Architecture of Streaming Layer as Core of Personal Robot's Middleware.

Vitaly Li*, Seongho Choo**, Kiduk Jung***, Donghee Choi****, Hongseong Park*****

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

Abstract: This paper, proposes concept of personal robot middleware core also called streaming layer. Based on openness and portability, the streaming layer is proposed in order to meet requirements of different kinds of applications. The streaming layer architecture provides effective management of data flows and allows integration of different systems with ease regardless software of hardware platform. With extensibility support additional features can be build in without affect to performance. Therefore, heterogeneous network support, real-time communications, embedded boards support can be easily achieved. In order to achieve high performance together with portability into different platforms, the most functions has to be implemented in C language, while critical parts, such as scheduling, priority assignment has to be made using native functions of tested platforms.

Keywords: personal robot, middleware, streaming layer, heterogeneous network,

1. INTRODUCTION

Today rapid growth of human needs in personal assistant in order to handle every day tasks, such as electric home appliance, communications, privacy, and entertainment has been detected. The personal assistant systems also called personal robots are studied extensively [1-4]. As mentioned in literature: Personal robots should be feature human-like characteristics in their behavior, regarding motion, intelligence, and communication [5]. One of the engineering tasks in personal robots area is robot communications and integration into home network system. This paper considers open architecture modular based robot consists of modules representing fully or partially autonomous system [6]. Therefore such robot is considered as special class of distributed systems and problem of appropriate choice of middleware arises.

A number of solutions have been proposed [7-19]. There are four group typically considered. First group is transactional middleware. Transactional middleware [7, 8] uses two-phase commit protocol [9] to implement distributed transactions. It can simplify the construction of distributed systems; however it has restricted portability issues and creates undue overhead if there is no need to use transactions. Second group is message-oriented middleware [10, 11]. It supports the communication between distributed system components by facilitating message exchange. It is well-suited for implementing distributed event notification. However there is limited support for heterogeneity and scalability. Third group is procedural middleware [12]. It supports the distributed environment by implementation of remote procedure call. However, there are serious problems in reflexivity and scalability. Fourth group is object and component middleware [13-19]. The idea here is to make object-oriented principles, such as object identification through references and inheritance, available for the development of distributed systems. Object-oriented middleware provides powerful component model. However, the scalability and heterogeneity of existing solutions is still restricted.

Choice of middleware has to be made under fact that robot environment presents some restrictions and requirement. First, middleware has to be portable over different kinds of platforms existing today and in future. Hence it has to be simple and light, providing only general functionality. For example, embedded system lacks some of the functions provided by workstation. However, both of them have to provide some set of basic services, while allowing applications communicate with each other. From other hand middleware has to allow engineer to use power of particular platform. Hence it has to be scalable meaning that functionality could be extended easily without loosing compatibility. Second, middleware has to allow using different network interfaces, such as Ethernet, IEEE1394 [20], RS232, USB [21], CAN [22] and so on. Hence, in opposite to classic middleware, it has to include network controlling functionality. Third, robot's middleware has to be more reliable comparing to general distributed systems.

Since existing solutions cannot satisfy above requirements, we decided to develop new robot specific middleware. The proposed solution based on two layer scheme. First layer is Network Adaptation Layer (NAL). It deals with network interfaces, providing naming, addressing and routing services. It provides transparent data transfer and allows above layers act without doubt of network heterogeneity. Second layer is Streaming Layer (SRL). It deals with application management, marshalling and coordination. For the sake of reliability it also deals with binding service and provides migration service. Several interfaces per application are supported, so extension can be made easily.

In order to achieve high performance together with portability into different platforms, the most functions has to be implemented in C language, while critical parts, such as scheduling, priority assignment has to be made using native functions of tested platforms.

2. MIDDLEWARE STRUCTURE

In this section we define middleware structure as shown in figure 1.

This middleware is separated into network control level (NAL) application control level (SRL). Communications between SRL and NAL goes through interface provided by NAL. This guarantees that details of implementation of hardware-dependent part are hidden from the SRL and makes porting of middleware or adding new network interface component easy without affect to performance of SRL.

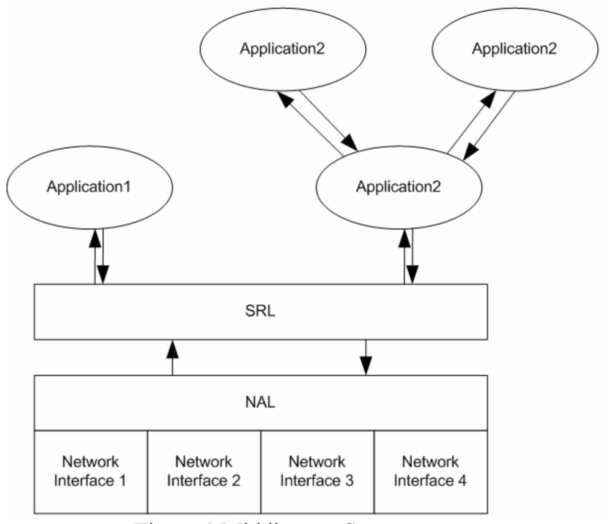

Figure 1 Middleware Structure

Main functions of NAL are naming, addressing and routing. By naming we mean the global name of hardware platform (node) over distributed system. It has to be unique and provide location of remote module clear. The addressing service from other hand provide name of the node over particular network interface meaning that for set of nodes connected by such network interface, location is unambiguous. The example of such addressing is IP. For the set of connected nodes, usage of unique IP address allows to locate certain node precisely. Hence, the one node has name over distributed system and address for every network interface it use. Finally routing is a process of location of certain remote node by its name. Routing involves two functions: transmission data over the path to the destination by source node and forwarding data to destination for intermediate node. While first function is clear, forwarding data is more complicated operation due to heterogeneity of the network. Forwarder has to receive data from one network interface, decide appropriate network interface for forwarding and send data to that interface. Since different network interfaces have different parameters such as maximum packet size, bandwidth and so on, it is non trivial task; however it is out of the scope of paper.

Functionality of SRL is defined by application needs. It provides application management, marshalling, coordination, binding service and migration service. It is described in details in next section.

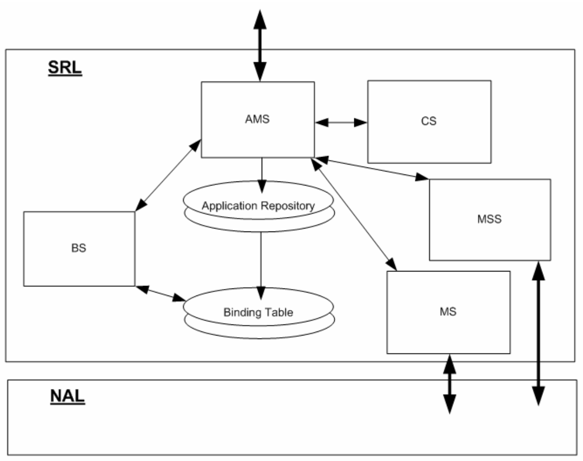

Figure 2 Streaming Layer Architecture

3. STREAMING LAYER ARCHITECTURE

The overall architecture is shown in figure 2. Detail explanation of each part is given below.

3.1 Application management

We define application interface as independent set of variables and functions. Application management (AMS) provides application interface registration/deregistration, location, data transmission to/from application interface and procedure invocation by application interface. Every interface defines set of specific descriptors based on functionality it provides. Hence, upon registration of the application interface, unique ID over same set of descriptors is generated. For example, if there are two vision interfaces with exact same functionality they will get different IDs, while motor control interface might have ID coincident with one of the vision interface. Thus couple of ID and interface descriptors uniquely defines application interface, making location service easy. AMS operates with one information structure – application repository that keeps information for identification and location of application interfaces. The data about application interface is added on registration and removed upon deregistration. The data transmission to/from application interface and procedure invocation are handled by platform specific API.

3.2 Marshalling service

Marshalling service (MSS) provides packetization of data transmitted to remote application. It defines types of the packets and headers structure.

3.3 Coordination service

Coordination service (CS) provides admission management. Whenever source application interface wish to establish connection to remote application interface, CS verifies existence of the remote application, agreement in parameters, matching functionality of the remote application interface with requested from source application interface and access rights of local application interface.

3.4 Binding service

Specific character of the most robot application is periodicity. Therefore long life connections with periodic exchange of the data, procedure invocation, request/response exists. In order to reduce unnecessary overhead, binding service (BS) exists. Whenever bind request arrives binding service inserts bind record to the binding table in both side of connection link. Such record consists of local application interface and remote application interface currently bind. Such connections have highest priority and serves without involving coordination service each time transmission occurs. SRL support two types of binding: binding by couple and binding by descriptors set. Binding by couple suppose one-to-one communication link. Binding by descriptors set allow one-to-many link.

3.5 Migration service

For the sake of the agents system and robots vitality (feature of the robot to remain functional after removing some part of it) migration service (MS) is introduced. The main functionality is to allow application interface migrate to another node keeping established connections. On receiving migration request MS send notification messages to the network suspending all transmission to migrant. Then application interfaces is moved to another node with notification of target node's MS, which updates all information structures (application repository, binding table) and sends update notification to the network. Every node's MS on receiving such notification updates information structures and resume transmission to migrant if any.

4. NOTES

In this section we discuss the scalability, real-time and portability issues.

4.1 Scalability

As shown by figure 1 one application may have several application interfaces. So, by registering application that manages other applications in some manner, functionality of proposed middleware can be extended. We call such application extension. The number of extension levels restricted only by operating system. From other hand, special cases (such as embedded sensor board that has static connection without needs in routing, forwarding binding and restricted multitasking) can be handled by reducing functionality of SRL. This can be achieved by several techniques, such as embedding application functionality into SRL and disabling unnecessary services.

4.2 Real-time

We introduce simple priority assignment by using binding table and handling periodic tasks with highest priority. Also, advanced scheduling can be implemented as extension. Since several routes from source to destination exist, high priority tasks are always routed with respect to minimum delay, while low priority tasks routed with respect to available bandwidth.

4.3 Portability

As mentioned above, proposed middleware has simple structure and independent layers. Therefore the porting of middleware can be made easily. Note that the NAL mostly has to be ported with respect to hardware layer, while the SRL with respect to operating system features.

5. CONCLUSION

We propose the architecture of middleware for modular based personal robot. In particular, architecture and functional description of Streaming Layer is proposed. The design of Streaming Layer is made based on various need of personal robot platform. The portability and scalability is in focus as well as heterogeneity of the network. Possibility to use extensions, assign priority to application interfaces, migration, and on-line coordination makes proposed middleware powerful solution for needs of personal robot's applications. High portability potential makes proposed middleware also suitable for use on various platforms including embedded and real-time systems. Therefore, proposed middleware architecture is suitable for distributed environments such as modular based personal robot.

ACKNOWLEDGMENTS

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

REFERENCES

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

[2] Makelainen, T, Kaikkonen, J, Hakala, H," Interfacing functional modules within mobile robots," *Intelligent Robots and Systems '91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on, 3-5 Nov 1991*

[3] Fryer, J.A, McKee, G.T, Schenker, P.S, "Configuring robots from modules: and object oriented approach ", *Advanced Robotics, 1997. ICAR'97. Proceeding. 8th International Conference on, 7-9 Jul 1997*

[4] Ishiguro, H., Kanda, T., Kimoto, K, Ishida, T," A robot architecture based on situated modules" *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on. Volume:3,1999*

[5] 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*

[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'90. 'Towards a New Frontier of Applications', Proceedings. IROS'90, IEEE International Workshop on, 3-6 Jul 1990*

[7] Hudders E.S., "CICS: A Guide to Internal Structure", *Wiley, 1994*

[8] Hall C.L., "Building Client/Server Application Using TUXEDO", *Wiley, 1996*

[9] Bernstein P.A., Hadzilacos V., Goodman N., "Concurrency Control and Recovery in Database Systems", *Addison Wesley, 1987*

[10] Gilman L., Schreiber R., "Distributed Computing with IBM MQSeries", *Wiley, 1996*

[11] Hapner M., Burridge R., Sharma R., "Java Message Service Specification. Technical Report", *Sun Microsystems, http://java.sun.com/products/jms, Nov 1999*

[12] Open Group, editor, "DCE 1.1: Remote Procedure Calls", *The Open Group, 1997*

[13] Object Management Group, "The Common Object Request Broker: Architecture and Specification Revision 2.2", *Feb. 1998*

[14] Orfali R., Harkey D., Edwards J., "Instant CORBA", *Wiley, 1997*

[15] Box D., "Essential COM", *Addison Wesley Longman, 1998*

[16] JavaSoft, "Java Remote Method Invocation Specification", *revision 1.50, jdk 1.2 edition, Oct.1998*

[17] Moonson-Haefel R., "Enterprise Javabeans", *O'Reilly UK, 1999*

[18] Chung P., Huang Y., Liang D., Shin J., Wand C.-Y., Wand Y.-M., "DCOM and CORBA: Side by Side, Step by Step and Layer by Layer", *C++ report, pp18-29, Jan. 1998*

[19] Emmerich W., "Engineering Distributed Objects", *John Wiley & Sons, Apr. 2000.*

[20] IEEE standard for a High Performance Bus, IEEE std 1394-1995, IEEE std 1394a-2000

[21] Universal Serial Bus Specification revision 1.1, Sep. 23, 1998

[22] CAN Specification Part A and Part B