the communication system for all the federates, providing the possibility for them to exchange data, and also to synchronize their activities. But the gateway federate and the agent entities still have different threads that proceed in parallel. Thus a crucial problem arises of how to synchronize the threads of the gateway federate, the mailbox agent and the common autonomous agents to keep the consistency of the simulation.

For this purpose, a condition variable called $latch$ is introduced here, both between the mailbox agent and the gateway federate, and between the autonomous agent and the gateway federate. Every $latch$ is an object shared by an agent entity and its gateway federate. We let the gateway federate wait on the $latch$ until the agent thread signals it, so that some synchronization points for parallel proceeding are established. Figure 3 shows how a gateway federate synchronizes with its mailbox agent and autonomous agents. The arrows show time dependency, for example, $m3 \rightarrow f3$ means $t_{m3} < t_{f3}$, i.e., there exists a time dependency between them. To ensure that all the sent ACL messages have reached the agents before the agents start to process them, we make a restriction, that is, to let the mailbox agent wait until it gets a return receipt for each sent message.

5 The Prototype System

Currently a prototype system named MSG (Mine Sweeping Game) [10] has been implemented, in which several soldiers roam to detect and clear all mines out of an area as to keep the area away from dangers. Figure 4 gives a snapshot of this game.

The soldiers and the environment are represented by different federates distributed in a local area network. We can have a group of soldiers executing in the same federate

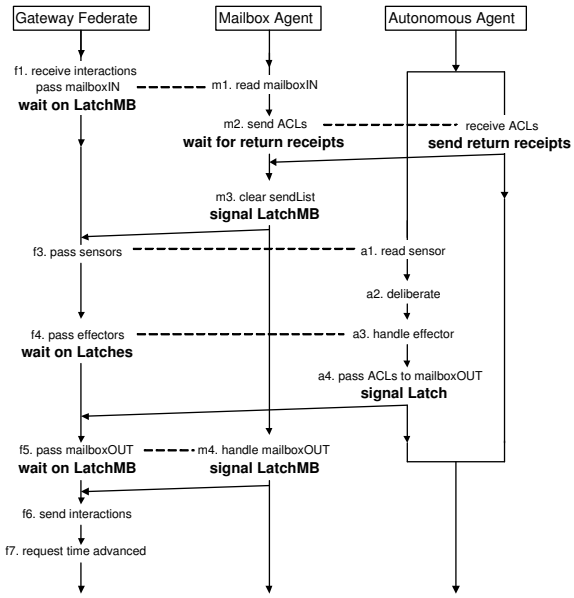

Fig. 3. Synchronization points

or make each soldier execute in a separate federate. Different federates may also be in the same machine or different machines. The federates are connected by the RTI, and they execute synchronously with each other, controlled by the HLA time management services. None of the federates can advance its local time until the other federates advance too. In other words, the present simulation is a synchronous, distributed, time-driven system where the simulation advances in time steps.

The game's main objective is to demonstrate how the architecture enables distributed agents to move in a shared environment performing tasks and to communicate with each other freely. Interaction patterns such as cooperation and collaboration can be adopted in this game too. For cooperation, soldier agents can perform tasks individually and concurrently, each one detecting mines and trying to clear them out. Their common aim is to sweep as many as mines out of the area to reach the final goal that no mine is left, although during this procedure they do not exchange information with each other. This case consists of agents simply working together and no communication exists between agents.

Since communication is a key capability which distinguishes MASs from other forms of software and provides the underlying power of the paradigm, our soldier agents have been enabled to communicate using identified templates of messages for collaboration. In the current application, soldier agents utilize ACL messages to exchange information. In our MSG policy, we allow a soldier to carry only one mine at a time. That means, after a soldier has discovered and picked up a mine, his task is to reach border to clear the mine out of the *safe area*. During this period, he is able to detect another mine, however his hands are full at that time, thus he cannot pick it up. What

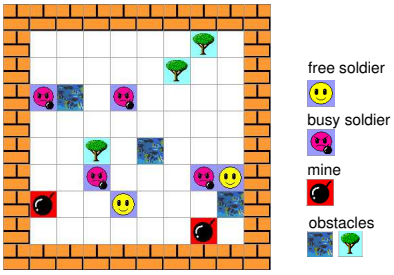

Fig. 4. Six soldiers move in a shared environment avoiding obstacles

he is able to do then is just inform the other roaming soldiers to come and clear the mine away. In this case, an ACL message carrying the information about the detected mine needs to be sent indicating the position. Obviously, to achieve better performance, the message need only be sent to those soldiers with free hands. Five kinds of ACL templates are utilized by the soldier agents (Table 2).

Table 2. ACL templates applied in the MSG

Template	Implication	State of soldier
informPosition	inform about the position of himself.	unspecified
informPick	inform that he has picked up a mine.	busy
informDiscard	inform about clearing a mine away completed.	free
informMine	inform another soldier about a detected mine's position.	busy
informAck	a return receipt to an *informMine*	free

No doubt a certain level of coordination provides a larger viewpoint for every individual agent that has only limited information about the environment, and this always brings a better working efficiency than the situation if no ACL messages and coordination are applied. An experimental result gives the same conclusion even for a small environment that needs to be cleaned up where not many soldiers are involved. We created 4 soldier agents and let them try to clear 9 mines out of a 10×10 grid environment. A 11.63% improvement was gained. This advancement increases if a larger environment is used, e.g. a 20×20 grid environment or even larger.

6 Experimental Results

We have measured the execution times where a number of ACL messages are transmitted between agents in each time step of the simulation. These are tested using three PCs connected via the network. All are of the same type of machine with 2.20GHz Pentium 4 processor and 1.00GB RAM. One of them builds a shared environment for four agents, and the other two support these agents which are located in two different gate-

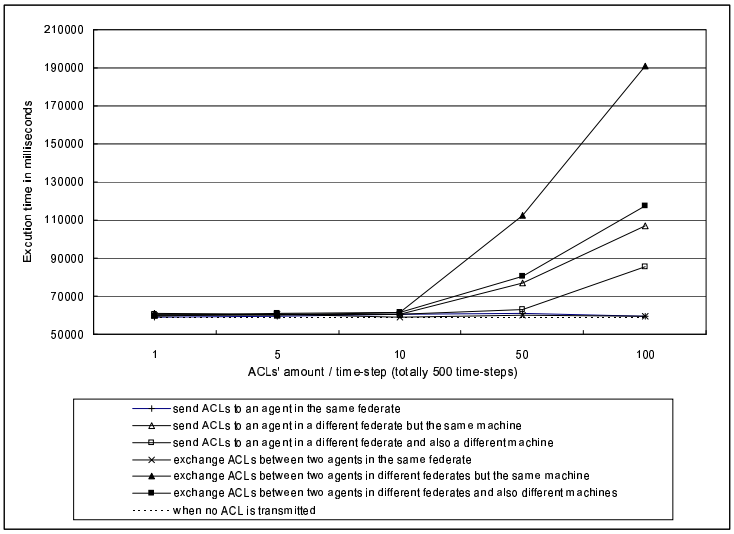

Fig. 5. ACL messages' transmission time

way federates. As shown in Figure 5, three situations are investigated for both one-way transmission and two-way transmission.

The execution time does not change much when 1, 5 or 10 ACL messages are transmitted from an agent to another or in both directions at the same time. These results remain stable and are very close to the situation when there is no message transmitted. When the number of messages reaches a large value, such as 50 and 100, the difference among the execution time of the three situations becomes distinct. It should be noted that sending messages to an agent located in the same federate always keeps a stable value of execution time. This is because the messages transmitted across federates need to be converted to/from byte sequence, before the gateway federate can send them as interactions and after receiving them. The encoding and decoding are time-consuming processes that cost computing resources. This also explains why the time of sending 50 messages to an agent in a different federate and different machine does not increase significantly compared to the case of sending messages to an agent in the same federate, while the case of sending messages to an agent in a different federate and the same machine differs a lot. For these results, a factor should be taken into account also that messages may encounter network delays when transmitted, but this is not crucial in a local area network.

In all these cases, the gateway federates and the mailbox agents work correctly, according to the conditions for consistency described in Section 3. We can see the transmission of ACL messages is also efficient unless the number of messages sent by an agent is very large, which is not a general case in practice. Actually agents do not need to send messages in each time step of the simulation. Commonly they only communicate if necessary, thus sending ten or even more messages in one time step is unlikely

for an agent in practice. However, those cases are still tested as upper bounds. They can also demonstrate that our synchronization mechanisms are quite effective.

7 Conclusions

Although there has been a lot of work on multi-agent and multi-agent based simulation, little has been done on using an MAS to represent entities in a distributed simulation. In this paper, we address the issues of how to integrate agents into an HLA-based distributed simulation with its focus on communication between agents in distinct simulators. We have proposed a general architecture for integrating an MAS in an HLA-based simulation. Its adherence to common standards can reduce the cost of development of simulation systems and facilitate the reuse and inter-operability of component simulators.

In our approach, a novel solution is adopted using agent gateway federates and mailbox agents to meet these requirements. A detailed mechanism is illustrated concerning synchronization problems between federates and agents. A prototype system has been developed to demonstrate the feasibility of this architecture.

So far, we have tested the mechanism and measured the efficiency. The experimental results show that the architecture is feasible and extensible. In the near future, we plan to investigate more scenarios to advance the proposed agent architecture, and further demonstrate its flexibility and extensibility.

References

1. Eiter, T., Mascardi, V.: Comparing Environments for Developing Software Agents. INFSYS Research Report 1843-01-02, Tech. Univ. Wien, Austria (2001)
2. Bellifemine, F., Poggi, A., Rimassa, G.: JADE – A FIPA-compliant agent framework. Proceedings of PAAM'99 (1999) 97–108
3. Andersson, J., Löf, S.: HLA as Conceptual Basis for a Multi-Agent Environment. Technical Report 8th-CGF-033, Pitch Kunskapsutveckling AB (1999)
4. Logan, B., Theodoropoulos, G.: The distributed simulation of multi-agent systems. Proceedings of IEEE **89** (2001) 174–186
5. Uhrmacher, A.M., Gugler, K.: Distributed, parallel simulation of multiple, deliberative agents. In: Proceedings of the 14th Parallel and Distributed Simulation Conferenc (PADS'2000), Bologna (2000) 101–110
6. Fujimoto, R.M.: Parallel and Distributed Simulation Systems. John Wiley & Sons, Inc., Wiley (2000)
7. Lees, M., Logan, B., Oguara, T.: Simulating Agent-Based Systems with HLA: The Case of SIM_AGENT — Part II. In: European Simulation Interoperability Workshop (Euro-SIW), UK (2003)
8. Jennings, N.R., Wooldridge, M., eds.: Agent Technology: foundations, applications, and markets. Springer. UNICOM, Berlin, Heidelberg and New York (1998)
9. Fujimoto, R.M.: Time Management in The High Level Architecture. Simulation **71** (1998) 388–400
10. Wang, F., Turner, S.J., Wang, L.: Integrating Agents into HLA-based Distributed Virtual Environments. In: 4th Workshop on Agent-Based Simulation (ABS2003), Montpellier, France (2003) 9–14