Research article

A CORBA-based cooperative mobile robot system

Zhen Zhang, Qixin Cao, Lei Zhang and Charles Lo
School of Mechanical Engineering, Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai, People's Republic of China

Abstract
Purpose – The purpose of this paper is to present a distributed multiple mobile robot system that provides a collaborative control and simulation environment.
Design/methodology/approach – A CORBA-based cooperative system is designed to implement a robotic layered cooperative mechanism. The mechanism has three layers: mission, transport and execution. In order to realize a flexible and effective communication in the cooperative mechanism, an extended robot event service (federated event service) is proposed to improve the cooperative system's real time performance.
Findings – Experimentation has proved the validity and effectiveness of the system. The federated event service's latency is approximately 9 percent less than the standard event service latency when the CPU is determined.
Practical implications – The robotic modularized system includes the map-building, path-planning, robot task-planning, simulation and actual robot control function modules, and uses CORBA to integrate the whole system. It is easy to implement a layered cooperative mechanism for multiple mobile robots. Given the problem on multiple robots cooperation latency, a useful extended robot event service is proposed.
Originality/value – The paper focuses on the distributed functional modular architecture, and the multiple robots cooperative layered mechanism. In the mechanism, an extended robot event service (federated event service) is proposed to reduce the cooperative system's real time latency. The conducted experiment validates the proposed system with a good performance for multiple mobile robots' cooperation.

Keywords Robotics, Mechanics, Systems and control theory

Paper type Research paper

1. Introduction

Multiple mobile robotic systems are becoming more useful in many practical applications. Nuclear power plants, space or undersea exploration, and manufacturing systems are typical application domains. These systems must be reliable and coordinated while finishing different subtasks such as perception, planning and navigation. Thus, some robotic platforms have been built to test control or coordination algorithms, and evaluate the system performances. The Player/Stage/Gazebo project is an open architecture to provide robot control and simulation in 2D and 3D environment (Gerkey *et al.*, 2001, 2003). Breve is a simulation environment meant for the development of artificial life in a physically simulated world. It uses a scripting language that allows control strategies and event-based reactions to the environment for large numbers of robots (Klein, 2002). CARMEN (Montemerlo *et al.*, 2003) uses the middleware framework Mobile and Autonomous Robot Integrated Environment (Côté *et al.*, 2006) to build the mobile robots' control and simulation system. However, there is still not an integrated platform which supports customized environment modeling, graphical programming, distributed multiple robotic cooperation functions. Songmin proposes an internet-based robotic system that uses CORBA to implement networking connections. To cope with time delays on the communication, a robot control server is implemented to allow the user to control the tele-robotic system at a task-level (Jia and Kunikatsu, 2002). But it is designed for a single remote robot, and not suitable for multiple remote robots' communication. Thus, many robot systems use the CORBA event service to realize the robots' communication (Brooks *et al.*, 2006; Colon *et al.*, 2006; Kuo and MacDonald, 2005). However, the standard CORBA event service specification is limited to a single processor configuration. If a single centralized robot event channel is used for all of the sub-networks, extra communication overhead and latency will incur when the event channel is on a remote personal computer. The communication between robots in the same sub-network will cost less overhead and latency than the communication between robots in different sub-networks. The standard CORBA event service has not discussed this problem.

The current issue and full text archive of this journal is available at **www.emeraldinsight.com/0143-991X.htm**

Industrial Robot: An International Journal
36/1 (2009) 36–44

[DOI 10.1108/01439910910924657]

This work is supported in part by the National High Technology Research and Development Program of China under grants 2006AA04Z261 and 2007AA041700; the authors gratefully acknowledge YASKAWA Electric Cooperation for supporting the collaborative research funds and the SmartPal robot and thank Mr Ikuo Nagamatsu and Mr Kazuhiko Yokoyama at YASKAWA for their cooperation.

Considering the above inconveniences, a distributed cooperative mobile robot system based on CORBA is developed. Multiple normal PCs work together to realize the system functions, such as environment model building, task planning, simulation and actual robot control. A layered cooperative mechanism is proposed to simplify multiple robots cooperation. In order to realize a flexible and effective communication in the cooperative mechanism, an extended robot event service (federated event service) is proposed to provide an optimal correlation for both local and remote robots.

The remainder of the paper is organized as follows: Section 2 introduces a new distributed simulation and control system architecture. The proposed multiple robot cooperative mechanism is proposed in Section 3. Section 4 describes the results of the cooperation experiment in a simulated environment. Finally, the conclusion is given in Section 5.

2. System architecture

The architecture of the distributed system is shown in Figure 1. It includes three functional parts: a mapEditor server, a simulation server and some robot clients. The mapEditor server enables building of an indoor simulation environment model and path planning services. The simulation server provides the virtual sensors and simulation services. Its Omni database keeps records of all the information of the simulation. The robot client is the client which invokes the methods from the CORBA server and presents the 3D simulation result to the users.

As a result of using CORBA, the service implemented object in these servers can be remotely but transparently invoked from the clients regardless of their hardware, operating systems, and programming languages (Object Management Group, 1999; Henning and Vinoski, 1999; Object Management Group, 2005). All of the servers and clients can be distributed on different computers using CORBA middleware the Java™ IDL developed by Sun Microsystems (Sun Microsystems Inc., 2004). Java™ IDL is a freely available and fully compliant implementation of the CORBA standard. It provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems. Each module of the system is presented as follows.

2.1 MapEditor server

The mapEditor server is used to build a unique virtual environment for multiple robots. The server contains two agents: a map building agent and a path planning agent. The user can build a customized environment according to a real world via a map building agent, save the objects' shapes, positions and other geometrical data in a geometrical map, and save the path nodes in a topological map (Figure 2).

The path planning agent is used to determine a feasible path between the start and end points specified by the user. Figure 3 shows a communication example between the robot client and the path planning agent server. In Figure 3(a), the client specifies the start point and the end point, calls the "getPath" method using the CORBA interface, and receives a set of path points from path planning agent server. Figure 3(b) shows the map topological information. Figure 3(c) shows the format of path points returned by the server.

Figure 1 System architecture

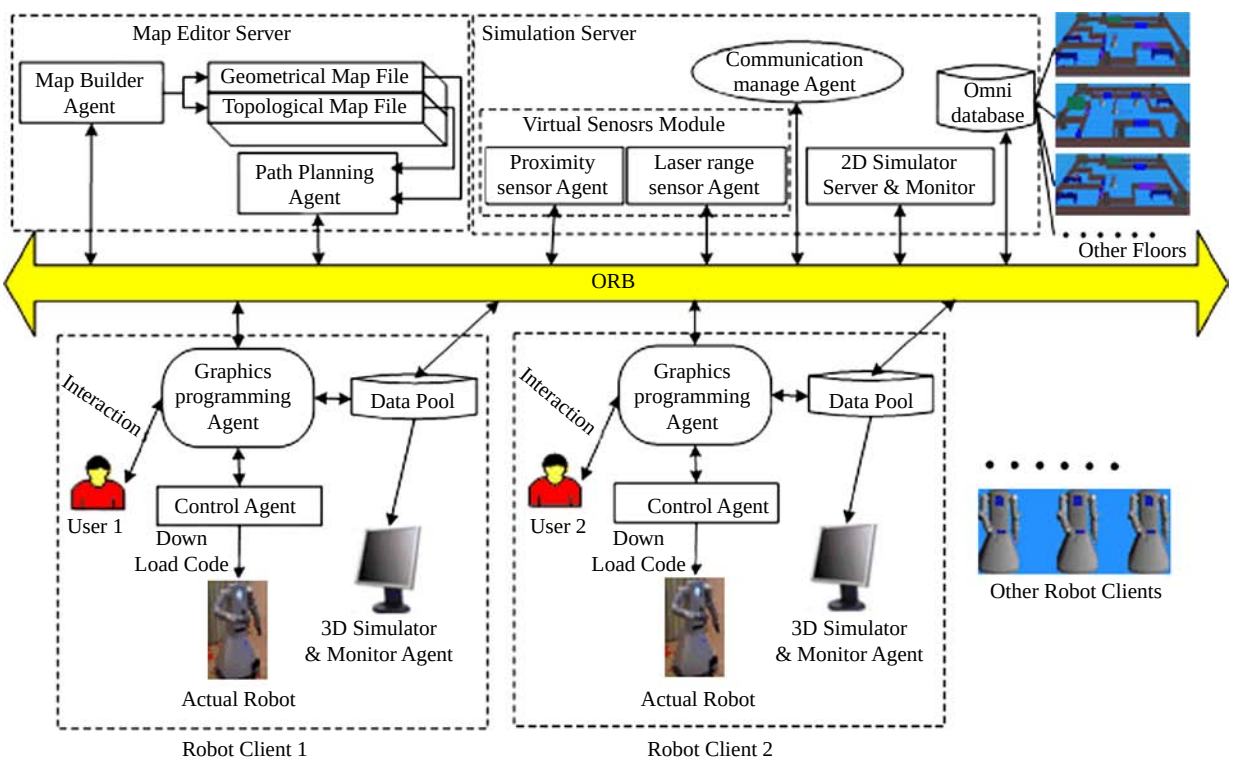
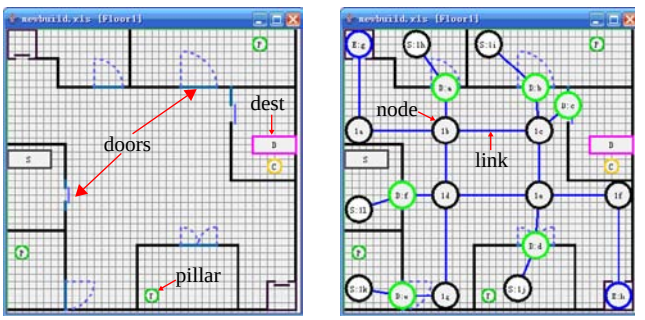


Figure 2 Two maps in a mapEditor: (a) geometrical map; (b) topological map



(a) (b)

2.2 Simulation server

In the simulation Server, the communication management agent is responsible for recording all the registered information from the clients and to realize a simple load balance. All robot clients wanting to join in the simulation must first register with this agent. Only the management agent accepts the client's request and puts the client's name in the register list, the robot client can call the methods using the CORBA interface. If the number of clients reaches the upper limit of permissions, the communication management agent will not allow anymore clients to log on to the server.

All the simulation information is stored in the Omni database. It is structured in an elevator data list and a floor data list. This data structure makes it easier to exchange the single floor's data between the Omni database (simulator server) and the data pool of a robot client.

The 2D simulator server loads the data from the Omni database, then draws the simulation scene in 2D image. This tool is capable of displaying video of the simulation scene, and allowing the administrator to monitor the simulation process in the whole building. In the monitor panel, users can observe different floor scenes, see the robots' positions, velocities and receive data from the virtual sensor.

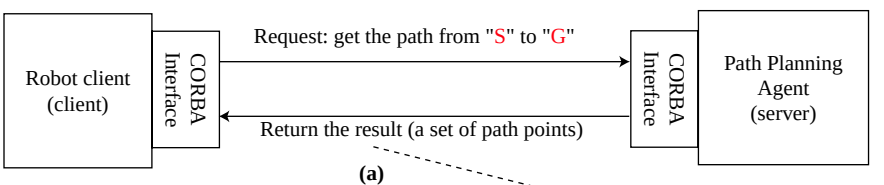
The virtual sensor server provides virtual laser range finder and proximity sensor agent's services. It receives the robot client's request with parameters such as the position of the robot, the position and direction of the virtual sensor, and returns laser scan angle, distance information. The information is similar with the information returned by a real sensor, and used by the virtual robot to detect the positions of obstacles.

2.3 Robot client

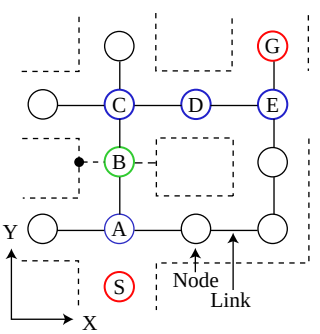
The robot agent is a development module for users to program, control and observe the virtual robot in a simulated environment or an actual robot in the real world. It is structured in the CORBA client that calls methods from different servers. One robot agent stands for one virtual or actual robot. The components in a robot agent include:

- *Graphic programming agent (GPA).* It is used to specify the robot's task by using a list of icons. One icon describes one movement of the robot and the user can tell the robot to finish those movements by editing the related icons' parameters. For example, edit a "Mbase" icon to control an omni-directional wheel platform for the motion, a "Mhand" icon to control the related arm's status (Figure 4), a "gripper" icon to control the robot's gripper, etc.
- *Control agent.* There are two modes: virtual robot mode and actual robot mode. In the virtual mode, the agent calls the SmartPal robot's virtual function library (R&D Center YASKAWA Corporation, 2004) to calculate each link position of the robot's actuators, and return results to the GPA. Then the GPA sends the robot's data to the data pool for 3D simulation in the robot client. In the actual robot mode, the agent can upload the codes to an actual robot to execute the related task.
- *Data pool.* It is used to save simulation data related to the current robot. For example, if a virtual robot is on the second floor of the simulation environment, its data pool only keeps the objects' information on the second floor.
- *3D simulator and monitor agent.* It loads the simulation data from the data pool, and constructs the robot in the 3D

Figure 3 Path planning communication: (a) communication process; (b) metric and topological identifier; (c) data structure



(a)

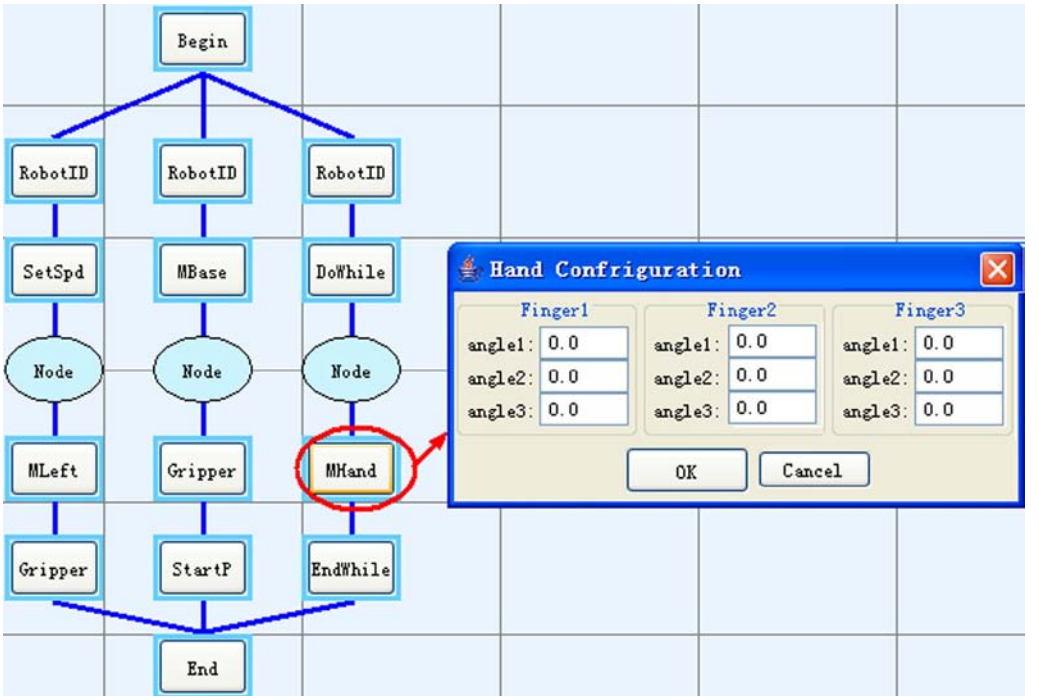


(b)

No.	Node	(x,y)	attributes
1	S	(X_S, Y_S)	station
2	A	(X_a, Y_a)	viapoint
3	B	(X_b, Y_b)	door
4	C	(X_c, Y_c)	viapoint
5	D	(X_d, Y_d)	viapoint
6	E	(X_e, Y_e)	viapoint
7	G	(X_g, Y_g)	station

(c)

Figure 4 The graphic programming agent



virtual environment. When the simulation result is verified, users can upload the codes to a real robot for execution. The cooperative robotic system has been realized on the dual-arm mobile robot SmartPal. By far it is mainly used as a service robot. Figure 5 shows a SmartPal used for experiments in an indoor environment. It is equipped with a pair of arms that have 7 degrees of freedom. An omni-directional wheel platform is used for planar motion. The sensors equipped in the robot are a laser range finder mounted on the waist and eight proximity sensors mounted around the omni-directional wheel platform. The liquid crystal display and touch panel is attached to the front of the robot's upper body. It is used for displaying the robot's current internal state (e.g. working, waiting and exceptional) or for user interaction (e.g. choice of the work model).

3. Multiple robot cooperative mechanism

In order to realize multiple robot cooperation, we propose a layered cooperative mechanism which is shown in Figure 6. There are three layers: mission, transport and execution layers.

3.1 Mission layer

The mission layer is responsible for describing a robot mission which uses a list of icons to express the related robot's action sequence in a robot leader's GPA. The GPA generates the icon list through an interaction with users, and analyses all of the icons' attributes. If there are cooperative icons such as moving a table, the robot becomes a leader. Then the mission is refined and divided into a number of SubPlans. The cooperative plan is a combination of all cooperative

Figure 5 SmartPal robot

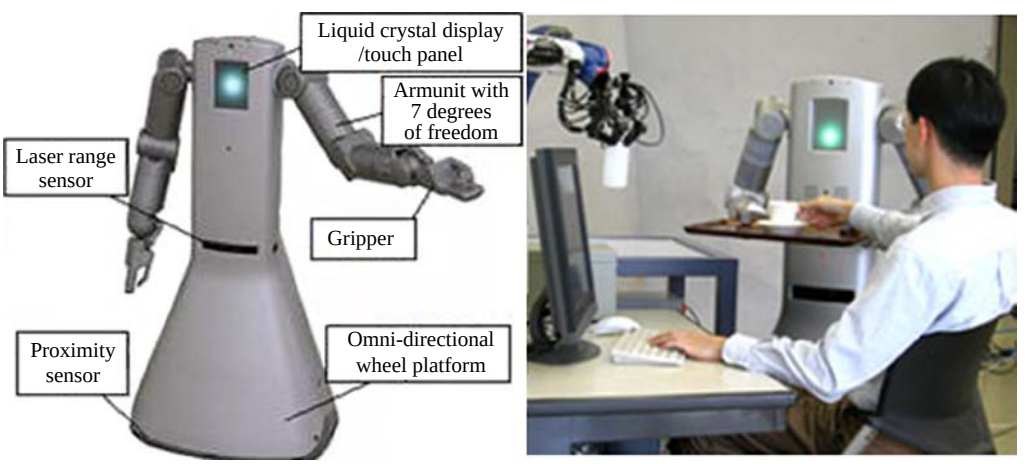
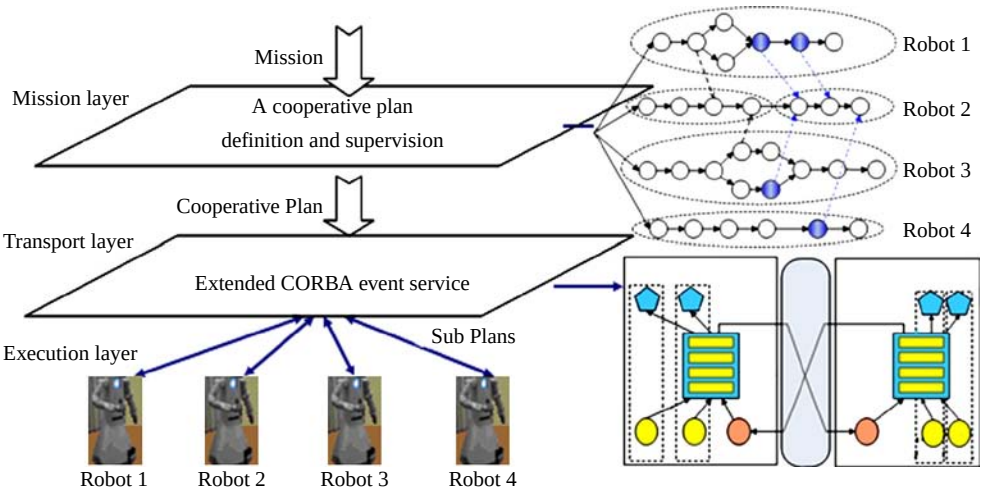


Figure 6 Multiple robot cooperative mechanism



SubPlans, and is stored in the leader's data pool for other robots to read. In the example shown in Figure 6, the robot 2 is the cooperative leader, and will receive robot 1, 3, and 4's coordinated SubPlans.

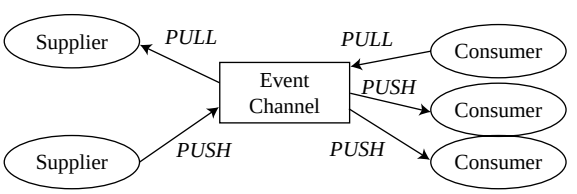
3.2 Transport layer

On the transport layer, the global cooperative plan is divided into SubPlans for different robots to execute. In order to realize a flexible and effective communication, the requests are sent to the related robots by the CORBA event service (Object Management Group, 2004). Figure 7 shows the key participants in the CORBA event service architecture. The role of each participant is outlined as follows:

- *Consumers and suppliers.* Consumers are the ultimate targets of events generated by suppliers. Consumers and suppliers can both play active and passive roles. An active push supplier pushes an event to a passive push consumer. Likewise, a passive pull supplier waits for an active pull consumer to pull an event from it.
- *Event channel.* At the heart of the COS event service is the event channel, which plays the role of a mediator between consumers and suppliers. The event channel manages object references to suppliers and consumers. It appears as a proxy consumer to the real suppliers on one side of the channel and as a proxy supplier to the real consumers on the other side.

The CORBA event service model simplifies application software by allowing decoupled suppliers and consumers, asynchronous event delivery, and distributed group communication. Thus, many robot systems use the CORBA event service to transport messages (Brooks *et al.*, 2006; Colon *et al.*, 2006; Kuo and MacDonald, 2005). However, the standard CORBA event service specification (Figure 8) is limited to a single processor configuration. If we configure a single centralized robot event channel for all of the sub-networks, extra communication overhead and latency will incur when the event channel is on a remote personal computer. The communication between robots in the same sub-network will cost less overhead and latency than the communication between robots in different sub-networks. To alleviate the limitations, we have developed an extended robots event service (federated event service) which provides mechanisms to connect several event channels to form a federation, as shown in Figure 9. In the federated architecture, multiple event channels are connected through a gateway. A gateway is a servant that connects to one event channel as a consumer and forwards all events it receives to other event channels. It therefore plays the role of supplier for the second event channel. The gateway usually connects as a supplier to a collocated event channel. Thus, the remote communication mechanisms are used only when events are sent from a remote supplier to a local consumer. A gateway can be collocated with the event channel that supplies events to it. The configuration shown in Figure 9 minimizes network traffic since the gateway only subscribes to events of interest to its consuming event channel. The gateway must subscribe to the disjunction of all the subscriptions in its consuming event channel since it is possible that a correlation is satisfied from events arriving from both local and remote consumers.

Figure 7 CORBA event service architecture



The current event service implementation requires that each event channel in a federation be connected to every one of its peers. Thus, distributed interactive simulations can comprise several nodes in different sub-networks, where most of the traffic destination is within the same network. Configuring an event channel on each network helps to reduce latency by avoiding round-trip delay to remote nodes.

3.3 Executing layer

After all the SubPlans have been assigned and transported successfully, all of the related robots will execute their work synchronously, and return the executive result to the leader robot. The robots in an executing layer can be applied to both virtual and physical robots. If the user assigns the robot a new job and satisfies the performance in simulation, the code can

Figure 8 Standard CORBA event service

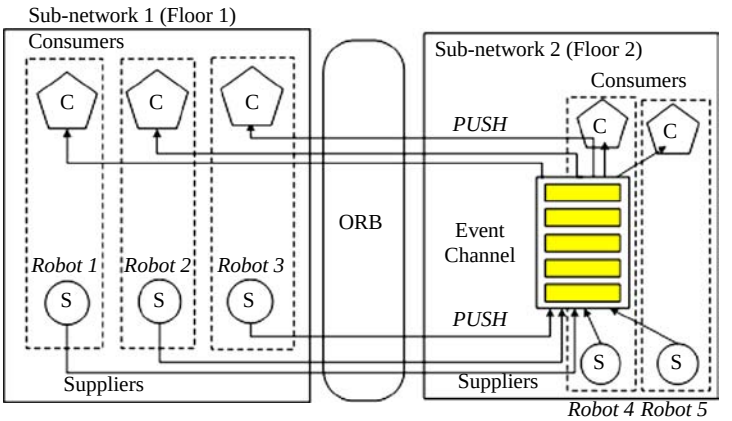
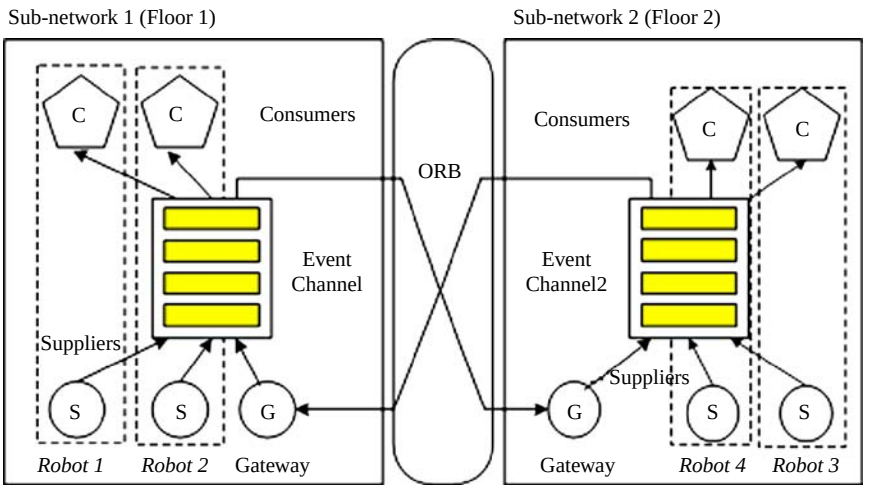


Figure 9 Federated CORBA event service



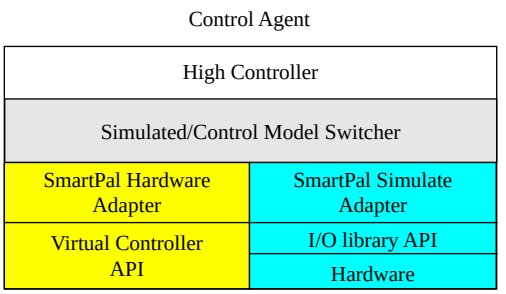
be uploaded to real robots at once. The related architecture is shown in Figure 10.

The control agent uses a model switcher to set it to either a simulated or a real robot control model. The unification of a controller is realized by introducing the adapter whose API is the abstracted controller API. The simulate adapter is layered on the controller API of the virtual controller which reads the outputs of the low APIs(), and return to graphic programming environment; the hardware adapter is put on the API of I/O board of SmartPal. The API of the adapter is identical, as follows:

```
class robot_adaptor
{
  public:
    virtual boolean start(int argc, string argv[]);
    virtual boolean stop();
    virtual boolean read(robot_state st);
    virtual boolean write(actor_command ac);
}
```

So the program code can both be implemented in virtual robots and actual robots. In Figure 11, the system programs a virtual robot in the simulation environment; the code is loaded to control a real robot to hand over a piece of paper.

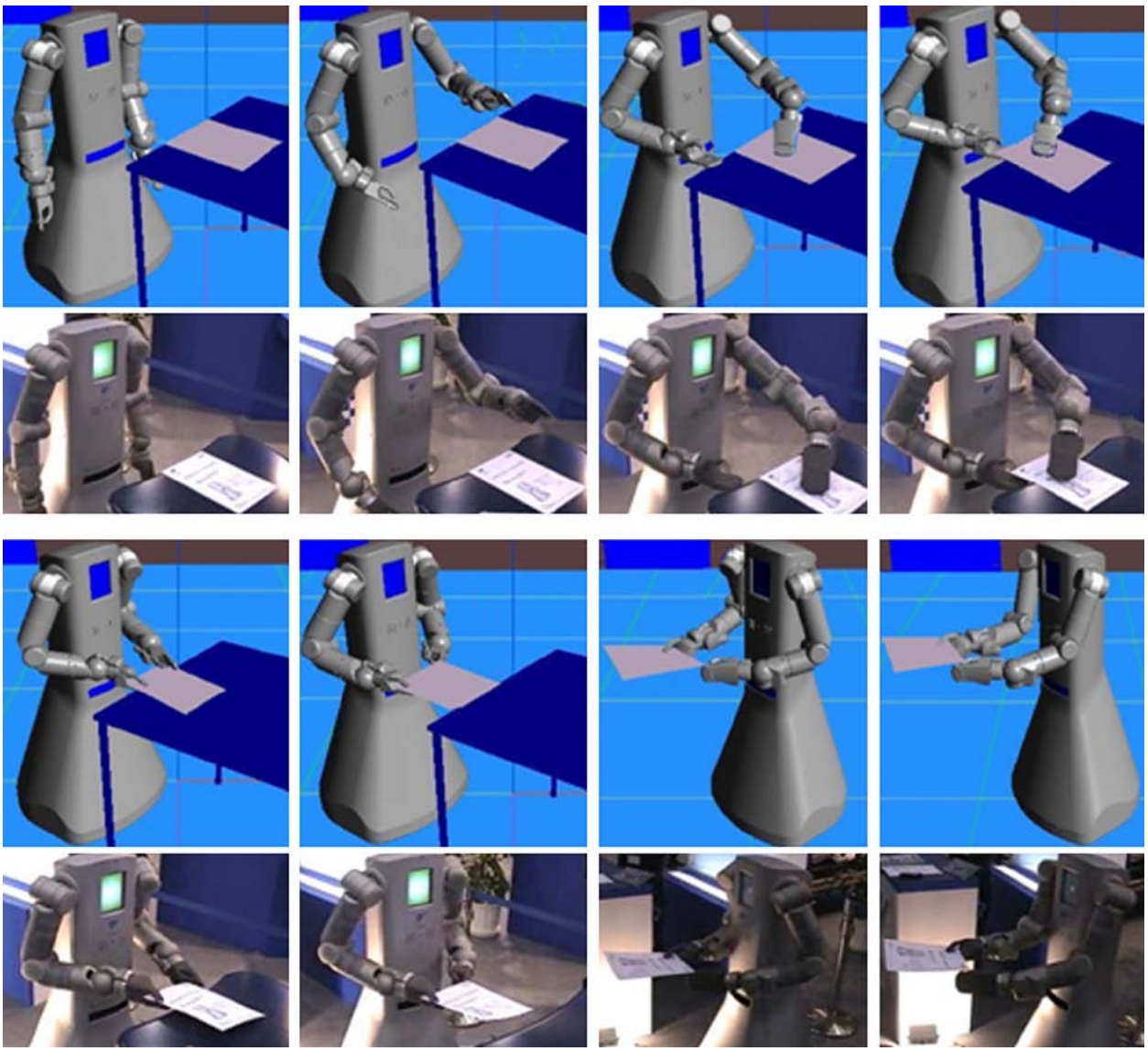
Figure 10 The architecture of migration between simulated and actual robots



4. Simulation and result

In order to evaluate the proposed system performance, we design a "move table" simulation. There are three SmartPal virtual robots in our system working together to move the table to destination desired location.

Figure 11 Seamless migration between simulated and real robots



4.1 Hardware

The computer used for the simulation is a Celeron (R) CPU 2.40 GHz and 1,230 M usable memory (512 M physical memory and 718 M virtual memory). The operating system is Microsoft Windows XP Professional with Service Pack 2. The computers are interconnected via a fast Ethernet (10 Base T) for control-communication.

In the distributed system, the mapEditor server and the simulation server are installed on a computer that provides the system services. Three virtual robots are running on three other separate computers.

4.2 Simulation process

In Figure 12(a), robots A and B work in the same room, and they are on a subnet I whose gateway address is 192.168.0.1. Robot C is in another room and it is on a subnet II whose gateway address is 192.168.8.1. As shown in Figure 12(b), robot A receives the user's command to move the blue table. It then becomes a leader robot, and sends the SubPlans to robots B and C. Robot C enters the room, together with robot B, helps the leader robot A to move the table. The scene is shown in Figure 12(c). In Figure 12(d), they move the table to the destination successfully.

4.3 Results and analysis

The latency is an import metric of the event service performance, and we keep records of the latency value for ten times in a standard CORBA event service model and our federated event service model, and calculate the final average value. The results are shown in Figure 13. With the leader robot CPU's utilization increasing, the latency decreases. In order to improve the performance of CORBA event service, the federated robots event service connects several event channels to form a federation. Multiple event channels are connected through a gateway which is a servant connecting to one event channel as a consumer and forwards all events received to other event channels. So the federated robots event service can provide an optimal correlation for both local and remote robots,

Figure 12 Scenes of robots cooperative simulation

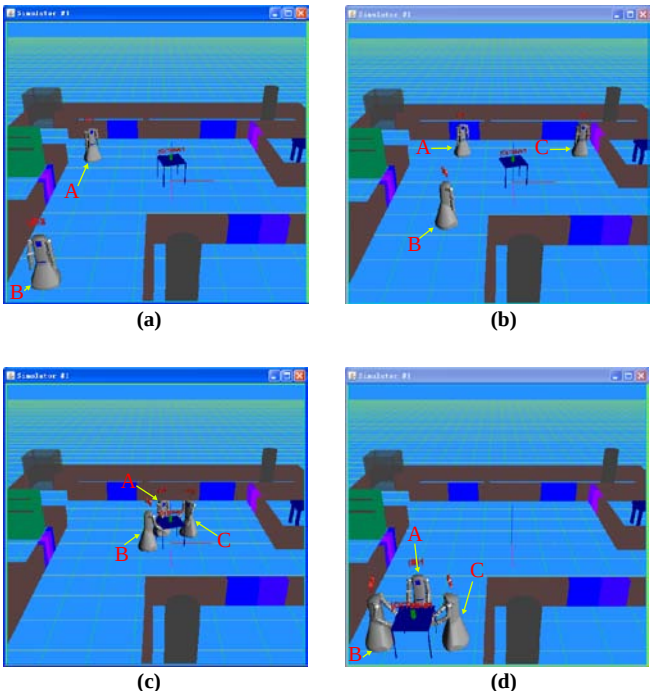
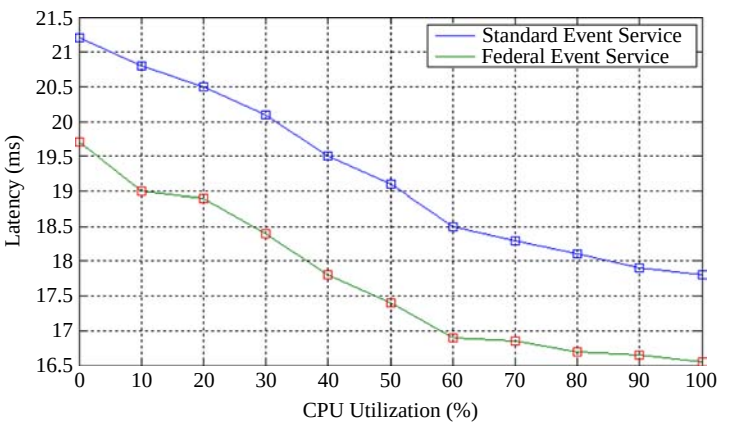


Figure 13 Latency of different CORBA event services



the remote communication mechanisms are used only when events are sent from a remote supplier to a local consumer. The experiment result shows that the federated event service's latency is approximately 9 percent less than the standard event service latency when the CPU is determined. For example, in Figure 13, when the CPU's utilization is 60 percent, our federated event service average latency is 16.9 ms, while the standard event service latency is 18.5 ms. The result shows that the federated CORBA event service has better real-time performance than the traditional CORBA event service.

5. Conclusions

In this paper, a distributed multiple mobile robot system is proposed to realize a collaborative control and simulation environment. The robotic modularized system includes the map building, path planning, task planning, simulation and actual robot control function modules, and use CORBA to integrate the whole system. It is easy to implement a layed cooperative mechanism for multiple mobile robots. Given the problem on multiple robots cooperation latency, a useful extended robots event service (federated event service) is proposed. The federated event service provides mechanisms to connect several event channels to form a federation and provides an optimal correlation for both local and remote robots to improve the cooperative system's real time performance. An experiment is conducted to validate the proposed system resulting with a good performance for multiple mobile robots' cooperation.

References

Brooks, A., Kaupp, T., Makarenko, A., Williams, S.B. and Oreback, A. (2006), "Orca: a component model and repository", *Principles and Practice of Software Development in Robotics*, Springer, New York, NY.

Colon, E., Sahli, H. and Baudoin, Y. (2006), "CoRoBa, a multi mobile robot control and simulation framework", *International Journal on Advanced Robotics*, Vol. 3 No. 1, pp. 73-8.

Côté, C., Brosseau, Y., Létourneau, D., Raïevsky, C. and Michaud, F. (2006), "Robotic software integration using MARIE", *International Journal of Advanced Robotic Systems*, Vol. 3 No. 1, Special Issue on Software Development and Integration in Robotics, pp. 55-60.

Gerkey, B.P., Vaughan, R.T. and Howard, A. (2003), "The player/stage project: tools for multi-robot and distributed sensor systems", *Proceedings of the International Conference on Advanced Robotics (ICAR 2003), Coimbra, Portugal, June 30-July 3*, pp. 317-23.

Gerkey, B.P., Vaughan, R.T., Støy, K., Howard, A., Sukhatme, G.S. and Mataric, M.J. (2001), "Most valuable player: a robot device server for distributed control", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), Wailea, Hawaii, October 29-November 3*, pp. 1226-31.

Henning, M. and Vinoski, S. (1999), *Advanced CORBA Programming with C++*, Addison-Wesley, Reading, MA.

Jia, S. and Kunikatsu, T. (2002), "Internet-based robotic system using CORBA as communication architecture", *Journal of Intelligent and Robotic Systems*, Vol. 34, pp. 121-34.

Klein, J. (2002), "BREVE: a 3D environment for the simulation of decentralized systems and artificial life", *Proceedings of Artificial Life VIII, 8th International Conference on the Simulation and Synthesis of Living Systems*, MIT Press, Cambridge, pp. 329-34.

Kuo, Y-H. and MacDonald, B.A. (2005), "A distributed real-time software framework for robotic applications", *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, April*, pp. 1964-9.

Montemerlo, M., Roy, N. and Thrun, S. (2003), "Perspectives on standardization in mobile robot programming: the Carnegie Mellon Navigation (CARMEN) toolkit", *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Las Vegas, NV, October 28-30, pp. 2436-41.

Object Management Group (1999), White Paper on Benchmarking, Version 1.0, OMG Document Bench, Object Management Group, Needham, MA, December 1.

Object Management Group (2004), *Event Service Specification, Version 1.2*, OMG Document Bench, Object Management Group, Needham, MA, October 2.

Object Management Group (2005), *OMG Robotics Domain Special Interesting Group (DSIG) Homepage*, Object Management Group, Needham, MA, available at: http://robotics.omg.org

R&D Center YASKAWA Corporation (2004), *Instructions for RTLab API (Ver 1.1.2)*, Yaskawa Robotics Technology R&D Department, R&D Center YASKAWA Corporation, Kitakyushu.

Sun Microsystems Inc. (2004), *Java IDL and RMI-IIOP Tools*, Sun Microsystems Inc., Santa Clara, CA, available at: http://java.sun.com/j2se/1.5.0/docs/tooldocs/index.html#idl

Further reading

Khamis, A., Abdel-Rahrnan, A. and Kamel, M. (2006), "A distributed architecture for mobile multirobot remote interaction", *Proc. IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, Czech Republic, June 15-16*.

Corresponding author

Zhen Zhang can be contacted at: zzh2000@sjtu.edu.cn; zhangzhen51@yahoo.com.cn