UPnP Robot Middleware for Ubiquitous Robot Control

Sang Chul Ahn, Ki-Woong Lim, Jung-Woo Lee, Heedong Ko, Yong-Moo Kwon and Hyoung-Gon Kim
Imaging Media Research Center, KIST
39-1 Hawolgok-dong, Sungbuk-gu, 136-791 Seoul, Korea
{asc, lkw, ricow, ko, ymk, hgk}@imrc.kist.re.kr

Abstract – The UPnP(Universal Plug and Play) architecture offers pervasive peer-to-peer network connectivity of intelligent appliances in dynamic distributed computing environment. This paper shows that internal software integration of robot and robot service in ubiquitous computing environment are the same thing in nature. This paper also proposes to use the UPnP architecture as a unified framework not only for robot internal software integration, but also for ubiquitous robot control. This paper shows that the UPnP technology could provide a unified framework by describing the usefulness of the UPnP and by showing some experimental result regarding it.

Keywords – UPnP, Robot, Middleware, Ubiquitous, Integration.

1. Introduction

For a long time, many researchers have been developing robot technologies in various fields: navigation, manipulation, vision, voice recognition, and even robot design. Recently the progress of technology developments has become more rapid than ever. However, it is yet hard to build a complex robot as a combination of those various technologies. Building a complex robot requires a lot of software components that are made by many researchers. It requires some 'glue' to connect the software components, too. Middleware can play the role of the 'glue' for software component integration.

Though middleware is not popular in robot field, the middleware that have been used most is the CORBA (Common Object Request Broker Architecture)[1]. The projects that adopted the CORBA include the HRP(Humanoid Robotics Project)[2], the OROCOS(Open Robot Control Systems) project[3], and the Miro(Middleware for mobile robots) project[4]. Currently many robot researchers consider the CORBA as a possible candidate of their middleware when they try to integrate robot software.

In the previous paper[5], we proposed to use the UPnP (Universal Plug and Play)[6] as a robot middleware, and examined its possibility. As can be seen in Fig. 1, the UPnP was developed for dynamic distributed environment while the CORBA was developed for distributed environment. Here the dynamic environment means that computing devices can join and leave the network dynamically. As internet was widely spread, the need to accommodate dynamic distributed computing environment was increasing, and new middlewares were devised. The UPnP architecture is one of them.

The background idea of our approach is that a robot system itself will or should be dynamic distributed computing environment. Every component of a robot will become modular, and they will be configured dynamically in the future. Another point of view is that robots will be deployed at home or at office like a TV or a refrigerator in the future. And they will have to dynamically join and communicate with home or office networks as home and office are expected to be ubiquitous computing environment in the near future. Therefore, robot will need an appropriate middleware internally and externally for accommodating these dynamic distributed computing environments. The UPnP technology has good features for this environment. It shares the service oriented architecture with emerging Web Service technology[7], so it provides loosely coupled architecture for component connectivity. It also provides late binding and automatic discovery of services, which makes it appropriate for dynamic distributed computing environment as well as ubiquitous computing environment.

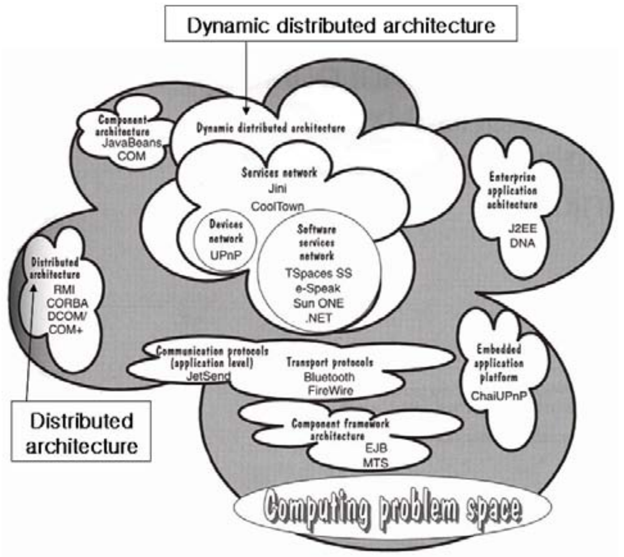


Fig. 1. Computing problem space of middlewares[8].

Recently there has been released the concept of URC(Ubiquitous Robotic Companion), which is ubiquitous service robots that provide users with the services they need, anytime and anywhere in ubiquitous computing environments[9]. This is the same vision to ours. In this paper, however, we propose to use the UPnP technology as a unified framework for robot integration and ubiquitous robot services. We show that the UPnP does not result in performance degradation by showing that the performance of the UPnP is similar to that of the TAO CORBA[10]. We also explain the reason why the UPnP architecture has good features and is appropriate for both of internal integration and external control of robot in ubiquitous computing environment. Lastly, this paper shows some experiments for it.

2. UPnP

2.1 Overview

The UPnP(Universal Plug and Play) architecture offers pervasive peer-to-peer network connectivity of PCs, intelligent appliances, and wireless devices. The UPnP architecture is a distributed, open networking architecture that leverages TCP/IP and the Web to enable seamless proximity networking in addition to control and data transfer among networked devices[6]. It was originally developed by Microsoft, and now is being upgraded by UPnP forum. The UPnP forum defines and publishes UPnP device and service control protocols built upon open, Internet-based communication standards. The UPnP forum is supported by more than 800 companies. Since the major companies such as Microsoft, Intel, HP, Sony, and Samsung Electronics leads the forum as the steering committee members, the UPnP is getting power as a device network architecture at home and office. Actually the UPnP is also adopted as the base protocol of device discovery and control in the home networked device interoperability architecture of the DLNA(Digital Living Network Alliance)[11]. The DLNA is a consortium of CE(Consumer Electronics), mobile, and PC companies, which is formerly called DHWG (Digital Home Working Group). The DLNA is working on delivering an interoperability framework among digital appliances on the home network, such as digital TVs, digital set-top boxes, mobile phones, PDAs, and PCs. The DLNA is currently supported by more than 280 companies.

2.2 Components

The main components of the UPnP architecture are Devices, Services, and Control Points. Fig. 2 shows the basic relation among Device, Control Point, and Service.

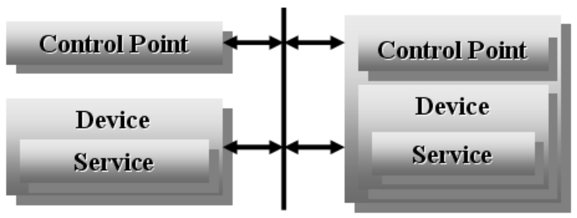


Fig. 2. Basic relation among Device, Service and Control Point.

A UPnP Device is an entity that provides Services. A UPnP device is a container of services and nested devices. A UPnP Device can contain zero or more Services. A Service is a unit of functionality implemented by a Device. A Service exposes actions and models its state with state variables. A Control point is a service requester. A Control Point in a UPnP network is a controller capable of discovering and controlling other devices.

2.3 Protocol Stack

The UPnP architecture defines a base set of standards and conventions for describing devices and the services they provide. It is designed to bring easy-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged networks. It provides the service oriented architecture, which is the base of emerging Web service technology as well. It uses standard protocols and web technologies such as TCP/IP, UDP, SSDP, SOAP, GENA, HTTP and XML. Fig. 3 shows the architecture of the protocol stack of the UPnP.

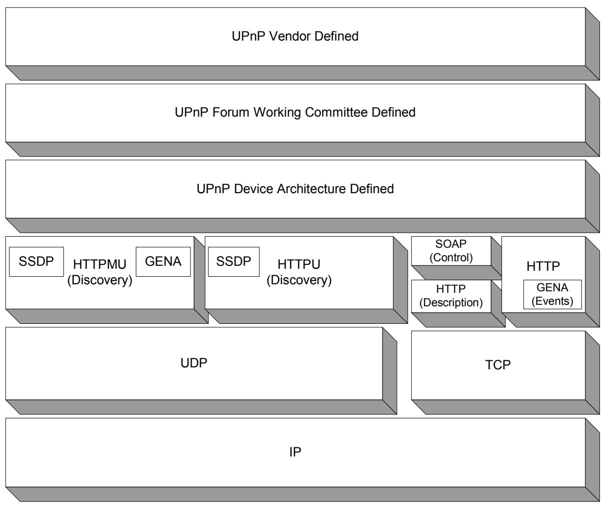


Fig. 3. The UPnP protocol stack.

Here the UPnP vendors, UPnP Forum Working Committees and the UPnP Device Architecture document define the highest layer protocols used to implement UPnP. Based on the device architecture, the working committees define information global to specific device types

Table 1. Caption of a sample table.

Steps	Description
Addressing	Control point and device get addresses
Discovery	Control point finds interesting device
Description	Control point learns about device capabilities
Control	Control point invokes actions on device
Eventing	Control point listens to state changes of device
Presentation	Control point controls device or views device status using HTML UI

2.4 UPnP Networking

The UPnP provides support for communication between Control Points and Devices. On top of the open, standard, Internet based protocols, the UPnP defines an architecture to handle the networking operation. The UPnP networking is accomplished through 6 steps: Addressing, Discovery, Description, Control, Eventing, and Presentation. Table 1 explains the 6 steps of the UPnP networking.

3. Performance Comparison

3.1 TAO CORBA

CORBA is a middleware architecture of the Object Management Group's(OMGs) standard for language and platform independent distributed computing. It supports object oriented programming in distributed computing environment[1]. CORBA is an acronym for Common Object Request Broker Architecture. As its name means, the object request broker(ORB) is the basic engine for this architecture. The ORB, known as the object bus, is responsible for locating server objects, and transporting method invocations and results to and from the server.

TAO(The ACE ORB) is an open-source high-performance, real-time CORBA ORB end system. It was developed for distributed realtime and embedded(DRE) systems by the TAO project[10]. So, there was a case to apply TAO to robot development[4]. TAO is based on ACE[12] and extends CORBA by adding realtime I/O subsystem and high speed network interface. It allows applications to manage various resources such as communication and memory resources. It also provides a run-time scheduler to enforce QoS property. It adopts and provides many services of realtime OSes.

3.2 Comparison of Response Time

When the UPnP architecture is applied to software components, each software component can be mapped to a Control Point or a Device. Since a complex robot with many functionalities would have as many software components as the number of functionalities. As the number of Devices and Control Points increases, there could be a possibility of performance degradation since the UPnP was not devised for software integration. So, we implemented UPnP Devices and Control Points, and checked the response time. At the same time, we also implemented client and server components with the TAO CORBA, and compare the response time. In the experiment, we implemented dummy client(Control Point) and server(Device) components. And we increased the number of clients and server components from 10 to 80 each, and measured the response time.

In the first experiment, we located the devices and the control points on a single computer. Figure 4 shows the configuration of the experiment. The computer had a Pentium 4, 1.7GHz CPU and 256MB memory. The operating system was Redhat 9.0 with the kernel upgrade to version 2.4.26. The clients were programmed to send a dummy request every 50ms, and to measure and save the response time. With increasing the number of client(and server) from 10 to 80, we run the program for 20 min, and analyzed the result. Figure 5 gives the result.

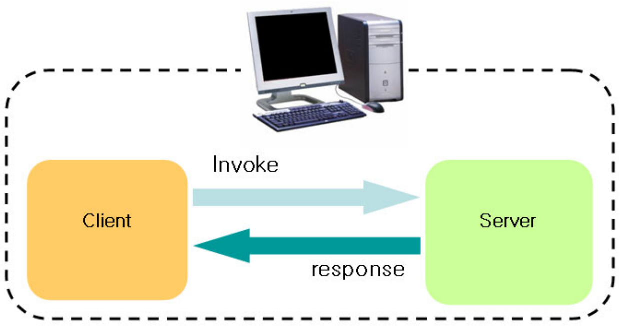
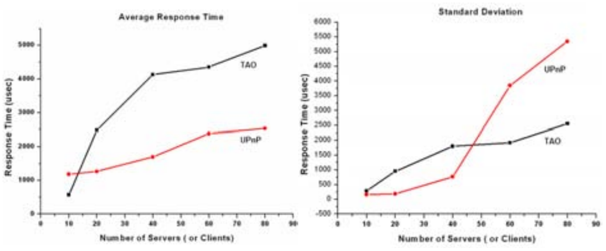


Fig. 4. The configuration of the single computer experiment.



(a) (b)

Fig. 5. (a) Average response time and (b) standard deviation for single computer experiment.

As can be seen in Fig. 5, the response time shows a little change along the increase of the number of devices. The average is almost 5ms for TAO CORBA and 2.4ms for UPnP when the number of devices is 80. The standard deviation is less than 2.5ms for TAO CORBA and 5.5ms for UPnP. In the single computer experiment, the UPnP showed better performance in average time and worse performance in standard deviation than the TAO CORBA.

In the next experiment, we used two computers. We located the clients on one computer, and the server components on the other computer. The client computer had a Pentium 4, 2.8GHz CPU and 512MB memory. The operating system was Redhat 9.0 with the kernel upgrade to version 2.4.26. Figure 6 shows the configuration of the experiment.

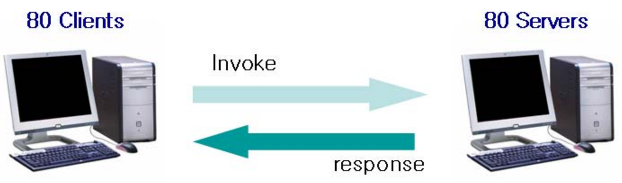
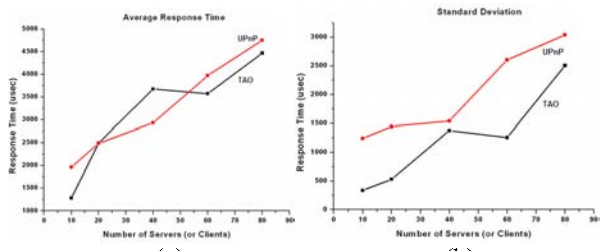


Fig. 6. The configuration of the two computer experiment.

We used the same client and server components as before, and measured the response time as we increased the number of client and server components. Figure 7 shows the result.



(a) (b)

Fig. 7. (a) Average response time and (b) standard deviation for single computer experiment.

In this case, the average is almost 4.5ms for TAO CORBA and 4.7ms for UPnP when the number of devices is 80. The standard deviation is less than 2.5ms for TAO CORBA and 3ms for UPnP. The graph shows that the performance of the UPnP is similar to that of the TAO CORBA though it shows a little bit more standard deviation. From the both of experiments, we could conclude that there is no severe performance degradation when the UPnP is used for software components, and that the performance of UPnP is similar to that of TAO CORBA.

4. UPnP and Ubiquitous Robot Control

Though the UPnP showed similar performance to the TAO CORBA, it has better features for dynamic distributed computing environment than CORBA. First of all, the UPnP has automatic discovery and configuration mechanism. In the UPnP community, it is called zero-configuration. The Addressing, Discovery, and Description of UPnP networking steps are related with automatic discovery and configuration of Control Points and Devices. It is the strong point that makes the UPnP to be appropriate for dynamic computing environment. Every time devices are connected to the network, there is no need for users to configure the devices. Since ubiquitous computing environments will be mostly based on ad hoc networks that are spontaneous, completely distributed, and dynamic, the UPnP is quite appropriate for the environment.

In the view point of robots, automatic discovery and configuration makes it easy to integrate software components internally. The software components can participate in integration dynamically. If there is a task planner or an intelligence component, it can configure other software components dynamically according to the goal of services it wants to provide. This point would be essential for implementation of intelligence. The intelligence component can reside even outside of a robot since software components can cooperate with each other regardless of the location. In this case, the network connection could not be permanent, and the robot should work on the ad hoc network. It makes the UPnP technology to be more attractive since it was designed for dynamic computing environment.

In the recent concept of the URC(Ubiquitous Robotic Companion), some software components are located outside of robots. In this way, the price of robot can go down, and various services can be provided. The outside software components can be an intelligence component or service components that provide desired services by cooperating with the software components inside the robot. By the way, this URC configuration of robots is what the UPnP is quite appropriate for.

Software components can be located in any of internal environment of robot or ubiquitous computing environment, and they can be configured dynamically. Then, internal software integration of robot and software configuration for services in the URC concept are the same thing. Thus, it would be better to have a unified framework for it. The UPnP architecture is a good candidate for it. Figure 8 shows an example of the configuration with the UPnP technology. Each small block represents a software component, and it is mapped to a UPnP Control Point or a Device or a combination of them. The 'internal robot environment' means inside of the robot. The figure shows the software integration can be done regardless of the location of software components. The integration framework is unified. In this way, internal software components of robot can be controlled directly from outside, too.

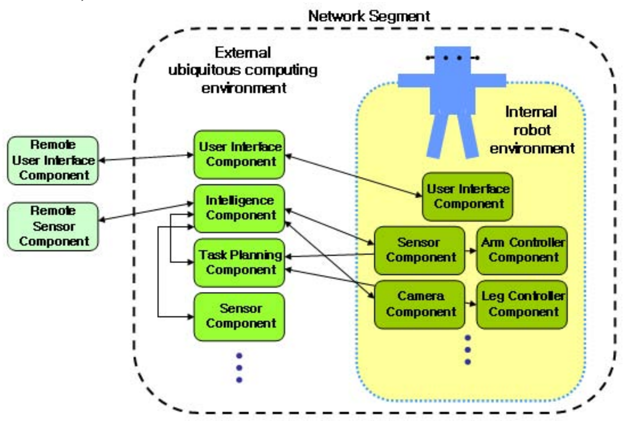


Fig. 8. An example configuration with the UPnP technology in ubiquitous computing environment.

Another good feature of the UPnP is that it has 'Presentation' step. Each UPnP Device can have a web page, and can use it freely. It can help to build user interface. If we develop a software component so that it has a web page with user interface for its services, then users can access the software component through the internet, and control it. In other way, the web page can be used to check the status of the component. Users or other high level software component can check the healthiness of the component through the web page. Though the UPnP limits the operation within a network segment, the 'Presentation' mechanism provides a way to circumvent the limitation. The figure 8 shows this concept with 'remote user interface component.'

5. Experiments

In the first experiment, we built a Lego robot with a camera on it. The goal of robot control is to make the robot to follow a marker. In more detail, we wanted to make the robot to maintain the direction and the distance to the marker within some ranges. For instance, if the marker goes out of the camera view to the right, the robot should turn right to the marker, and vice versa. If the marker moves far away from the robot, the robot should move toward the marker so that it can maintain the size of the marker in the camera image in a certain range. If the marker moves toward the robot, the robot should move away from the marker. We used Lego RCX controller to build the robot. We assigned a computer to control the motion of the Lego robot. We called it 'Main computer.' The Lego RCX could be controlled using IR(Infra red) signal. The Lego tower was USB-to-IR signal converter, and it was connected to the computer. We mounted a 1394 camera on the robot. We also assigned another computer to the camera for image processing. We called it 'Vision computer.' Firstly, we implemented 3 software components: Motor control, Vision tracker, and Planner. The Motor control component controlled the motion of the Lego robot. The Vision tracker tracked the marker, and generated the direction that the robot had to move to. The Planner issued start and stop commands. Though its name was planner, it did not make any plan, rather its role was user interface. Later, we added Video renderer and UPnP AV components to see the camera view on the computer. Figure 9 shows the architecture of the system with the software components. Each component was developed as a UPnP component. For instance, the planner was a Control Point while the vision tracker and the motor control were combinations of Control Point and UPnP Device.

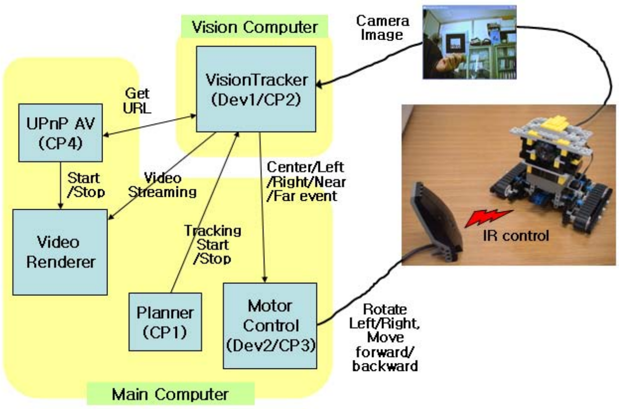


Fig. 9. The architecture of the Lego robot control system.

The robot worked quite well, and followed the marker. Though the camera was attached to the Lego robot physically, they could be separated. The software components were separated, too. But, they cooperated with each other very well though they were located on different computer. While implementing the software components, we felt that the integration was so simple and easy. The only thing we had to do was to wrap our programs with UPnP protocol stack and to place them on the computers. The UPnP wrapping was composed of defining services and combining action code with UPnP protocol code. It was easy, too.

In the second experiment, we developed a 'Robot Hello' program with UPnP components, and checked the usefulness of UPnP and possibility of ubiquitous robot control. The testbed robot consisted of 3 single board computers(SBC) in hardware, and they were connected each other with networks. The SBC's were based on Linux and adopted a realtime OS such as RTAI[13]. Figure 10 shows the appearance of the robot.

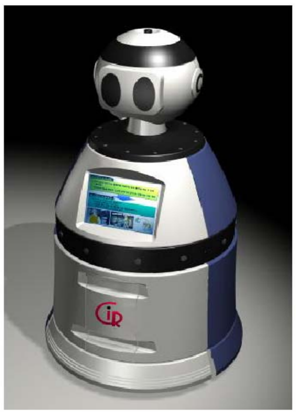


Fig. 10. The appearance of the testbed robot.

The goal of the Robot Hello program was to let a robot show a greeting expression to a man in front of it. The robot was supposed to move forward a little, to make a bow, to say hello, and to display 'Hello' characters on the display. The Robot Hello program was designed to have 5 software components. Each component was programmed to be UPnP Device or Control Point with our UPnP based robot middleware SDK. Figure 11 shows the relation of the UPnP components for Robot Hello program.

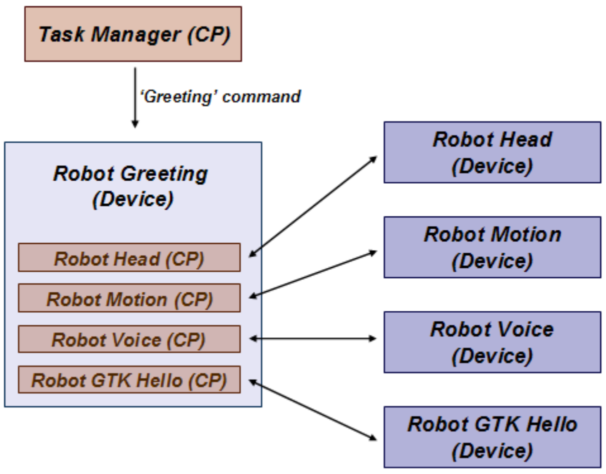
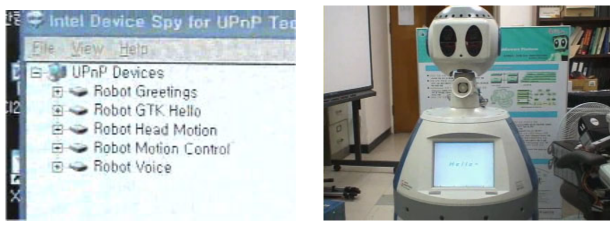


Fig. 11. The relation of UPnP components for the Robot Hello program.

The figure shows 6 components including the task manager component, and the relation between them. The 'Robot Head', 'Robot Motion', 'Robot Voice', 'Robot GTK Hello' components were programmed into UPnP Devices. The 'Robot Head' component controlled the motion of robot head. The 'Robot Motion' component controlled the movement of robot. The 'Robot Voice' component generated 'Hello' sound using voice synthesis when it received a command. The 'Robot GTK Hello' component was supposed to display 'Hello' characters on the display of the robot body. The 'Robot Greeting' component was transformed into the combination of UPnP Device and Control Point. We let the 'Robot Greeting' component include 4 Control Points inside it. The 'Robot Greeting' component acted as UPnP Device to a task manager. When we tested the program, we could find that the new UPnP-based Robot Hello components worked together without any problem. In addition, we tested the situation of dynamic computing environment by tearing down and reloading software components in realtime. In this experiment, dynamic tearing down and reloading of UPnP components did not cause any problem and showed smooth work. For example, if we unloaded the 'Robot Head' component, the program worked well without it. If we reloaded it again, the program detected it and worked well including head motion. Figure 12 shows two shots of the experiment. Figure 12(a) shows the installed the UPnP components. Figure 12(b) shows the robot was displaying the 'Hello' word on the front panel.



(a) (b)

Fig. 12. Experimentation with the testbed robot, (a) user interface that shows the S/W components in the robot (b) the robot displaying 'Hello' on its display panel.

In the experiment, we located the 'Task Manager' component outside the robot. We used another computer to run it, and let the computer communicate with the robot by wireless LAN. It did not pose any problem. We used a user interface program as the 'Task Manager'. Figure 12(b) is the screen shot of the user interface. With the user interface program, we could not only control the robot, but access other services of internal robot components such as 'Robot Head' and 'Robot Motion' as can be noticed in Fig. 12(a). So, we could see the possibility of UPnP as a unified framework for both of robot internal environment and ubiquitous computing environment with robots.

6. Conclusion

Ubiquitous computing environment will be mostly based on ad hoc networks that are spontaneous, completely distributed, and dynamic. Robots will be deployed like a TV or a refrigerator at home or at office in the ubiquitous computing environment. In the recent concept of the URC(Ubiquitous Robotic Companion), some software components are located outside of robots. Robots are supposed to cooperate with outside components to provide desired services.

By the way, in our view, internal computing environment of robots should be distributed and dynamic in order to achieve intelligent behavior. A robot will need to configure software components dynamically according to its goal. So, we need appropriate middleware and framework for efficient internal and external integration of software components of robots.

In this paper, we have showed that the UPnP technology could provide a unified framework for it. The UPnP defines an architecture for pervasive peer-to-peer network connectivity of intelligent appliances. While the UPnP showed similar performance as the CORBA, it had good features such as automatic discovery and configuration for dynamic distributed computing environment. Using the UPnP technology, we could access and control software components uniformly regardless of location. It was good for handling dynamic change of software configuration, too. While the UPnP is getting more power as a good candidate for middleware of home and office network, the UPnP is a good candidate for a unified framework for future robots, too.

References

[1] CORBA, www.omg.org
[2] H. Hirukawa, F. Kanehiro and S. Kajita, "OpenHRP: Open Architecture Humanoid Robotics Platform," Int'l Symp. Robotics Research, 2001.
[3] E. Li, D. Chen, H. I. Christensen and A. Oreback, "An Architecture for Indoor Navigation," Int'l conf. on Robotics and Automation, vol. 2, pp. 1783-1788, April, 2004.
[4] H. Utz, S. Sablatnog, S. Enderle and G. Kraetzschmar, "Miro – Middleware for Mobile Robot Applications," IEEE Tr. on. Robotics and Automation, vol. 18, no. 4, pp. 493-497. August, 2002.
[5] Sang Chul Ahn, Jin Hak Kim, Kiwoong Lim, Heedong Ko, Yong-Moo Kwon, and Hyoung-Gon Kim, "UPnP Approach for Robot Middleware," ICRA 2005, Apr. 2005.
[6] UPnP, www.upnp.org
[7] Web Services, www.w3.org/2002/ws
[8] S. Ilango Kumaran, Jini Technology: An Overview, Prentice Hall, 2002, p.307.
[9] Young-Guk Ha, Joo-Chan Sohn, Young-Jo Cho, and Hyunsoo Yoon, "Towards a Ubiquitous Robotic Companion: Design and Implementation of Ubiquitous Robotic Service Framework," ETRI Journal, vol. 27, no. 6, pp.666-676, Dec. 2005.
[10] TAO, www.cs.wustl.edu/~schmidt/TAO.html
[11] DLNA, www.dlna.org
[12] ACE, www.cs.wustl.edu/~schmidt/ACE.html
[13] RTAI, Realtime Application Interface, www.aero.polimi.it/~rtai/