

Andrey A. Loukianov · Hidenori Kimura  
Masanori Sugisaka

## Implementing distributed control system for intelligent mobile robot

Received and accepted: December 1, 2003

**Abstract** The control system of a mobile robot has a number of issues to deal with in real time, including motion control, mapping, localization, path planning, and sensor processing. Intelligent reasoning, task-oriented behaviors, human–robot interfaces, and communications add more tasks to be solved. This naturally leads to a complex hierarchical control system where various tasks have to be processed concurrently. Many low-level tasks can be handled by a robot’s onboard (host) computer. However, other tasks, such as speech recognition or vision processing, are too computationally intensive for one computer to process. In this case, it is better to consider a distributed design for the control system in networked environments. In order to achieve maximum use of the distributed environment, it is important to design and implement the distributed system and its communication mechanisms in an effective and flexible way. This article describes our approach to designing and implementing a distributed control system for an intelligent mobile robot. We present our implementation of such a distributed control system for a prototype mobile robot. We focus our discussion on the system architecture, distributed communication mechanisms, and distributed robot control.

**Key words** Mobile robot · Control · Distributed system · CORBA

---

A.A. Loukianov · M. Sugisaka (✉)  
Department of Electrical and Electronic Engineering, Faculty of  
Engineering, Oita University, 700 Dannoharu, Oita 870-1192, Japan  
Tel. +81-97-554-7831; Fax +81-97-554-7841  
e-mail: msugi@cc.oita-u.ac.jp

H. Kimura · M. Sugisaka  
The Institute of Physical and Chemical Research (RIKEN),  
Bio-Mimetic Control Research Center, Nagoya, Japan

H. Kimura  
Department of Complexity Science and Engineering, Graduate  
School of Frontier Science, University of Tokyo, Tokyo, Japan

---

This work was presented, in part, at the 8th International Symposium  
on Artificial Life and Robotics, Oita, Japan, January 24–26, 2003

---

### 1 Introduction and discussion

Mobile robot controllers are rather complex systems that have to deal in real time with a number of tasks in order to allow the robot to operate autonomously. These tasks include motion control, sensing, planning, navigation, etc. Robot controllers are usually designed as modular and hierarchical systems. This makes it easier to realize complex system functions using a composition of more simple task-oriented modules. There are numerous works on designing mobile robot control architectures.<sup>1–4</sup> A number of mobile robot controllers were successfully implemented using proposed architectures.<sup>4,5</sup>

If mobile robots are to perform useful tasks in open environments, their control systems should be provided with high-level intelligent features like natural human–machine interfaces: speech recognition, face and sign recognition, and other ways of human–robot interaction. The problem here is that algorithms which may provide these features are usually very computationally intensive. If these high-level programs use the same computer which also handles low-level control and sensing then this will affect the system performance and worsen the response time of run-time system components. To avoid this bottleneck, the control system can be implemented as a distributed one using a number of computers connected by a network. Then the computationally intensive tasks can be run on separate computers or can even be spread between them.

The distributed architecture also offers many benefits, such as openness, dynamical extensibility, and mobility.<sup>6</sup> The components of a distributed system can be created in different programming languages, run on different platforms, and interface with existing systems. The system can be easily extended and configured by adding and removing components, and this can be done even at runtime. In distributed systems, wireless networks allow to equip onboard robot controllers with only critical functions while moving intensive computational tasks and rarely used procedures to the stationary off-board computers. This will make individual mobile robots cheaper and extend their battery life.

This will also make possible the sharing of off-board system components between a group of robots, thus enabling tighter robot-to-robot interaction in multirobot systems.

This article reports our approach to designing and implementing a distributed-network control system for an intelligent mobile robot. We present our implementation of the distributed control system on our prototype mobile robot. The control system of our robot consists of three computers connected to a high-speed 100-Mbps network. Distributed design of the control system allows us to share computational load between available computers; add, remove, and replace system components at runtime; and share services of one component with the others. We discuss the system architecture, distributed communication mechanisms, and robot control approaches in the following sections.

## 2 Control system architecture

From a general point of view, our system architecture follows a well-established structure.<sup>4,5</sup> There are two levels of control built one on the top of the other: the hardware control level and the robot control level (Fig. 1). The hardware control level includes a set of procedures for controlling robot hardware: motors, encoders, sensors, pan-tilt-zoom cameras, etc. On this level, all hardware-specific control issues are resolved and presented to the higher level in an abstract robot state. Motor input voltages are represented by desired velocities, encoder pulses are processed to reflect current robot velocity and odometry status, ultrasonic sensor readings are filtered, and so on.

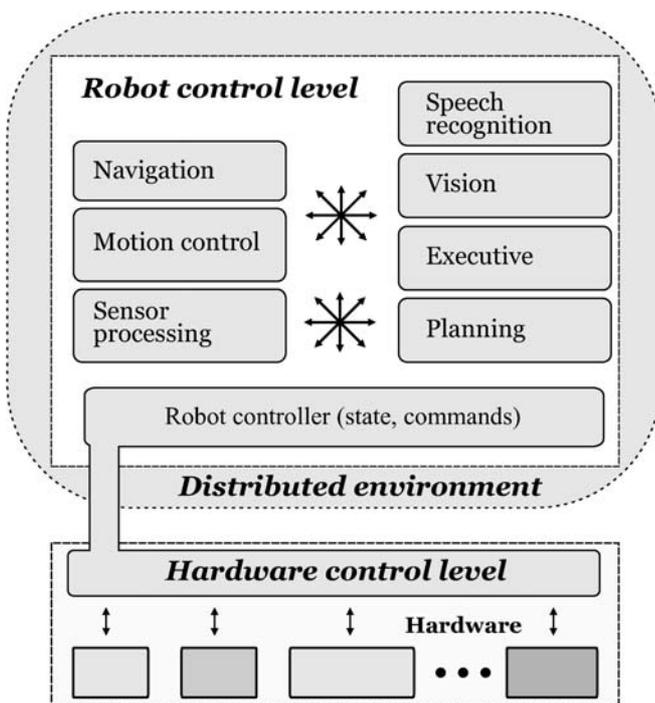


Fig. 1. Distributed system architecture

The robot control level includes procedures for dealing with higher-level control issues from motion control and sensor data interpretation to navigation, planning, and intelligent interfaces. To reduce complexity, these procedures are realized as a set of task-specific separate modules (components) that share the information and services with the others. On this level, the hardware control level is represented as a server component that shares with other modules the information about the current robot state and the set of commands to control this state. In conventional control system implementations, the components of the robot control level are closely integrated between each other to establish a coordinated basis for robot control.

In the case of a distributed network system there are several additional issues to be addressed. The tight coupling between components of the robot control system is no longer possible because the components are executing on different computers and do not have easy access to the internal state of the others. Therefore, the distributed implementation should introduce loose coupling between system components where the information is exchanged only through communication mechanisms. These communication mechanisms have to be flexible enough to allow system components to use different communication strategies such as broadcasting, one-to-one or one-to-many (server-client) connections. In our system, we used common object request broker architecture (CORBA) open distributed object architecture to achieve loose coupling and flexible communication mechanisms.

The distributed components and communication protocols also need to support resource sharing and provide fast system response in certain cases. Resource sharing is needed to handle situations when several modules compete for a single resource, for example, to control robot motions or to move a rotating camera. In our system we used queues, locking, and priorities to resolve these problems.

There is also a problem of finding an approach to integrate all distributed components together, making them all work as a complete system. This requires some form of central management that will coordinate, direct, and oversee operation of distributed components in order to execute desired robot actions and achieve their goals. In our system, we delegated this role to one or more distributed components, called the executive.

## 3 Communication mechanisms

In this section we discuss the communication mechanisms we use to implement the distributed system. We used CORBA architecture and infrastructure to link distributed modules of the control system together. This architecture is vendor and platform independent and can be used in a variety of areas including real-time applications. We used the omniORB-free implementation of CORBA.

In our system, there were two levels of communication between our modules: object level and messaging level.

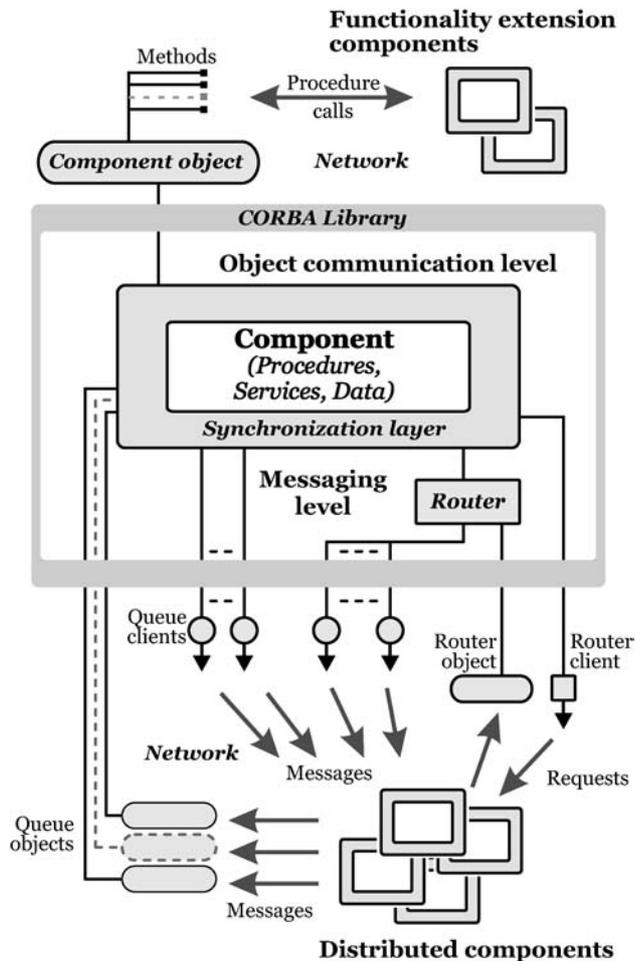


Fig. 2. Distributed communication mechanisms

Figure 2 shows all distributed communication mechanisms that are available to the module.

The object level allows a component to export its procedures and internal state to the networked environment. To access component's exported procedures and data, other system components use the CORBA object that provides interfaces to call procedures and query data. For example, the robot controller component uses object communication level to export low-level motor control routines, encoder, and sensor readings. More advanced or specialized versions of the robot controller can be developed on top of the existing controller using its exported procedures and data without having to deal with complex hardware issues. This makes the system architecture open and easy to extend. Therefore, the object communication level is used to provide the means to extend component functionality and when tighter coupling between system components is needed.

A greater extent of information exchange is performed at the messaging level of communication. The system components are encouraged to communicate on this level because it is flexible and it allows the use of various communication strategies. On this level, system modules communicate by exchanging messages, each one consisting of the header and the message data. The message header includes

information about message context (category), message identifier, message recipient or sender, message priority, and message marker. This information is used by system communication algorithms to handle messages more efficiently while keeping network use to a minimum.

Messages are sent between modules with the help of two basic but powerful mechanisms – queues and routers (see Fig. 2). Queues are communication objects that are used to receive the incoming messages and sort them according to their priority for later processing by the recipient. The routers provide configurable message broadcasting in case there are many recipients. Queues and routers are exposed to the distributed network environment through corresponding CORBA objects. Every queue and router object is given a unique name identifier that is used by CORBA service libraries to locate the appropriate queue or router on the network and deliver messages to them. The module may use all communication mechanisms or just some of them depending on its purpose within the control system.

Queues receive messages from other distributed components on the network through their CORBA queue object interfaces. The interface of the CORBA queue object is minimal – it contains only one message reception method. This method accepts the message from the caller and puts it into a local queue for later extraction and processing. To perform its functions, a control system component can create as many queues as needed. A static one-to-one communication channel between two components may be formed by posting messages to the queue object of the opposite component. To send a message to the desired queue, the distributed component uses the CORBA queue client, the function of which is to find the CORBA queue object on the network and to call its message reception method. The queue client uses the CORBA name service library to locate the desired queue by its name on the network.

Routers in our system provide flexible message broadcasting and dynamic one-to-one communication channels. The broadcasting allows a system component to send outgoing messages to any number of system modules. Using this mechanism, for example, the robot controller can send updates of its state to all modules that need this information. Other components subscribe for messages that they need to receive from the router. The component sends outgoing messages to its router, which then distributes these messages to all subscribed components.

The router consists of: (a) a CORBA router object used by other components to subscribe/unsubscribe for broadcasting and configure message filters; and (b) a dynamic register of the subscribed components. Similar to the queues, every router is exposed to the distributed network environment through the CORBA router object, which is accessed by subscribing components through CORBA router clients.

In order to subscribe, a recipient component sends the router the name of the queue object from which it wishes to receive messages. After subscription, the component can also create a set of message selection filters. The router uses these filters to select or discard outgoing messages based on the message header information before sending them to the

subscribed component. The messages are delivered to the recipient queues by CORBA queue clients. The router object uses CORBA service libraries to check if all subscribed recipient queues are still present on the network. If the service library cannot find the recipient queue object on the network, the register of subscribed components is updated accordingly.

If the component shares its resources with other distributed components, there is a problem of serializing access to this resource. A module can have resources, functions, services, or data that cannot be accessed by many modules simultaneously. To handle this problem, modules use synchronization mechanisms to provide correct sharing. In our system, these synchronization mechanisms include command queues, request queues, and resource locking. These instruments form a synchronization layer around shared resources of the component.

---

#### 4 Robot control with distributed components

Components in the distributed system can operate with little or no supervision from outside. Nevertheless, in order to operate as a complete control system, the collection of distributed modules needs to be integrated together. Coordinating, directing, and overseeing the operation of distributed components in order to achieve control goals requires some form of central management. In our system, this management was provided by the so-called executive component.

The executive component task is to coordinate the work of other components. The executive communicates with other modules on the messaging communication level. Its scripting engine allows the programming of new robot actions and behaviors (we call them “activities”), and executes these programmed activities in real time. Robot activity scripts follow finite state automaton (FSA) semantics and use executive kernel, threading, message processing, and event processing libraries to perform their functions and state transitions.

The executive kernel and threading library provide activity programs with the environment in which they can execute and schedule themselves. In FSA semantics, the activity execution is represented as a sequence of states and interstate transitions. Some of these states are predefined (initialization, termination, suspended, etc.), while other states are defined by the activity itself. The activity also defines all possible transitions between its states and the operations that should be performed during these transitions. The executive kernel schedules activity execution within FSA framework. The threading library allows activities to coordinate and synchronize their execution. Running activities can spawn child activities that can execute sequentially or in parallel with the parent activity.

The message processing library enables robot activity programs to communicate with distributed system components and coordinate their work. The activity can send messages directly to a component or use the router of executive module to broadcast messages. Using message filters, the

activity can ask the messaging library to send it an event when a specific message arrives or it can create a separate message queue for that purpose. Activity scripts can contact routers of other components to subscribe for required messages.

The event processing library binds threading and messaging libraries together and plays an important role in FSA execution semantics. In our case, events trigger almost all activity state transitions. The messaging library uses message triggers and message filters to signal the arrival of the message of interest. The threading library uses events when scheduling activities and when working with timers and synchronization. The executive component’s kernel handles all interactions between described libraries and running activities.

Although in our implementation we use only one executive component, for the distributed system, it is possible to include a number of executive components. In this case, a control hierarchy can be constructed, where each executive component deals with a limited selection of control problems on a certain level and reports the results and its status to the higher level.

---

#### 5 Conclusions

In this article, we presented our implementation of a distributed control system. The distributed design provides open and dynamic framework for realizing mobile robot control. The distributed design enables sharing of computational load between available computers to achieve maximum system performance when using computationally demanding control algorithms and intelligent robot–human interfaces. The system consists of collection of separate executable task-specific components that communicate between each other over the network. CORBA-based communication mechanisms that allow the linkage of system components together using different communication strategies were considered. The structure and implementation details of the executive component, the task of which is to integrate and coordinate the operation of distributed components, were discussed.

---

#### References

1. Brooks RA (1986) A robust layered control system for a mobile robot. *IEEE J Robotic Autom* 2:14–23
2. Connell J (1992) A hybrid architecture applied to robot navigation. *Proceedings of IEEE International Conference on Robotics and Automation*, pp 2719–2724
3. Gat E (1992) Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. *Proceedings of the Tenth National Conference on Artificial Intelligence* pp 809–815
4. Konolige K, Mayers K, Ruspini E (1997) The Saphira architecture: a design for autonomy. *J Exp Theor Artif In* 9:215–235
5. Burgard W, Cremers AB, Fox D, et al. (2000) Experiences with an interactive museum tour-guide robot. *Artif Intelligence* 114:3–55
6. Martin DL, Cheyer AJ, Moran DB (1999) The open agent architecture: a framework for building distributed software systems. *Appl Artif Intell* 13:91–128