Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

A Flexible, Hierarchical and Distributed Control Kernel Architecture for Rapid Resource Integration of Intelligent Building System

Wen-Ya Chung, Li-Chen Fu, and Shih-Shinh Huang

Department of Computer Science & Information Engineering, National Taiwan University, ROC

Abstract

In the past years, buildings are designed to be more and more powerful and intelligent mechanism. But it is still far from being perfect since building systems are in lack of a uniform specification to establish standards for various appliances and communication protocol, and are also in lack of a robust kernel to integrate all sub-systems in the buildings. For the former problem, we expect that the major appliance factories will eventually set up the standard of the appliances and communication protocol. For the latter problem, this paper proposes complete system architecture with integrated control kernel to construct an intelligent building system rapidly and efficiently.

Introduction

Due to the advanced developments in computers and the wide-band network, people's lives are made more convenient, and their quality is made better. Especially, appliance automation is closely related to our daily life. In the past, the appliances are working stand-alone and cannot cooperate with one another. In the recent years, these appliances can be monitored and controlled by controllers and be displayed on terminals, but are still in lack of integration. Since the present building automation system is not equipped with efficient integration mechanism, it cannot fully extend the worth of these developments.

In order to achieve this goal of integration, many appliance manufacture companies focus on the development of intelligent appliances to integrate them into a complete building automation system for their monitoring and controlling. Due to the advent of advanced computer and wide-band network as has been pointed out earlier; the personal computer-based environment is the most suitable platform for system integration. The personal computers can be linked by the network and are capable of powerful computation and easy display. We can take advantage of such abilities to develop an integration system. This situation is certain to cause some problems, such as how to integrate different appliances from different vendors, and how to communicate them more flexibly and efficiently.

For this reason, we devise a flexible architecture of building automation system and intelligent control kernels to integrate various appliances into a building automation system with high level performance in system responsiveness and energy saving. This architecture called Multi-Server and Multi-Agent architecture (MSMA) and the intelligent control kernels are called the Enhanced Resource Integrated Control Kernel (ERICK) and the Resource Integrated Control Kernel (RICK).

Intelligent Building System

In this section, we will briefly review the development of intelligent building systems and analyze the advantages and disadvantages of the present system, and describe the development trend for the future. An intelligent building system must integrate different resources so as to make them work more efficiently to increase system performance, and to minimize investment and cost for system. These resources include (1) power sub-system, (2) sensor sub-system, (3) lighting sub-system, (4) HVAC sub-system, (5) audio/video sub-system, (6) elevator sub-system, (7) security sub-system, (8) warning sub-system, etc. However, how to integrate such a complex system is an interesting challenge.

In the past, the integration strategy for the traditional intelligent building system consists of a central management workstation and several device controllers to control physical devices, and has the field data bus to link controllers with physical devices. Under this strategy, the system will be complex and be in lack of efficiency because the load of the central management workstation is too heavy to handle so many controllers.

For solving these problems, the American National Standards Institute (ANSI) and the American Society of Heating Refrigeration and Air-conditioning Engineers (ASHRAE) establish the standard, named Building Automation and Control Network (BACnet), to integrate all the building system equipment. BACnet is an open communication protocol standard, which allows different building equipment vendors to work together [9]. BACnet also incorporate several Local Area Networks (LANs) techniques, such as Ethernet, ARCNET, MS/TP, and LonTalk. By this technique, various pieces of equipment can be linked to the underlying system via a uniform communication media.

In a distributed intelligent control system, the central computer mainly evaluates the system performance and collects information from each distributed workstation for system analysis and monitoring. Each distributed workstation has its own intelligence to deal with all the sub-systems connected, and has its own building management system to evaluate local system performance.

With distributed intelligent control system, the central computer can reduce the handling load by not confronting each piece of equipment and instead only dealing with the distributed workstations. This setup not only increases system performance but also makes the system more reliable due to the load sharing and the job decomposition.

After the architecture of the intelligent building system has been described, the most important part in an intelligent building system is that what kinds of information exist and how they influence the system. The first kind of information in the system is request, which may involve several pieces of equipment. Since each kind of equipment has its own communicating language, it has to negotiate with some translators (or gateways), through which it can talk to others and exchange information using the same language. Once all the pieces of equipment can communicate with one another, the system supervisor or occupants will also have to negotiate with them. For this purpose, a proper user interface for users to talk with the system will be necessary.

We will then discuss how they operate within the intelligent building system. Since there are so many information exchanges in the system, a more reliable and efficient method should be adopted, such as handshaking and encoded message exchange, so as to reduce the network traffic jam and to increase system performance.

Since the building automation system is a complex control system, we have to decompose the entire system into structured sub-systems. In order to meet the requirements of these features, here we propose a hierarchical and distributed architecture to construct the intelligent building system, and decompose the overall system into sub-systems according to different spaces and different functionalities, etc. For integrating the entire system, in the first place, we have to link all the physical devices to the system. Furthermore, we define the system request format with encoding, and request flow within the system. Hence, we provide system monitor and control mechanism with flexible and reliable features.

Multi-Server & Multi-Agent Architecture in IBS

Distributed control system (DCS) is now in widespread use under building automation system. Since single server architecture is not proper to handle so many varieties of resource within a building, we introduce Multi-Server and Multi-Agent (MSMA) architecture. In MSMA architecture, we can easily establish and set up the building automation system into a more flexible and efficient intelligent building system. Several different or same types of resources can work together by cooperation. There are two resource integrated control kernels in MSMA architecture; one performs as interface between servers and agents, named Enhanced Resource Integrated Control Kernel (ERICK), and the other interface agents and device control objects, named Resource Integrated Control Kernel (RICK), to deal with the message exchange without requiring all relevant parties to know one another and to handle the cooperation between the associated layers.

In this section, we present the detailed description of MSMA architecture. The definitions and features in MSMA architecture will be revealed in detail for the concept of establishing intelligent building system.

In Figure 3.1, we decompose the entire system into four layers, and a media between every two adjacent layers. In this figure, there is an external request to ERICK. Note that each request can be presented by a supervisor or an occupant. Besides these layers, the communication media between the device control objects and the physical devices can be BACnet, LonWorks, or Ethernet communication media. We will describe the layers as follows, and define them in detail in the later sections.

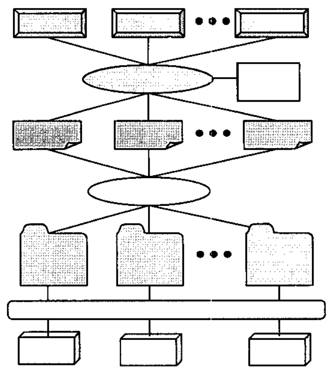

FIGURE 3.1 MULTI-SERVER AND MULTI-AGENT ARCHITECTURE

- **Layer 0:** This layer mainly handles the entire system processes. It is also called the Multi-Server layer aiming for serving requests in the system. This layer, which can be scalable, may consist of more than one server for the system. This mainly explained why the Multi-Server layer is preferably introduced in this paper.
- **Layer 1:** This layer mainly deals with the jobs received from servers or device control objects. Such layer is called Multi-Agent layer, and the cooperation among agents is accomplished through the kernel. With this cooperation mechanism, the complexity of message exchange can be reduced.
- **Layer 2:** This layer consists of device control objects (DCOs), which can be classified into several types. One type of DCO collects data for sensor devices, the other triggers the actuator devices, and the more complex types, handle audio/video and HVAC system.
- **Layer 3:** This layer consists of physical devices. All devices transmit their data to device control object and are controlled by the device control object through the communication media.

From the above description of each layer, we can realize that the entire system operates under a distributed and hierarchical architecture. Farther, the cooperation among various entities is accomplished by the kernels, which can reduce the complexity of mutual message exchange. After gross explanation of each layer, we will describe in detail about each layer in the next sections.

■ Multi-Server

In MSMA architecture, the servers are in charge of handling the processes associated with the whole system. Each server must have common modules for communication and registration, and a module for service. Since servers provide service to the request, each of them must let ERICK know what its service content is, and then register before it can become active. We describe the major modules as follows.

- **Communication Module:** This module provides communication protocols for servers, including

Ethernet communication channel. Each server can send and receive requests and other messages through Ethernet communication channel.

- **Registration Module:** This module deals with registration between a server and the kernel with the help of the aforementioned communication module. The registration indicates that an available server is ready to participate the system and reveals the information of service.
- **Service Module:** This module depends on the functionality that the underlying server can provide. As has been mentioned before, each server will inform the kernel of its kind of service through registration process. A server can have one or several of the following types · of service, and/or even the user-defined service, such as Integrating agent service, recovering error service, monitoring system service, dispatching request service, and emergency service.

With above common modules, the server can control the overall system and serves the relevant requests through these modules. Since the server can be more than one, we need a control kernel to integrate and to manage them. The control kernel not only handles the requests but also coordinates the servers, and store the status of each server such that the kernel can understand what services are present and how many servers remain active in the system.

■ Multi-Agent

In MSMA architecture, the agents handle the processes associated with every request. Each agent also has common modules for communication and registration, and a module for execution. Since it is the agent executes the requests, each agent must inform the kernel of its abilities through the process of registration. The difference between an agent and a server mainly lies in the fact that there are two kernels serving the agents, but only one kernel serving the servers, as depicted in Figure 3.1. Obviously, both of the kernels must know the abilities of each agent. Therefore, the kernels can dispatch various requests to some suitable agents based on their current status. We now describe the major modules as follows.

- **Scheduling Module:** This module deals with request schedule. Since an agent may receive more than one request, it is necessary to schedule the entire received requests. To that purpose, a priority queue is devised to share all the requests. In order to avoid starvation, as time gradually elapses, the priority of each request in the queue will be increased.
- **Request Execution Module:** This module will depend on the functionality that the agent can provide. And as mentioned before, the agent has to let the kernels know its capabilities through the process of registration. The execution module is responsible for fulfilling the received request by interacting with appropriate set of device control objects.

Since the major mechanism of Multi-Agent is the same as that of Multi-Server, such Multi-Agent structure also has the superiority of distributed computing environment and can make the system more scalable by adding more agents through registration to share the load of other agents. There are two kinds of agents: function agents and space agents. The former is an agent with a major functionality, such as security function or power utilization function, to handle different situations, whereas the latter is an agent located depending on the geometric distribution of physical devices. We now describe these two kinds of agents respectively as follows.

□ Function Agent

A function agent always consists of a major functionality, such as security function or power utilization function. We claim that the function agent is able to make decision and to report the status of the whole system. For this reason, this kind of agent must self-activated to control the devices according to its functionality, such as security function, power utilization function, and power-on & power-off function.

With these function agents, the supervisor and users can control the system more flexibly and efficiently. After description of the space agents, we will give instances about function agent and space agent later on.

□ Space Agent

A space agent is an agent depending on where it is located. We can decompose the building into several zones either by geometric partition or by logic distinction, such as room and corridor or bedroom and living room. In a school building, for example, we can divide building into several laboratories, such as robotic laboratory, multimedia laboratory, and offices, etc. This division can help occupants to monitor and control the local system more efficiently and may not call for the central coordination control in an entirely open distributed environment.

■ Device Control Object

In MSMA architecture, there is another important role, called Device Control Object (DCO), to handle the communication between the control kernel and the physical devices. The mechanism of device control object is relatively simpler than that of the server or the agent. Their missions are to collect device data or to trigger device according to the commands received from agents. In order to accomplish these missions, a device control object must have the Execution Module. This module depends on the functionality that the device control object is going to provide. A DCO can have in its execution module one or several of the following types of executor, such as collection executor, trigger executor, audio/video executor, and HVAC executor, etc.

Functionality in MSMA Architecture

We now describe how to apply the proposed MSMA architecture to establish intelligent building systems. Since an intelligent building system is a complex control system, we have to classify the entire system into several sub-systems. Moreover, since the MSMA architecture is distributed and hierarchical, we have to describe the way to associate each resource with suitable functionality.

The servers provide system services, such as temperature management service, alarm control service, elevator control service, monitor service, security management service, and power saving service, etc. In order to provide these services, some system information must be extracted first, such as current temperature, current humidity, and human activity, etc. Then, the server will evaluate the system based on the information, and will make decision and dispatch requests to appropriate agents for execution. In order to avoid unexpected situation and to enhance system performance, it is preferable to have more servers providing the same service as for mutual backup and/or load sharing. By employing such distributed computing environment, the system performance will be more powerful and efficient.

Every server has its knowledge base for decision-making. Of course, one may ask how to perform the service evaluation before providing service. For this, when some resource requests a service, the corresponding server will collect the current system information, and take that as a set of parameters for the subsequent evaluation procedure. After evaluation, the server dispatches the request to some appropriate agents for execution and reports the result of execution. After all, we can view these servers as brain trust of this system.

The agents are responsible for executing the requests issued by servers in order to provide some service. When a function agent or a space agent receives a request, it will execute some relevant actions to achieve the goal of the request. Since the system has been decamped into sub-systems, various agents are assigned to correspond to those underlying functionalities or spaces.

On the other hand, for space agents, usually their locations will depend on their either physical spatial representation or logic association, and deal with the devices in the area around those locations. They are not like function agent working across the spaces. What they need to do is to handle the device control objects associated with them, and deal with the requests from servers. To realize why space agents will be need in this system, we see that very often a supervisor or the occupants want to know the status of a desired zone, and the space agent can exactly perform this kind of operation for them.

Device control objects perform as device drivers in this system. Since there are too many types of device to design drivers for each of them. Thus, we provide a device control object that can handle more than one device. For this reason, we propose fundamental device control objects to deal with physical devices, such as sensor devices, actuator devices, audio/video devices, and HVAC system, etc.

The MSMA architecture provides a rapid and flexible model for message exchange, system monitoring and system control in the intelligent building systems. It offers not only system management but also on-line reconfiguration mechanism to maintain high efficiency. From the occupant viewpoint, it provides an extremely good and systematic method of system construction. But from the system's viewpoint, ERICK and RICK nicely fit the MSMA architecture into the intelligent building system.

Enhanced Resource Integrated Control Kernel & Resource Integrated Control Kernel

We realize that how important ERICK and RICK in the MSMA architecture are. In this section, we will describe the complete data structure of the intelligent control kernels and how they work with servers, agents, and device control objects within the MSMA architecture. Since the resource integrated control kernels play the most important role in the MSMA architecture, they are equipped with several modules to accomplish the missions, namely "Communication Module", "Request Fetching and Dispatching Module", "Request Management Module", "Resource Management Module", "Error Recovery Module", "Monitor and Control Module", and "GUI Module". In the next section, we will give a complete description of each module, including its behavior, policy, and functionality. Even though they can be realized in the same computer it is however more efficient to realize them in different computers within the distributed computing environment.

ERICK and RICK fetch requests from resources through communication module, and pass the request to the request management module for scheduling. Another mission is to re-dispatch some recovered request from the error recovery module or re-fetch a similar but error-free request from the same resource. Since the request arrival time is random, it is necessary to hold the request in a buffer. For this reason, we apply a priority queue to keep the requests.

ERICK and RICK deal with request in the request management module, and handle the following jobs: parsing request, scheduling request, and dispatching request. Because the request is encoded with a pre-specified message format, this module must parse the request first no matter whether it is correct or not. If the request message is error, this module will pass the error request to error recovery module to recover the error. After the error recovery module examines the request, if the error has been recovered, then the module will send it to the request fetching and dispatching module to re-dispatch the request; otherwise the request fetching and dispatching module will send it to the original resource to resend the request. Furthermore, this module consists of a priority queue with five priorities: real-time priority, above normal priority, normal priority, below normal priority, and low priority. Hence, the request management module will dispatch the request from the head of the priority queue.

The resource management module holds the status of all resources; including servers, agents, and device control objects. The status of each resource includes its operational situation, idle, busy, or dead, and the information of location. Although each resource has different functionalities and capabilities, this module has to determine which kind of requests should be sent to which suitable resource. When a resource is in the dead state, this module will send an error recovery request to the error recovery module for recovering the dead resource.

The error recovery module will receive a request from the request management module and resource management module on the occasion of error occurrence. This module, in the first place, will try to recover the error. After this module has examined the error, it will acknowledge the module which sends out the recovery request about whether the error has been successfully recovered or not. If the request management module has received a negative acknowledgement, the request fetching and dispatching module will ask the original resource to resend the request. Similarly, if the resource management module has received a negative acknowledgement, it will readily detach the resource.

To strengthen human intervention of the proposed intelligent building system, the system should provide monitoring and controlling mechanisms for supervisors and residents. In order to provide this mechanism, this module has to be closely tied to the graphical user interface module to facilitate the aforementioned parties to review the status of the entire system. In this module, the monitoring mechanism is built upon either polling or event driven techniques. In fact, some resources status should be more preferably extracted by use of the polling method, such as temperature and humidity, but some may be better to use event driven technique, such as fire alarm and emergency events.

The intelligent building system is within a computer-based environment, and thus the system should have friendly graphical user interfaces (GUIs) to allow the supervisors and the residents to monitor and to control the entire system. The GUI module supports two control modes: kernel mode GUI and user mode GUI. The former supports interfaces for supervisor whereas the latter provides capability of remote access for general users.

We now explain how to encode request. When ERICK or RICK receive raw request that has never been recognized yet, the request will be encoded by the "encode function" as shown in Figure 4.2(A). Then, the resource will receive the encoded request. Hence, the resource knows how to encode message, and sends request with encoded request as shown in Figure 4.2(B).

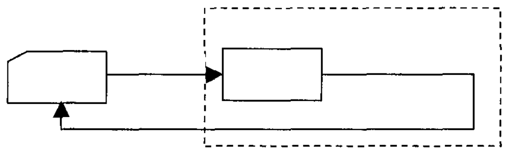
FIGURE 4.2(A) RAW REQUEST ENCODING

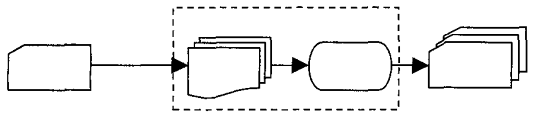
FIGURE 4.2(B) REQUEST MANAGEMENT WITH ENCODED REQUEST

The system consists of three main resources: servers, agents, and device control objects. Each resource has its own missions and executes them according to the request from other resources. Servers provide services for system integration, and agents have to accomplish the requests from servers, and device control objects control physical devices directly and respond to the agents the result of control.

Since resources are located in different places, ERICK and RICK must keep their status reliably. For this reason, ERICK and RICK set a priority queue to hold them. In the priority queue, there are several attributes to determine their priority. These attributes include critical functionality and response time. For the same type of resource, such as servers, agents, and device control objects, we give them different priority according to those attributes. The most important reference attribute is critical functionality, such as emergency, security, and common control function, etc. The other reference attribute is response time, which is determined by the time period between sending and receiving request. Network transmission rate and the physical distance between the resource and the control kernel can determine this attribute.

According to the above description, we can know that the resource management must handle those resources by dynamic scheduling. In order to reduce the complexity and increase system performance, we use polling technique with long time slice. This mechanism can avoid frequent changing in the priority queue and achieving the goal of dynamic scheduling.

There may cause starvation in the priority resource policy. For this reason, we apply the threshold filter policy to solve this problem. After the time slice has gone, all the priority will increase in the priority resource policy. If there exist some resources have not served, ERICK and RICK will dispatch request to these resources for service. Since the slower resource can also serve the request, this threshold filter policy will avoid the resource starvation.

Servers provide system services, such as comfort control service, energy saving service, and emergency handle service, etc. Since the servers provide services, they consist of some knowledge base, algorithms and policies. We now explain how ERICK manages these servers as follows. First, ERICK has known how many and what services the available servers have by registration. After knowing the information about servers, ERICK will give them the initial priority. After the time slice has gone, ERICK will give them different priorities according to the attributes described above.

Agents execute the requests from servers, and handle the regional device control objects. We propose that agents should allocate the device control objects according to the space consideration. For example, apply one or more agents for a robotic laboratory control and agents for an office. This allocation can take advantages in the distributed computing environment. Although an agent can deal with the device control objects in different rooms, we still recommend retaining the distributed environment. Because of the distributed environment the system can localize the fault and can be more reliable.

Since device control objects are the media between the physical devices and the control system, it is important to develop templates for every kinds of devices. Here, we support only dynamic data exchange (DDE) communication channel for LonWorks based devices, and some devices with BACnet based and Ethernet based devices for the sake of demonstration.

The device control objects are classified according to the functionality of the devices. We can classify device control objects into following types, such as collection (sensor device), trigger (actuator device), both together (sensor and actuator device), or complex types (audio/video device, etc). With these types of device control objects, all complex devices can be made up of them. For example, a HVAC can be controlled with a temperature sensor device and a motor actuator device.

Resource management must handle resources with suitable control polices. It can manage the resources more systematically and efficiently according to these policies described as follows.

We regard these resources as elements in a resource buffer with their status, including the socket properties and functionalities. However, the resource may be unavailable in the system sometimes. If this situation happens, the resource must be preempted from the buffer until it has been available by error recovery. After recovering the dead resource, the resource will be resumed for service. So, we apply this preemptive resume policy for this situation.

Each resource has its states: idle, busy, and dead state. Here, we describe the resource state transition diagram and the detailed state description. The resource always stands in the idle state without receiving any request. When the resource receives a request, it will transfer current state to busy state indicating that the resource is executing request. After accomplishing a request, the resource will return to idle state waiting for next request. However, if the resource occurs any error, it will transfer current state to dead state and send an error recovery request to error recovery management to recover error. If this error has recovered, then the resource will return to idle state for serving this system. Otherwise, the error recovery management will detach the resource from the system.

In this system, we provide both local and remote access. For local access, the supervisor and residents can monitor and control this system through the kernel mode graphical user interface to access the system. For remote access, we supply graphical user interfaces with standard browsers, such as Microsoft Internet Explorer and Netscape Navigator, with ActiveX or Java technique for authorized users to access the system. With this mechanism, wherever one travels, he can use a personal computer, notebook, or Personal Digital Assistant to get on the Internet, and he can monitor and control this system easily and conveniently.

In any kind of information system, the error recovery mechanism is needed for both self-diagnosis and interactive diagnosis. For error recovery mechanism, the control kernels consist of error recovery management to recover errors, such as request or resource error. If the request can be recovered by checking the contents of this request, then it will execute normally. Otherwise, error recovery management cannot recover the request, so that the request will be resent again or the supervisor or residents can demand to resend request. On the other hand, if the resource is erroneous, then the intelligent control kernels or the supervisor can detach this resource.

Experiment and System Evaluation

In our experiments, we apply several communication protocols, such as LonWorks for the temperature sensors, the person infrared radio sensors, the smoke sensors, and the brightness sensors, and BACnet for the air conditioner, and Ethernet for the camera device. Such an experiment reveals that the system can support any kind of devices and the designer can apply different communication media according to the device characteristics. We will explain how to manage these devices inside our proposed system.

We also simulate the conditions for an intelligent building with twenty floors, and there are 100 devices in each floor. In this simulation, we compare the response time with the formal experiment. The response time is the time period that the request sent from ERICK to devices and return to ERICK. We list the experiments parameters in Table 5.1, and the experiment results in Table 5.2.

From Table 5.1, Table 5.2, and Figure 5.2, the response time do not obviously increase following the size of the intelligent building. Furthermore, the most response time of the requests is 10 milliseconds to 50 milliseconds, and this reveals that no matter how large size the intelligent building is the response time can keep with fast responsiveness. Hence, the proposed system is suitable for the large size of the intelligent buildings.

TABLE 5.1 EXPERIMENTS PARAMETERS

Resources	Experiment 1	Experiment 2
Server	2	2
Agent	1	20
DCO	2	22
Computers for DCOs	3	7
Random Requests	1050	93630
Evaluation Requests	7000	7000

TABLE 5.2 EXPERIMENT RESULTS

Resources	Experiment 1	Experiment 2
Number of requests	8050	100630
Max. Response time	260 ms	300 ms
Min. Response time	10 ms	10 ms
Average Response time	23.64 ms	32.55 ms

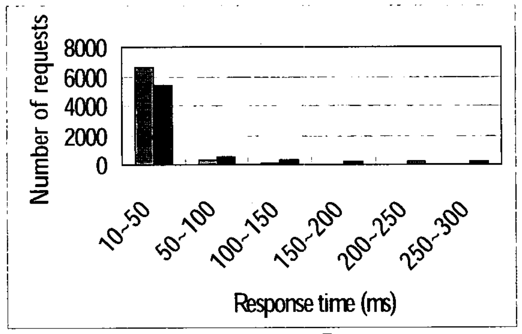

FIGURE 5.2 RESPONSE TIME DISTRIBUTION

Conclusion

In this paper, we propose a hierarchical and distributed architecture, namely Multi-Server and Multi-Agent (MSMA) architecture with Enhanced Resource Integrated Control Kernel (ERICK) and Resource Integrated Control Kernel (RICK) to handle the entire system. In MSMA architecture, there are three major resources: servers, agents, and device control objects, and each of them handles their own jobs. The servers mainly provide services for the whole system, and the agents mainly execute requests from the servers, and the device control objects mainly link the physical devices to the system and provide local monitoring and controlling mechanism. The most important roles are ERICK and RICK. They have to manage system resources and requests efficiently and flexibly, and supply graphical user interfaces for the supervisor and residents to monitor and control the entire system. The features of the MSMA system are flexible, efficient, scalable, and re-configurable, and ERICK and RICK can integrate the system easily and rapidly.

In the proposed MSMA architecture, every types of device can be easily and rapidly linked to the system, and make the devices visible and controllable. With this powerful mechanism, we can easily integrate the past, the present, and even the future devices into the system. Furthermore, the proposed system support remote access, and this development is the trend of future development, i.e. network development.

We implement these resources and resource integrated control kernels in this paper. ERICK and RICK support on-line management and configuration and analysis, such as resource management, request management, scheduling, and control mechanism, and artificial intelligence. Since the system is developed on the personal computer-based environment, we can take advantages of computer, such as computing capability, transmission capability, and display capability, etc. With these capabilities, we can easily construct the intelligent building system and make the system more powerfully and flexibly.

References

[1] Wong, A.C.W.; So, A.T.P., "Building Automation in the 21st Century, " *Proceedings IEEE International Conference on Advances in Power System Control, Operation and Management,* Vol.2, 819-824, 1997

[2] Jae-Chul Moon, Hyo-Sang Lim, and Soon-Ju Kang, "Real-Time Event Kernel Architecture for Home-Network Gateway Set-Top-Box (HNGS), " *IEEE Transactions on Consumer Electronics*, Vol. 45, 488-495, 1999

[3] AdAstra Research Group, Ltd, http://www.adastra.ru/

[4] George P. Lekkas Nicholas M. Avouris, and George K. Papakonstantinou. "Development of Distributed Problem Solving System for Dynamic Environments," *IEEE Transaction on Systems. MAN, and Cybernetics*. Vol. 25, No. 3, 1995

[5] Liu, Song-Han, and Li-Chen Fu, "Multi-agent Based Control Kernel for Flexible Automated Production System," *Proceedings IEEE International Conference on Robotics and Automation*, 1998

[6] Sarit Kraus, "Negotiation and Cooperation in Multi-Agent Environments", *Journal of Automation in Artificial Intelligence*, 1997

[7] Abeck, S.; Koppel, A.; Seitz, J., "A Management Architecture for Multi-Agent Systems," *Proceedings IEEE Third International Workshop on Systems Management*, 1998

[8] B. KADAR, MONOSTORI, E.SZELKE, "An Object-Oriented Framework for Developing Distributed Manufacturing Architecture," *Journal of Intelligent Manufacturing*, Vol. 9, 173-179, 1998

[9] Steven T. Bushby, "BACnet: a Standard Communication Infrastructure for Intelligent Buildings," *Journal of Automation in Construction*, Vol 6, 529-540, 1997