Streaming Layer of Personal Robot s Middleware

Vitaly Li, Seongho Choo, Hyemin Shin and Hongseong Park

Dept. of Electrical and Computer Eng., Kangwon National University
192-1 Hyoja 2 Dong, Chuncheon, 200-701, Korea
Email: {vitaly, somebody, atlas, hspark}@control.kangwon.ac.kr
Tel: +82-33-250-6346 Fax: +82-33-242-2059

Abstract: This paper proposes streaming layer for personal robot's middleware. Under assumption that robot has open architecture, i.e. consists of modules created by different vendors and intercommunication between these modules is necessary, we have to consider that there are many different network interfaces. To make communication between modules possible it is necessary to develop new type of middleware. Such middleware has to support different platforms, i.e. OS, network interface, hardware, etc. In addition, it is necessary to implement effective interface between network and application in order to manage inter application communications and use network resources more effectively. Streaming layer is such interface that implements necessary functionality together with simplicity and portability. Streaming layer provides high level of abstraction and makes communication between distributed applications transparent as if are located in same module. With possibility of extension by user defined application interfaces it is suitable for distributed environments, i.e. module based architecture including small-embedded systems like as DSP board. To verify the proposed streaming layer structure it is implemented using C and tested.

Keywords: robot, middleware, streaming, application control

1. INTRODUCTION

Recently, many researchers have been studying the robots called a personal robot, which is mainly used at home result in having a small size [1]. In addition, the personal robot has an architecture that should support various technologies in order to be easily extensible when new functions will be developed and added to the robot capacities, without major perturbations of the already existing robot system. There have been studies about the architecture for robots [2-5], which is composed of heterogeneous hardware components called module.

However, there have been few studies about interfacing between modules in the module based personal robot. In [6], VME and Ethernet are used as interfaces in the robot system, which is separated into two networks of Ethernet and VME. If the network configuration is changed in this fixed robot system, the robot will become unworkable. And in [7], CAN is only used as an interface for robot system. But there are actually many kinds of interfaces for modules in the personal robot like as buses (VCM, PCI) and networks (IEEE1394 [8], USB [9], CAN [10], Ethernet, Wireless LAN, Bluetooth [11]). So we should find out what is the more efficient interface-type for personal robot modules, before it is used. The bus-type interfaces like as VME and PCI need a motherboard and sockets to connect modules. Therefore the size of the integrated modules becomes large or it can be limited in its size because of additional equipment. On the other hand it is possible to solve these problems if a network is used as an interface. Therefore the network-type interface is more efficient than bus-type one. As we have referred, there are many kinds of networks, which are potential interfaces for personal robot. So the interface should have open architecture because any network can be employed as an interface for module based personal robot.

To support one or more open interface for the personal robot effective middleware [12] structure is necessary. There are many kinds of middleware, which is used in present like as RMI [13], CORBA (Common Object Request Broker Architecture) [14], and DCOM [15]. Most of these middleware are based on TCP/IP [16] in order to access or communicate with remote distributed objects. Although ESIOP (Environment-Specific Inter-ORB Protocol) of CORBA can support networks based on non-TCP/IP, it cannot support several networks simultaneously. Therefore, these all middleware don't support heterogeneous interfaces. These current middleware don't seem to be suitable for supporting open and heterogeneous interfaces in personal robot. The middleware for the personal robot should be able to support heterogeneous interfaces.

Another challenge is portability of overall middleware with equally well functionality. So middleware has to be simple, light and scalable in order to be easily adopted for the specific platform such as embedded board, PC, handheld device.

In [17] it is proposed to divide middleware into two layers: Network Adaptation Layer (NAL) as the lower layer and Streaming Layer (SL) as the upper layer. Scope of this paper is SL as mediator between network interfaces and application. SL provide high level of abstraction and make communication between distributed applications transparent as if they are located in same module.

It is necessary to study about application management for module based personal robot that supports open platform. This paper suggests structure of Streaming Layer for personal robot's middleware and verifies performance using PC and DSP board connected via CAN bus. In section 2, we propose the middleware structure for personal robot. In section 3, we describe SL architecture. In section 4 we show implementation results followed concluding remarks in section 6.

2. MIDDLEWARE STRUCTURE

In this section, we define a middleware structure for the personal robot, as shown on fig. 1. This middleware is separated into network control level and application control level by the middleware core. Network level is composed of several network components, which are dependent elements on each network.

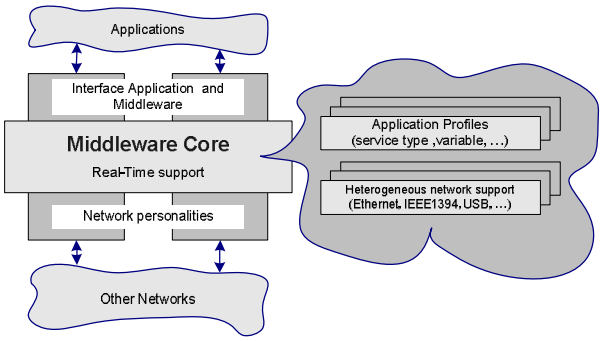

Figure 1: Overview of the Middleware Structure

In this structure, the topmost and bottom layers are completely independent, and one layer may be replaced without changing the other. In other words, components of network level can be changed or added without any perturbations on application level. This structure, which is mutually independent each layer is very suitable for the personal robot to support open and heterogeneous interface between modules. As shown in fig. 2, the middleware core consists of two layers. The lower layer is Network Adaptation Layer (NAL) and the upper layer is Streaming Layer (SL). NAL supports several open and heterogeneous interfaces. NAL manages open and heterogeneous networks in order to transmit messages between modules with stability and efficiency.

In order to communicate between modules, there should be an addressing strategy like as IP under the heterogeneous network environment.

In our module addressing strategy, an identifier that is called module identifier (MDID) is assigned automatically by NAL when the middleware of each module is initialized. MDID is a global location reference. After NAL makes local information, it informs the inter networks of the personal robot about its information. By this way, each module can make routing table. The routing table allows that the module addressing strategy and message routing. A routing table including special routing information because heterogeneous interfaces have different packet sizes and link speeds each other. Based on this information a data is delivered correctly from a source module to a destination module via one or more interfaces.

SL manages applications-transaction using middleware service and communication transactions, and marshals (or unmarshals) application data (or network data) to network data (or application data). It is described in details in next section.

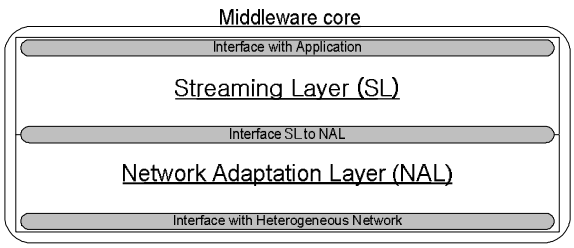

Figure 2: Middleware Core

3. STREAMING LAYER ARCHITECTURE

Streaming layer is application management layer of personal robot's middleware. It has to be able to provide specific service and must be simple, portable and reliable according to personal robot's tasks and necessity of adaptation for different types of hardware and software. The functional diagram of Streaming layer is shown in fig. 3.

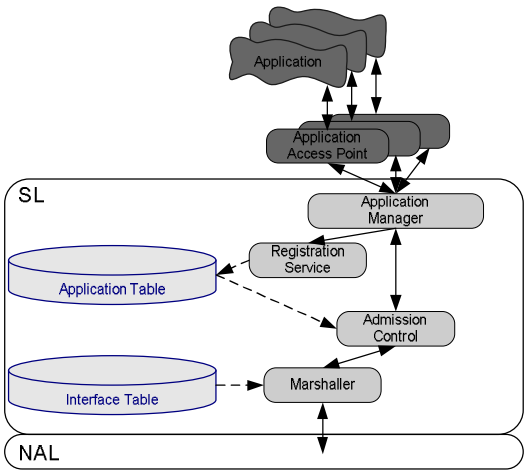

Figure 3: Streaming Layer Structure

The main functions of SL have to be supported by every network module. These functions are admission control, registration service and message representer or marshalling service. In addition there are two information structures in SL – application table and interface table. Application table keeps information about active applications in order to make communications between them possible. There are two kinds of applications from point of view of locality – local applications and remote applications. Whenever new personal robot's application occurs it must register itself at the middleware. The registration service generates global application identifier (GAID) which consists of two parts – local application identifier that is unique for this module and module identifier (MDID) that is unique within network. Then information about newly application is placed at local application table and propagated into network so every module adds this information at remote application table. The sequence of registration is shown in fig. 4.

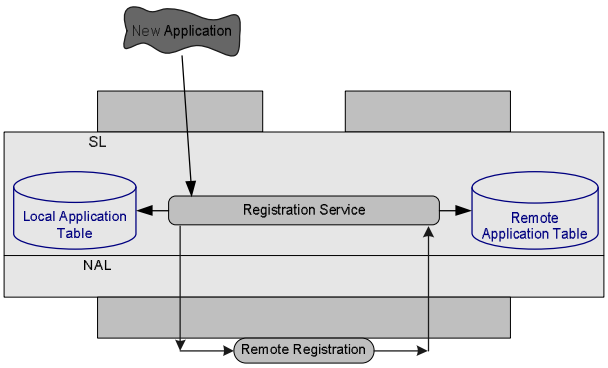

Figure 4: Local and remote registration in middleware

The interface table keeps information about services and data provided by particular application. Several common interfaces are embedded into middleware describing personal robot's specific applications. However SL provides possibility to register new interfaces defined by user application. There are constant mapping between application table and interface table established at registration stage. This lets applications to define whether requested data or service is available or not.

Admission control decides whether or not the remote middleware accepts the request of the caller application using

the remote application table. For example when an application wants to read remote object from another application, there should be target application and it should provide that object. If there is no target application or this object is not provided by existed target application, the request is unnecessary. Another function of admission control is to decide whether request has to be sent to remote middleware or it has to be sent to the application at same module. The functionality of Admission control is shown in figs. 5-6.

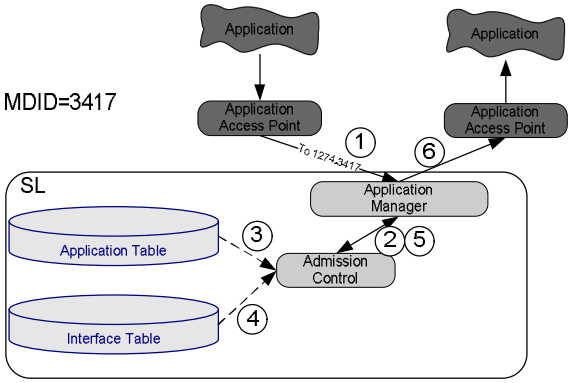

Figure 5: Local request

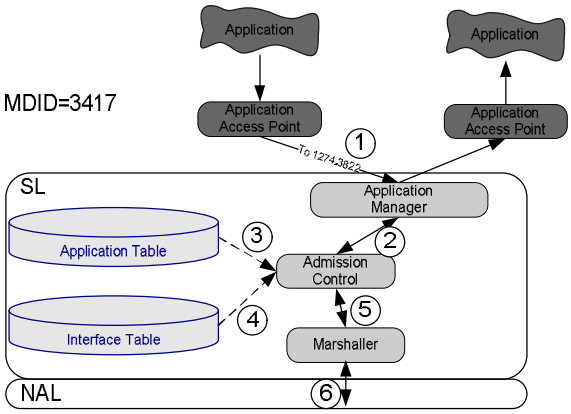

Figure 6: Remote request

Both figures show functionality of admission control on receiving outbound request from local application. Admission control checks both application table and interface table in order to decide whether this request could be proceed or not. The locality of the target application is known by GAID that is consisted by local application identifier and MDID as mentioned before. Hence, if MDID of source application match MDID of target application there is no need to check remote application table or marshal/unmarshal request.
The marshaller converts request sent from one application to another into format suitable for transmission over network as shown in fig. 7.

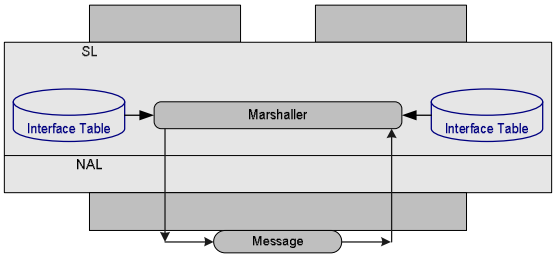

Figure 7: Marshalling/Unmarshalling

The conversion is made by using interface table associated with target application and sent to remote module that is defined from GAID. On the receiver side the reverse conversion has to be done and finally the request is delivered to the target application.
For instance, the middleware at the sending and receiving ends of the communication channel have to agree on the size and type of data. Therefore, the interface tables in both sides have to have same information about requested interface.

4. IMPLEMENTATION

In this section, we show implementation of the proposed SL. Our purpose in this test is to communicate between applications and show portability of the proposed middleware at all and SL in particular.
To verify the proposed SL structure, it is implemented using C with CAN network interface. In order to have fully configurable environment proposed by open source, Linux, with kernel version 2.4.x is used as an operating system (OS) for x86 based personal computer (PC). To show portability issues the embedded DSP board platform TMS320VC33 with 32bit processor and two network interfaces (RS232C, CAN) has been used. Middleware has been ported from Linux OS to the DSP environment according to specific conditions, such as single threading, limited memory and code optimization needs. The test configuration is shown in fig. 8.

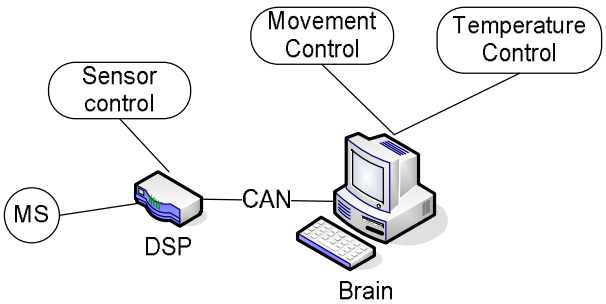

Figure 8: Test configuration

In our system “Sensor control” is a DSP’s application name. “Movement control” and “Temperature control” is Brain’s applications. DSP board has movement sensor connected that is denoted by circle with “MS” inside in fig. 8. That sensor is presented by DynaSight active sensor device. “Sensor control” application is associated with “Sensor – Movement” interface and has set of three real numbers to send which are coordinates of tracked object related to the sensor position.
SL of Brain module manages an application table which includes information of which applications are where they are, what kind of interface, as shown in fig. 9.

Figure 9: Application table of Brain Module

For instance, an application in Brain module periodically requests data from movement sensor and if there is difference between previous received data and current received data, it is shows distance between previous and current point as shown in figs. 10-11.

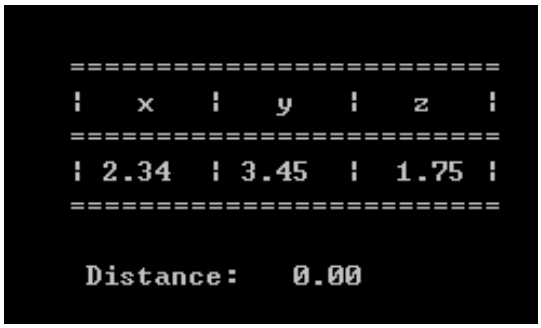

Figure 10: Distance tracking (no movement)

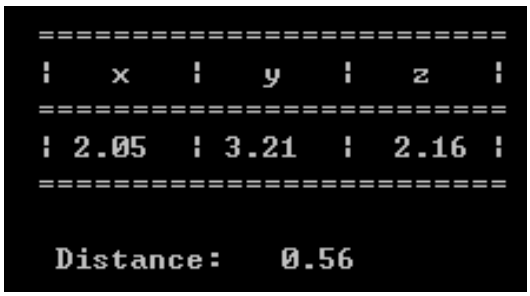

Figure 11: Distance tracking (movement detected)

As we have shown in this section, the proposed SL works well under distributed environment and serves as light and portable while keeps functionality.

5. CONCLUSIONS

We propose the streaming layer for personal robot's middleware, which is simple enough to be easily adopted for any platform yet fully functional and support all necessary tasks required for effective inter applications communication. The proposed streaming layer is consist of tree main functions and includes two data structures. With possibility of registration user defined interfaces it is suitable for distributed environments, i.e. module based architecture including small-embedded systems like as DSP board. To verify the proposed streaming layer structure it is implemented using C and tested.

ACKNOWLEDGMENT

This research was partially supported by the Brain Korea 21 Program through Kangwon National University

REFERENCES

[1] T.Fukuta, R. Michlini, V. Potkonjak, S. Tzafestas, K. Valavanis, and M. Vukorbrativic, "How far away is "Artificial Man?"", *IEEE Robotics Automation Magazine.* pp. 66-73, Mar 2001

[2] Rondey A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation* RA-2(1):14-23, 1996

[3] Makelainen, T, Kaikkonen, J, Hakala, H," Interfacing functional modules within mobile robots," *Intelligent Robots and Systems . Intelligence for Mechanical Systems Proceedings IR S . IEEE RSJ International orkshop* on, 3-5 Nov 1991

[4] Fryer, J.A, McKee, G.T, Schenker, P.S, "Configuring robots from modules: and object oriented approach ", Advanced Robotics, 1997. *ICAR . Proceeding. th International Conference* on, 7-9 Jul 1997

[5] Ishiguro, H., Kanda, T., Kimoto, K, Ishida, T," A robot architecture based on situated modules" *Intelligent Robots and Systems . IR S . Proceedings. IEEE RSJ International Conference* on. Volume:3,199

[6] Chatila, R, Ferraz de Camargo, R, "Open architecture design and inter-task/inter module communication for an autonomous mobile robot," *Intelligent Robots and Systems . Towards a New Frontier of Applications Proceedings. IR S IEEE International orkshop* on, 3-6 Jul 1990

[7] Hans Utz, Stefan Sablatnog, Stefan Enderle, and Gerhard Kraetzschmar," Miro-Middleware for Mobile Robot Applications", *Transactions on Robotics and automation*, vol. 18, No.4, August 2002.

[8] IEEE standard for a High Performance Serial Bus" IEEE std 1394-1995, IEEE1394 std 1394a-2000"

[9] Universal Serial Bus Specification revision 1.1: September 23. 1998

[10] CAN specification Part A and Part B

[11] Bluetooth SIG groups, Specification of the Bluetooth System, Ver1.1 Draft Oct 2000.

[12] P.A.Bernstein, "Middleware: A Model for Distributed System Services," *Communications of the ACM*, vol. 39, pp.86-98, 1996

[13] RMI specification. http://java.sun.com/products/jdk/rmi /index.html

[14] The Common Objcet Request Broker: Architecture and Specification revision 2.3: Jun 1999

[15] COM and DCOM specification. http://www.microsoft. com/com/resources/specs.asp

[16] William Stallings, "High-Speed Networks and Internets Performance and Quality of Service" 2nd Edition. 2000

[17] Gun Yoon, Hyoung Yuk Kim, Ju Sung Lee, Hong Seok Kim, Hong Seong Park, Middleware Structure for Personal Robot, *The th International Conference on control and Automation* (ICCA '03), pp.153-157, June, 2003