Advanced Engineering Forum *Online: 2011-09-09*
ISSN: 2234-991X, Vol. 1, pp 16-20
doi:10.4028/www.scientific.net/AEF.1.16

Cooperation design system based on Mobile-C agent platform

Bo Yu[1,a], Zixian Zhang[1], Yixiong Feng[2], Luis Ariel Diago[1], Ichiro Hagiwara[1,b]

[1]Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku, Tokyo, Japan

[2]State Key Lab of Fluid Power Transmission and Control, Zhejiang University, Hangzhou 310027, China

[a]yu.b.aa@m.titech.ac.jp, [b]hagiwara.i.aa@m.titech.ac.jp

Keywords: Agent, Cooperation design, Multi-agent system, Mobile-C.

Abstract. Over the past decades, Distributed Systems (DS) have been adopted for industrial applications to improve the system efficiency because distributed architecture has advantages in resource utilization, fault toleration .etc. Multi-Agent System (MAS) arises from combination of the theories of artificial intelligence and distributed systems. One character of MAS is their self-organization, so how to implement an effective mechanism for self-organization of agents is important to a MAS system, this paper describes the design and implementation of a Mobile-C based agent management system, in which Mobile-C was adopted as the implementation platform, and this paper also described an agent-based cooperative design application using this system to manage all the agents involved.

Intorduction

The concept of Multi-Agent Systems (MAS) arises from combination of the theories of artificial intelligence and distributed systems. It is widely used in variety of applications, such as collaboration of multi-robotic systems [1], detection system for identification of vehicles on the highway [2].etc.

The term ‘mobile agent’ was first introduced by Telescript, it is a piece of programs that can migrate from host to host in a network, at times and to places of their own choosing. The state of the running program is saved, transported to the new host. Mobile agent is an effective choice for many applications for several reasons, including improvements in latency and bandwidth of client-server applications and reducing vulnerability to network disconnection.

Currently, most of the mobile agent systems were developed to support only Java mobile agents, such as JADE [3], Repast .etc. Furthermore, many of them are standalone platforms. In other words, they were not designed to be embedded in a user application to support code mobility. Mobile-C is an agent platform for supporting C/C++ mobile agents in networked intelligent mechatronic and embedded systems, it is an IEEE the Foundation for Intelligent Physical Agents (FIPA) compliant agent platform developed by Harry H. Cheng [2] from University of California-Davis, and their system has been tested on the highway with real traffic and the results indicate that this methodology can achieve high accuracy for vehicle identification. As they mentioned, Mobile-C is designed for real time and resource constrained applications with interface to hardware.

One character of MAS is their self-organization. However, there did not exist an effective mechanism to manage all the agents involved in the MAS, and all the agents are in a situation of disordered, efficiency can not be guarantee because of the time and resources wasted in finding another agent to execute, so all the agents should be arranged in a statement which is ordered so the system can get a whole view if all the resources have been utilized effectively, and make job distribution more effectively.

This paper implemented a mechanism to manage the agents involved in a Mobile-C based multi-agent system, after using this management mechanism in the multi-agent system. Execution path of a particular job has been managed effectively.

System Structure

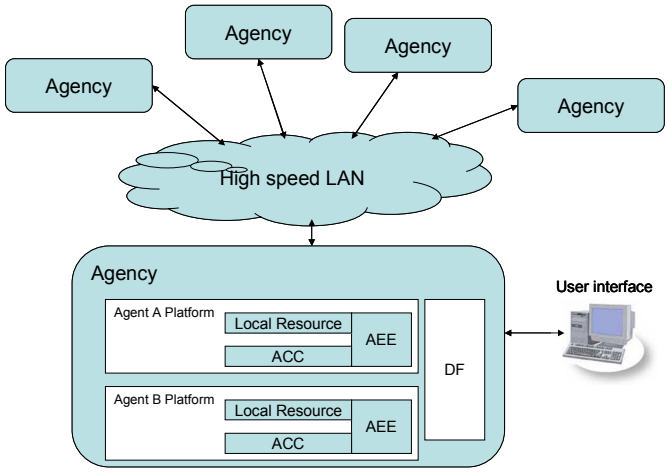


Fig. 1 Inside system structure of a MAS

Fig. 1 represents the system structure of our system, each agency represents an execution node for Mobile-C agent, and they are connected via high speed Local Area Network (LAN), an agent platform represents the minimal functionality required by an agency in order to support the execution of agents.

In this research, an agent platform is developed with the following functions.

- Directory Facility (DF): the yellow page of the multi-agent system.
- Agent Communication Channel (ACC): message router in an agent platform.
- Agent Execute Engine (AEE): execution environment for the mobile agents.
- Local Resource: the data, database and executable program an agency possessed that can be invoked by other agents.

Agent Management System (AMS) is responsible for managing agents, including creation, registration and migration. It manages the life cycle of agents. Once an agent is created by AMS, the information of agent ID, agent address and agent creator will register to DF. AMS provides an interface to users, through which they can encapsulate their command to executable agent code. AMS also plays a role as agent dispatcher, which means AMS selects an agent that is ready to be dispatched, and send to the destination according to DF from multi-agent system. AMS should know the mission or role of the agent after the agent code was generated.

Key technologies to implement AMS

ACL message generation Method. Instead of writing programs all by the users, agent management system embeds particular agent code generation mechanism, so all the user has to do is to provide their requirements and submit them to multi-agent system, then wait for the results. The message include agent code is called agent communication language (ACL), which is illustrated in Fig. 2.

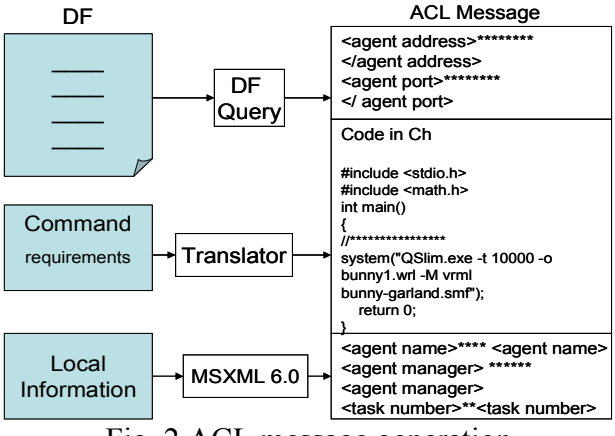


Fig. 2 ACL message generation

Microsoft Core XML Services (MSXML) is an XML encapsulate and decode method, it is adopted as the ACL message generation class, MSXML is W3C Conformance and System.Xml 2.0 Compatibility, which means MSXML6 has gone through extensive testing and a number of issues have been addressed to improve W3C conformance and System.Xml 2.0 compatibility particularly in terms of the XML Schema 1.0 Recommendation.

ACL message generated in this system can be recognized by other agent platform because of it is strictly comply with FIPA protocol. So agent created by the MAS can easily communicate with other FIPA compatibility agent platforms.

Multi-agent execution monitor. The concept of TRIGGER [4] is adopted in this paper, TRIGGER is a useful tool in commercial database, like SQL Server from Microsoft, DB2 from IBM. A database trigger is procedural code that is automatically executed in response to certain events on a particular table in a database. TRIGGER can restrict access to specific data, perform logging, or audit data modifications.

Fig. 3 shows a typical trigger work process including three agent threads. Agent 1 is created by agent management system, then it will be send to some agency and execute its job, after agent 1 finishes its job, it will send a message to trigger in multi-agent system, the trigger judges the state of agent 1, and makes a decision weather to create another agent, with a positive answer, the trigger will look up the agent pool, and create another agent thread 2 which listening on the execution state of agent 2. Agent thread 2 will send the execution results to the trigger, and then the trigger creates thread 3 to continue the job, after Agent 3 finishes its job, AMS judges the whole job, and kills all the existing agent thread if the job is finished.

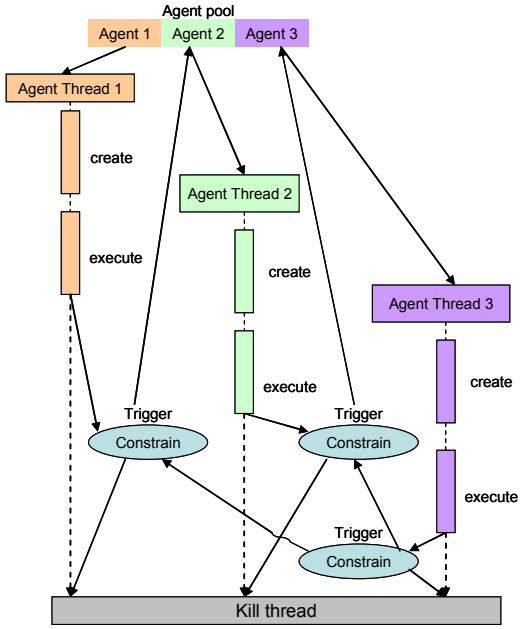


Fig. 3 Event-Trigger work process

We use AMS which include Event-Trigger as the agent monitor in our multi-agent system. For each agent, AMS creates a thread waiting to be triggered, the trigger waited for a signal such as agent finished its job or there are some exceptional. Then a function combined with that trigger will be executed.

Case study

A prototype of AMS is developed based on Mobile-C platform, and then we integrated the AMS to our multi-agent system, designer cooperated with shop-floor workers in this MAS to do a model simplification job, the initial model is showed in Fig. 4.

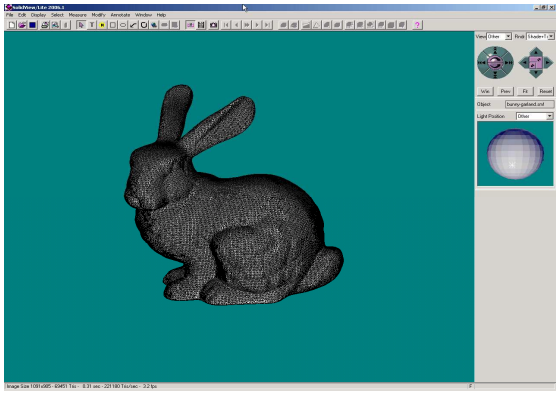
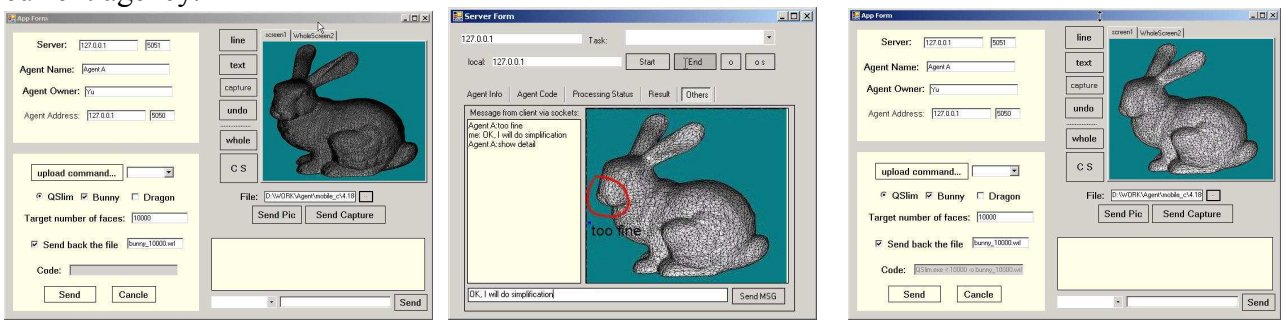


Fig.4 Initial model of a bunny

Their interface is showed in Fig. 5.A and 5.B, requirements and commands from the designers dispatched to multi-agent system in a FIPA ACL message format using the interface provided by AMS. On the other hand, shop-floor workers (Fig 5.B) register their service to DF, and waiting for an agent to come, once an agent comes, the agent will execute automatically, after agent finishing their job on that agency (Fig 5.C) , it will move to next destination with the results it gets from current agency.



(a) Main frame of designer (b) Main frame of shop-floor workers (c) After execution

Fig. 5 Main window

After the simplification job is finished, as we can see the model in Fig. 6, the last execution agent will send a message to the Trigger in AMS, the Trigger will judge the whole job condition and make a decision to continue the execution if the job is not satisfied, or else destroy all the active agent thread existed in the multi-agent system for the situation that all the job have been finished.

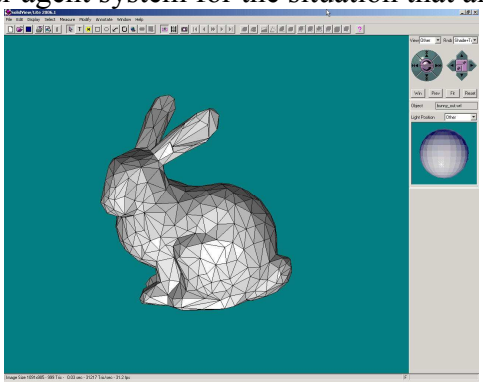


Fig. 6 Model after simplification

Conclusion

After the investigation of MAS existed nowadays, and considering the important role the AMS plays in a multi-agent system, a prototype of AMS is developed based on Mobile-C platform, and then we integrated the AMS to our multi-agent system based on Mobile-C platform. Key components have been discussed and implemented, and at last the AMS is integrated to our MAS based on Mobile-C. The system we developed works well in a cooperative design system, and we verified the validity of our design.

Acknowledgement

Several parts of the research work were carried out by the support of Grants-in-Aid for scientific research (category S) under Grant No.20226006.We acknowledge its aid dearly.

Reference

[1] K.G. Jolly, K.P. Ravindran, R. Vijayakumar, R. Sreerarna Kumar, Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks. Robotics and Autonomous Systems. 55 (2007) 589–596.

[2] Harry H. Cheng, Benjamin D. Shaw, Joe Palen, Bin Lin, Bo Chen and Zhaoqing Wang, Development and Field Test of a Laser-Based Nonintrusive Detection System for Identification of Vehicles on the Highway. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 6, NO. 2, (2005) 147-155.

[3] Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood, Developing Multi-Agent Systems with JADE, (2007) Wiley Series in Agent Technology.

[4] Dennis McCarthy, Umeshwar Dayal, The architecture of an active database management system, ACM SIGMOD Record, Volume 18, Issue 2, (1989) 215 – 224.