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.
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.

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
ACKNOWLEDGMENTS

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

REFERENCES

[22] CAN Specification Part A and Part B