Connection Methodology for Two Ubiquitous Robot Spaces

– Connection of RT-Middleware and CAMUS

Hyun Min Do, Yong-Shik Kim, Bong Keun Kim,
Tamio Tanikawa, Kohtaro Ohba

Intelligent Systems Research Institute
AIST, Tsukuba, Ibaraki, Japan

{hyunmin.do, yong-shik.kim, bk.kim, tamio.tanikawa,
k.ohba}@aist.go.jp

Janarbek Matai, Young-Ho Suh, Hyoung Sun Kim,
Jae-Yeong Lee, Wonpil Yu

U-Robot Research Division
ETRI, Daejeon, Korea

{janarbek, yhsuh, kimhs, jylee, ywp}@etri.re.kr

Abstract—This paper presents a connection method for two ubiquitous robot spaces. The ubiquitous robotics is recently proposed and many methods are suggested to implement this concept. In many cases, software modularization and middleware technology are adopted. However, different middlewares are used for each case since there is no standard for such a technology. Thus, the implemented robotic space keeps a ubiquitous concept only under its own standard. To remedy this problem, the method of connection for two middlewares is proposed. In the proposed method, the target middlewares are RT-Middleware of Japan and CAMUS of Korea. By making a bridge between two middlewares, the service provided by RT-Middleware side can be available to CAMUS side and also the service provided by CAMUS side can be used by RT-Middleware side. Experimental results are included to verify the performance of the proposed method.

I. INTRODUCTION

The ubiquitous robotics, as a new breed of robotics, is recently proposed and many studies have been made to realize this concept. In the ubiquitous robotics, a robot will provide us with various services by any device through any network at any place anytime[1]. In this scheme, a distributed object technology is one of the key elements. All resources will be distributed in the space and form a ubiquitous robot space. Each object will do its own function and the result of each function will be connected to each other through network. Thus each object can take the needed information and can get the needed action. It can be sensor data or some action of actuator including the movement of a specific robot. Through such a distributed computing, a robot can get needed information and provide required services to human.

To implement such a ubiquitous robot space, many schemes are proposed and implemented[2], [3], [4], [5], [6], [7]. The main problem is how to develop distributed objects and how to connect them. This can be considered as the problem of designing software module and middleware. Since the purpose and the requirement are different from each method, the implemented robotic space has its own solution according to given conditions. It means that several robotic spaces may have incompatible specification each other and the concept of ubiquity is valid only in the each own space even though all are designed for ubiquitous spaces. For

example, AIST(National Institute of Advanced Industrial Science and Technology) of JAPAN has implemented u-RT space based on the RT-Middleware, which is distributed component middleware for RT (Robot Technology)[2], [8], and ETRI(Electronics and Telecommunications Research Institute) of KOREA has implemented URS(Ubiquitous Robotic Space) based on CAMUS (Context-Aware Middleware for URC (Ubiquitous Robotic Companion) Systems)[4]. Since each space has been implemented using different methodology, the information of one space cannot be available in other space. In the view of ubiquitous concept, this is somewhat distorted result. To overcome this problem, it is needed to connect these several spaces which were designed using different schemes. Since there are so many methods, however, it is impossible to develop a common method to connect all of them. In this paper, therefore, the connection methodology for two ubiquitous robot spaces of Korea and Japan is proposed as one attempt for the connection of ubiquitous robot spaces.

This paper is organized as follows. In section II, we will briefly introduce two middlewares for ubiquitous robot space of Korea and Japan. In section III, the connection scheme of two middlewares will be proposed and the detailed implementation method will be described in section IV. In section V, experimental results will be suggested and finally, conclusions will be made in section VI.

II. MIDDLEWARES FOR UBIQUITOUS ROBOT SPACE

In this section, two kinds of schemes to build a ubiquitous robot space will be introduced. One is the RT-Middleware and the other is CAMUS.

A. RT-Middleware[2], [8]

The purpose of RT-Middleware is to establish basic technologies for integrating robot systems with a new function easily by modularized software components which constitutes robot systems and finally to establish a systematic robot system design theory derived from the knowledge of component-oriented robot system integration. A design flow and goal of RT-Middleware is described in Fig. 1.

RT-Middleware indicates an aggregate of several necessary

Fig. 1. Goal of RT-Middleware

elements to make and manage RT component, which is basic functional unit of RT-Middleware based systems.

- **RT component framework:** This is the framework to make a RT component. An essential software part of these RT function elements is called as core logic. RT component framework covers this core logic with common interface and this is the mechanism to make RT-Component in unifying way. Fig. 2 shows this procedure.

Fig. 2. RT Component Framework

- **RT middleware:** In implementation, a RT component is a simple class or a shared library of that. It's the role of the RT middleware to carry out the action defined by a RT component. RT middleware makes an instance, manages a life cycle, executes an action, and bridges several services of a RT component.
- **Basic RT component group:** A RT component is the module which can be dealt with as a reusable software part. If we reuse RT components which are already made, only a minimum component will be needed additionally when making a new robotic system and thus it is possible to make a new robotic system easily.
- **Library, Basic service group, Basic tool group**

RT middleware and a RT component cannot be limited to some specific software. OpenRTM-aist is a loaded example of the RT middleware developed by AIST and it is implemented based on CORBA. OpenRTM-aist is consisted of component framework and the middleware to manage and

carry out a component. Also it supports a data port and service port as a communication port.

B. CAMUS[4]

The purpose of the CAMUS is to provide context-aware services to network-based robots. Generally, service robots need three major functions; sensing, planning and action. The basic concept of URC is to distribute those three functions through the network. It enables robots to reduce costs and improve service qualities by taking full advantage of IT infrastructure including internet and wireless network.

The CAMUS is composed of three parts: CAMUS Server, Planet and Service Agent Manager(SAM). The CAMUS server is the server-side framework to manage the information delivered from SAM, generate and disseminate appropriate events according to the context changes, and finally execute server-based robot tasks. The planet is the communication framework between SAM and the CAMUS server. ETRI implemented the planet to meet the following requirements. Firstly, the light-weight protocol is needed in order to minimize the size of the message payload. Secondly, communication framework should not be bounded to a specific platform and programming language. Thirdly, the framework should not only support disconnected operations but provide application developers with a transparent reconnecting mechanism. Finally, the communication framework also needs to consider the efficient way to exchange large size multimedia data such as voice and images.

The SAM is the robot-side framework to send robot's sensing data to the CAMUS server and receive control commands from the CAMUS server. SAM is a software container of service agents (sensors or actuators) that reside in a robot or a location, and manages and controls its service agents. SA is software component that represents a physical (or logical) device embedded in the environment or a robot. It acquires information from sensors and sends them to the CAMUS server. Also, it receives control commands from CAMUS server and controls actuators. Fig. 3 shows the architecture of SAM.

Fig. 3. Architecture of SAM

III. CONNECTION SCHEME

A. Concept

In this section, the connection scheme of two ubiquitous robot spaces will be described. Two robotic spaces are u-RT space of Japan and URS of Korea. u-RT space is

implemented based on RT-Middleware. A service is implemented by service provider component and service consumer component. When service consumer component requests a service, service provider component provides a requested service. A basic unit of service is RT component. On the other hand, URS is based on CAMUS server/client system. A Service is provided as a form of service agent (SA) and SAs are managed by service agent manager (SAM). Through CAMUS server, they are available to CAMUS clients. SA is a basic unit of service and loaded to CAMUS server by SAM.

For the connection, two methods should be considered. One is that the service is provided by RT service provider component and used by CAMUS clients. In u-RT space, all services are implemented and provided by RT service provider component. For example, several sensors distributed in the environment are implemented as RTC. Then, a robot can use this information through RT service consumer component. However, when a robot of URS moves into u-RT space and wants to use this service, that is impossible because a robot has only CAMUS client interface. Thus, the connection from the RT service provider component to CAMUS client is required. The other one is that the service is provided by SAM/SA and used by RT service consumer component. This is the opposite case. All services are implemented as SA and available through CAMUS server. If a robot of u-RT space moves into URS, a robot cannot use provided services because a robot has only RT consumer component interface. Thus, the connection from CAMUS server to RT service consumer component is needed. Fig. 4 shows the connection diagram of each case, where (a) shows the first case and (b) shows the second case. Dark box represents the connection part in both cases.

B. Structure

1) Connection of RT consumer component and SAM:

Firstly, the connection method between the RT service provider component and CAMUS client is described. In this case, the connection of RT component as service provider and SAM as service container is proper connection method with consideration of the structure of both systems. Since SA is registered on CAMUS server and managed by SAM, RTC will also be dealt as SA. The flow of development is as follows.

- 1) Design of RT consumer component
- 2) Definition of interface as PLANET IDL (CORBA IDL → PLANET IDL)
- 3) Development of SAM interface using PLANET IDL
- 4) Design of Proxy class
- 5) Connection of Service Consumer and SAM
- 6) Registration of RTC service on CAMUS server

Fig. 5 describes overall flow of connection. Fig. 5(a) shows each structure of RTC and SAM/CAMUS server. RTC is based on CORBA and SAM/CAMUS server is based on Planet. To access RT provider component, the pointer *m_service_p* which indicates the provider component object is used. SAM has several SAs and it loads them to CAMUS server based on PLANET. Fig. 5(b) shows a connection

Fig. 4. Connection diagram

process. By connecting RT consumer component and SAM, the role of RT consumer component is changed. Consumer component activates SAM and transfers the pointer information to SAM. Then, SAM can access RT provider component using this pointer. After SAM is started and service of RT provider component is registered on CAMUS server, the flow for service access will be changed like Fig. 5(c).

2) Connection of CAMUS client and RT provider component: Next, the connection method between SA/SAM and RT service consumer component is explained. In this case, CAMUS client program is wrapped by RTC framework and converted to service provider component. Then, RT service consumer component can use the services of SA. This way of connection is to make new RT service provider component. Core logic is CAMUS client program. The important work is to translate the PLANET IDL into CORBA IDL. If a CORBA IDL is given, a component can be easily made by a code-generation tool RTC-template. After the code of a RT service provider component is built, each interface function should be implemented using CAMUS client program. Each function of RT service provider does a role of proxy. It will just call a corresponding function of SA. Also, the procedure of accessing CAMUS server and getting service information of SA should be implemented in RT service provider component. The flow of development is as follows.

- 1) Design of RT provider component
- 2) Definition of interface as CORBA IDL (PLANET IDL

Fig. 5. Overall flow of connection

→ CORBA IDL)

- 3) Development of interface of RT provider component
- 4) Implementation of function to access CAMUS server
- 5) Activating component and registration on CORBA naming service

IV. IMPLEMENTATION

A. Design of RT consumer component

We assume that RT provider component is already developed and provided as a library. Then, this service can be used by creating the service consumer component. The interface is described in CORBA IDL file. Using rtc-template command, which is a tool used to generate skeleton RT components, the consumer component can be easily created. The connection between provider component and consumer component is done with the servant object on CORBA. Usually, the consumer component is composed of a series of user command to call service provider component. However, this job will be done by the client of CAMUS server in the proposed case. The role of the consumer component is to transfer the information of the servant object to SAM and to make a bridge between the provider component and SAM. To do this, SAM is defined and activated in the consumer component.

B. Design of proxy class

To handle the provider component as SA in the side of SAM, a standard interface needed for SAM should be implemented like other SAs. Since the interface of the provider component is described in CORBA IDL, the interface should be redefined as PLANET IDL. Example of CORBA IDL is shown in Fig. 6. This example implements three functions. One is for navigation of robot to target position, second one is for getting current position of robot and last one is for stopping robot. Example of PLANET IDL corresponding to CORBA IDL is shown in Fig. 7. Then, proxy class

```

Interface service_provider
{
    void navigateToXYT(target_xpos, target_ypos, target_theta, target_velocity);
    void getBodyPosture(xpos, ypos, theta);
    void stop();
};

```

Fig. 6. Example of CORBA IDL

```

interface Navigation
{
    async void navigateToXYT(xpos, ypos, theta, velocity);
    BodyPosture getBodyPosture();
    void cancelAll(string details);
};

```

Fig. 7. Example of PLANET IDL

should be designed for SAM, which is an implementation of interface function defined in PLANET IDL. SAM will treat this proxy class as usual SA and register its services to CAMUS server. However, proxy class does not provide services, but does a role of proxy. Each function of proxy class will call the corresponding function of RT provider component. Thus, if SAM calls the function of proxy class, then the corresponding function of RT provider component will be called. This is the role of proxy class.

C. Connection to CAMUS server and registration of RTC service

Now, the connection bridge between RT provider component and SAM is made. In the middle of that bridge, there is RT consumer component. Here, the role of the consumer component is changed. Usually, the consumer component is composed of a series of user command to call services from the provider component. However, user command will be given from the client program of CAMUS server. The modified role of the consumer component is to start SAM and to transfer to SAM the servant object which indicates the provider component. The flow of connection is as follows:

- 1) RT provider component and RT consumer component are connected and activated
- 2) SAM is started when RT consumer component is activated.
- 3) SAM connects to CAMUS server and registers services of RT provider component.

- 4) CAMUS client gets the service information from CAMUS server.
- 5) Function call of client is delivered from CAMUS server to SAM through PLANET.
- 6) Function call of client is delivered from SAM to RT provider component through CORBA.
- 7) Service will be provided from RT provider component.

V. EXPERIMENTS

To implement and verify the proposed connection scheme, some experiments were designed. A scenario of experiment is to make a mobile robot to navigate to target position which is given by a user and also to notify its position to a user. The robot gets its position using StarLITE sensor system[9] which gives a global position. StarLITE sensor system was developed by ETRI, Korea. StarLITE is composed of wireless tags and a detecting camera. Tags are attached on the ceiling and the detecting camera is loaded on a mobile robot. As a mobile robot, Pioneer 3DX of MobileRobots Inc. was used. Fig. 8 shows the P3DX mobile robot with StarLITE detecting camera part. Two kinds of experiments were established. First one is to run a square path and second one is to change a speed of robot during running. Fig. 9 shows the test space and a square path which a robot runs. Since the floor has a lattice pattern, the position of a robot can be easily distinguished. A robot moves to $(x, y) = (1.5, 1.0)$ firstly and moves to $(x, y) = (1.5, 3.0)$, $(x, y) = (5.0, 3.0)$, $(x, y) = (5.0, 1.0)$ and $(x, y) = (1.5, 1.0)$ sequentially with 500mm/sec . Second experiment is to change a speed during running a straight line. Changing order is as $200\text{mm/sec} \rightarrow 800\text{mm/sec} \rightarrow 500\text{mm/sec} \rightarrow 0\text{mm/sec}$. Same scenario is implemented with two methods. One is with RTC and the other is with the connection scheme of RTC and CAMUS like Fig. 4(a).

Fig. 8. P3DX mobile robot with StarLITE detecting camera

A. RTC implementation

First thing to do is to make StarLITE sensor system and P3DX as RTC. StarLITE provided the position information to P3DX and P3DX navigated to target position using the provided position information. Also, P3DX provided its position to consumer. Thus two providers and two consumers were

Fig. 9. Test space with a square path

implemented. Actually, P3DX module has both consumer and provider and they were implemented on one module. Through this implementation, we could see that robot moved according to the target scenario. User command to call services was conducted in consumer component. Some example codes for that is described in Fig. 10. Also, Fig. 11 shows the trace of a robot.

```
// running nLoop
for(i=0; i<nLoop; i++)
{
    m_p3service0->navigateToXYT(TGx[i], TGy[i], TGt[i], TGv[i]);
    while(1) {
        m_p3service0->getBodyPosture(x, y, theta);
        if(GetAchievedGoal==true)
            break;
    }
}

// changing speed
m_p3service0->navigateToXYT(TGx, TGy, TGt, TGv1);
m_p3service0->navigateToXYT(TGx, TGy, TGt, TGv2);
```

Fig. 10. Example codes for SAM

Fig. 11. Trace of a robot in RTC implementation

B. Connection of RTC and CAMUS

Connection procedure is just as that explained in previous section. By following that procedure, RT consumer

component was designed. Since the interface for SAM was already defined as NavigationProvider class, it was used in the implementation of proxy class. When RT provider component and consumer component was connected and activated, SAM was also started and connected to CAMUS server. Then client of CAMUS server began using services. On this experiment, we tested three functions and verified that those services were well provided. Since a client code was same as the case of using existing SAs, a client could not recognize whether this service was provided by SA or by RT provider component. In this case, we could also see that robot navigated successfully according to the target scenario. Fig. 12 shows example codes of CAMUS client program and Fig. 13 shows the trace of a robot.

```
// running nLoop
for(i=0; i<nLoop; i++)
{
    nav->navigateToXYT(TGx[i], TGy[i], TGt[i], TGv[i]);
}

while(1) {
    m_p3service0->getBodyPosture(x, y, theta);
}

.....
// changing speed
aop = nav->navigateToXYT(TGx, TGy, TGt, TGv1);
aop->cancel();
aop = nav->navigateToXYT(TGx, TGy, TGt, TGv2);
```

Fig. 12. Example codes for CAMUS client

Fig. 13. Trace of a robot in connection of RTC and CAMUS

VI. CONCLUSIONS

This paper has presented the necessity and the methodology of the connection of ubiquitous robot spaces. Two ubiquitous robot spaces were designed based on the distributed object technology, but different methods to mediate the distributed resources. One is the RT-Middleware, which was developed as a common platform for the RT technology by AIST, JAPAN and the other is the CAMUS, which was developed as a framework of providing context-aware services to network-based robots by ETRI, KOREA. Thus, the connection scheme of two methods has been proposed

to connect the ubiquitous space between two countries and thus the user of one space can use same services developed in other spaces without changing the interface.

Two kinds of approaches were considered. One is to make the service provided by RT-Middleware side be available to the user of CAMUS side and the other is to make the service provided by CAMUS side be used by the user of RT-Middleware side. For the former, RT provider component was got to be dealt as SA by connecting RT consumer component and SAM. For the latter, CAMUS client program was wrapped by RTC framework and dealt as RT provider component. In both cases, a bridge for connection was made and through this, services of both robotic spaces became available to all users of both spaces. It means that the service space which is accessible to a user is enlarged into the space which adopts a different standard.

The purpose of the connection is to maintain the ubiquity regardless of the specification of each ubiquitous robot space. To do this, most important thing is to define the interface function which can be applied to all spaces. This can be achieved by the definition of IDL file in this paper. Such a standard definition of services can be considered as a future work.

ACKNOWLEDGMENTS

This work was supported by the IT R&D program of Korea MIC (Ministry of Information and Communication) and IITA (Institute for Information Technology Advancement). [2005-S-092-02, USN-based Ubiquitous Robotic Space Technology Development]

REFERENCES

- [1] J.-H. Kim, Y.-D. Kim, and K.-H. Lee, "The third generation of robotics: Ubiquitous robot," in *2nd Int'l Conf. on Autonomous Robots and Agents*, Palmerston North, New Zealand, Dec. 2004, pp. 1–7.
- [2] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon, "Rt-middleware: Distributed component middleware for rt (robot technology)," in *2005 IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems(IROS2005)*, Edmonton, Canada, Aug. 2005, pp. 3555–3560.
- [3] B. K. Kim, N. Tomokuni, K. Ohara, O. Lemaire, T. Tanikawa, K. Ohba, and S. Hirai, "Ubiquitous localization and mapping for networked robots in the informative space," in *2006 Int'l Symp. on Flexible Automation*, Osaka, Japan, July 2006, pp. 956–961.
- [4] H. Kim, K.-W. Lee, Y.-H. Suh, J.-M. Cho, and Y.-J. Cho, "Client/server framework for providing context-aware services to network based robots," in *16th IEEE Int'l Conf. on Robot & Human Interactive Communication(RO-MAN2007)*, Jeju, Korea, Aug. 2007, pp. 475–480.
- [5] M. Broxvall, B.-S. Seo, and W. Kwon, "The peis kernel: a middleware for ubiquitous robotics," in *Proc. of the IROS Workshop on Ubiquitous Robotic Space Design and Applications*, San Diego, USA, 2007.
- [6] R. B. Rusu, A. Maldonado, M. Beetz, and B. Gerkey, "Extending player/stage/gazebo towards cognitive robots acting in ubiquitous sensor-equipped environments," in *Proc. of the ICRA Workshop for Network Robot Systems*, Rome, Italy, 2007.
- [7] S. C. A. et al., "Upnp robot middleware for ubiquitous robot control," in *The 3rd Int'l Conf. on Ubiquitous Robots and Ambient Intelligence(URAI2006)*, Seoul, Korea, 2006.
- [8] (2007) Openrtm-aist. [Online]. Available: <http://www.is.aist.go.jp/rt/OpenRTM-aist/>
- [9] H. Chae, W. Yu, J. Lee, and Y.-J. Cho, "Robot localization sensor for development of wireless location sensing network," in *2006 IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems(IROS2006)*, Beijing, China, Oct. 2006, pp. 37–42.