Design for Supporting Interoperation between Heterogeneous Networks in Personal Robot System

Seongho Choo, Vitaly Li, Ik-gyu Jang, Tae-kyu Park, Ki-duk Jung, Dong-hee Choi, Hongseong Park

Division of Electrical and Computer Engineering, Kangwon National University, Chunchon, Republic of Korea
(Tel : +82-32-251-6501/6346; E-mail: {somebody, vitally, ikgu, extasytk, toumai, blesscdh, hspark}@control.kangwon.ac.kr)

Abstract: Personal Robot System in developing, have a module architecture, each module are connected through heterogeneors network systems like Ethernet, WLAN (802.11), IEEE1394 (Firewire), Bluetooth, USB, CAN, or RS-232C. In developing personal robot system we think that the key of robot performance is interoperability among modules. Each network protocol are well connected in the view of network system for the interoperability. So we make a bridging architecture that can routing, converting, transporting data packets with matcing each network's properties. Furthermore we suggest a advanced design scheme for realtime / non-realtime and control signal (short, requiring hard-realtime) / multimedia data (large, requiring soft-realtime). By some application systems, we could test performance, interoperability and stability. In this paper, we show our design concept, middleware architecture, and some applications systems using this middleware.

Keywords: Middleware, Personal Robot, USB, Bluetooth, IEEE 1394, 802.11, CDMA, Ethernet, CAN, RS-232C

1. INTRODUCTION

On developing personal robot system in Korea, is the module-based system. Every internal function in personal robot is separated to each function module, for example, vision, void, brain, arm, main control, sensor, motor driving, and home network interfacing modules. Each module is connected with heterogeneous networks like Fig. 1., using Ethernet, WLAN (Wireless Local Area Network, 802.11), USB (Universal Serial Bus), Bluetooth, IEEE 1394, CDMA (Code Division Multiple Access), CAN (Control Area Network), RS-232C, respectively. [1, 6-8]

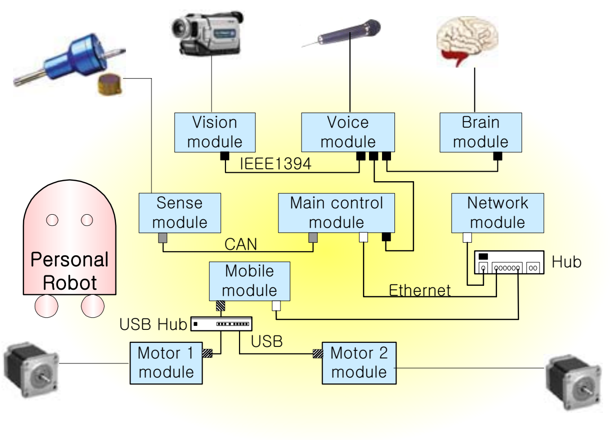

Fig. 1 A Scheme of the Module-Based Personal Robot System

The module-based architecture supports not only more function specialization but also easy integration of function modules. Because each module can be made by different developer who have expert skill about specific function but don't need a lot of detailed knowledge about personal robot. And they can use their own network continuously that has more experience and is industrial standard. This development platform gives us more independent and function specific design environment. Thus we expect to convenience co-working and rapid implementation of whole robot system.

We think that the key of module-based system is interoperability through networks. For this, we have being making a middleware for intercommunications among each module through heterogeneous network system

There are so many architectures named 'middleware', for examples, Jini, OSGi, HAVi, UPnP (Universal Plug and Play), CORBA, DCOM, and etc. [2-4, 10, 11], but they have some difference and constraints on running platform, supporting protocol, requiring amount of system resource. So our design concept is simple, light, portability to variety system platform, to give a unified network access scheme to developer, and stability to malfunction of each hardware module, and to support all operation system, all protocols used generally as possible.

In this paper we present the middleware for module-based personal robot system. The next section, we show general design concepts on environment of hardware module in personal robot. In section 3, it is discussed about our middleware structure more detailed. In section 4, we verify our middleware from some implemented network applications that have issues at the view point of network system. Finally, conclusions and future work are given in section 5.

2. NETWORK MIDDLEWARE

The middleware must be able to be used every hardware platform in personal robot and to support unified network independent interfaces with interoperating each network protocols.

On the view point of hardware platform, we made four versions of middleware, Microsoft Windows, Microsoft Windows CE for Hand-held PDA, Linux, and non-OS version for DSP board have a Texas Instrument chip. To reduce using system resource and network overhead, we selected simplicity, performance and OS dependent software architecture at network device level, but we got a little loss of porting effort because of using layered system design, component based software architecture and keeping source-level portability as possible. In Fig. 2, it is shown the abstraction of hardware module and simplified middleware layers.

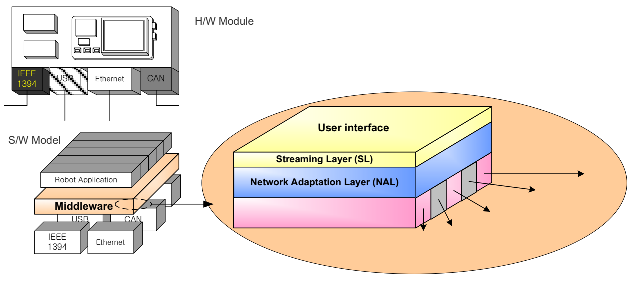

Fig. 2 The Overview of Middleware Structure

The main goal of this middleware is to support intercommunication easily through variety network protocols. Now we have implemented diverse protocols in wired environment, Ethernet (commonly using), USB (variety application used easily), CAN (used at field level devices like PLC, sensors, and actuators), RS-232C (for direct attached simple devices), also in wireless environment, IEEE 1394 (mainly used at home multimedia appliances), Bluetooth (form wireless multimedia communications), CDMA (cellular or mobile phone, to send some emergency information from robot to a robot owner at outdoor).

The middleware's main functions are packet routing, protocol conversion, maintenance a physical disconnecting/connecting of module and well transportation data packets have variety properties like realtime/non-realtime, short/long size, and possible/impossible dropping when the worst case.

Generally in hardware module, a main application program (we called 'robot application program') is at board memory, like hard-disk, EPROM, or Flash ROM, and this middleware is together formed a system library. This library accesses network hardware interface using system calls.

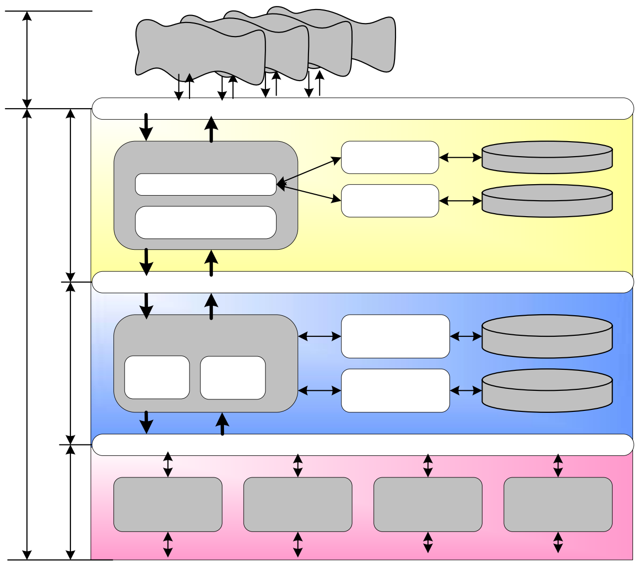

Fig. 3 The Middleware Structure Diagram

3. MIDDLEWARE ARCHITECTURE

The network middleware for personal robot system consists three layers, SL (Streaming Layer), NAL (Network Adaptation Layer), DIL (Device Independent Layer), like previous shown. More detailed structure is shown Fig. 3.

DIL has system specific software modules for each network device. NAL has a manager to do naming and routing service with each own tables. SL perform the marshaling/unmarshaling data and management many abstract objects existing in robot system. Also, all layer and inner software modules are implemented software components. Thus all interface are unified and linked dynamically.

3.1 Device Independent Layer (DIL)

DIL consists some network access library in a module. As system environment, this has a device driver or some hardware direct control codes. Briefly, it is a composition of network transmitting software modules of each protocol like socket library (TCP/IP), network device specific routines are provided from hardware venders.

The maximum packet size of each protocol is all different. For protocol conversion, fragmentation and reassembling are required, also appropriative filling every fields of protocol, too. DIL is monitoring the connection status according to be offered from each protocol. If a event is happened, DIL sends a notification message to the upper layer, NAL. So DIL.

DIL sends and receives data to the upper layer, NAL, through some queues. And each sending and receiving procedure operating on threads (Windows) or independent process (Linux).

This charges real data transferring through media, fragmenting/reassembling packet, and notification of changing network status if possible.

3.2 Network Adaptation Layer (NAL)

NAL charges protocol conversion and routing. Every internal modules in robot have a unique ID and a unique name, these two are identically 1:1 matched. All modules are distinguished with ID.

When a module is attached on network media, its middleware broadcast his ID to neighbor modules through all NIC of its. If a module receives a initialization broadcast packet, it propagates this message to its another neighbors after updating its local routing table. At the time that propagations are reached to end modules, all modules reconstruct own routing table. When detached a module, if a connected network can support this notification, a routing table update message is broadcasted by neighbor modules. For a network protocol can't support the notification, every module checks the connection of neighbors continuously. This routing method of NAL is similar to OSPI in Link State Protocol.

NAL is shown a virtual network to upper layer because it has the open architecture.

3.3 Streaming Layer (SL)

SL is offered network independent interfaces for managing object at local or remote module to robot applications. All resource of robot system are defined objects. SL charges application synchronization, management of transaction, and data representation.

For the object communication, we got a scheme from CORBA of OMG (Object Management Group) and redesigned more simple and light. An remote object is represented a stub in local module. The real execution code is a remote object, called implementation. The broker manages synchronization and data transferring between. Of course, a robot application can access broker interface directly. And SL manages more high level abstraction of data representation like marshalling and unmarshalling. A data from robot application is changed a appropriate format to be able to send through networks (marshalling). In the other side, network formatted data is changed to be understood to the peer robot application (unmarshalling).

Now, extended interfaces are in developing for VM (Virtual Machine) that offers the hardware independent execution environment.

4. MIDDLEWARE APPLICATIONS

When a new protocol or a redesigned internal scheme is involved to middleware, we verified through making a demo application for verifying and reviewing from others. In this section, we represent these applications and discuss its properties.

4.1 Connecting IEEE 1394 modules

For verifying this middleware structure, we did a simple middleware, can support only one protocol, IEEE 1394. The middleware was loaded three Linux machines, rolled core robot modules like brain, base, and arm. (is shown in Fig. 4)

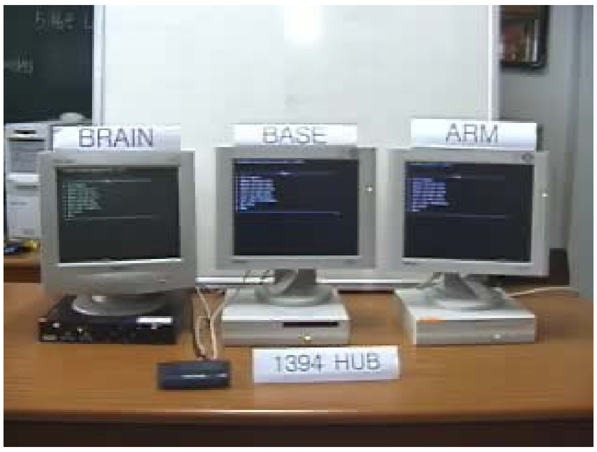

Fig. 4 Core modules through IEEE 1394

They are connected with IEEE 1394 hub. In this application we are able to check the stability and routing scheme. The whole middleware stack was well operated. When removed and added connector from/to hub, each machine reconstructed its routing table.

Each machine could send and receive variable and call remote procedures using unique module names.

4.2 Interoperation with Personal Robot and Home Network

In this demo application, we showed more realistic behaviors of the personal robot. With some simulated home appliances and devices, personal robot performs actions like turning on/off a digital TV, every jobs against emergency event using home network communication and internet if needed. In case to extend to home network and internet, coverage area enlarges significant to apply the personal robot system.

The robot owner can remote control at outdoor through home gateway and internet, so he or she can command jobs or behaviors to personal robot in home. The remote command sends from home gateway to home network, 802.11 WLAN AP (Access Point), 802.11 home network module in personal robot, sequentially. (is shown in Fig. 5)

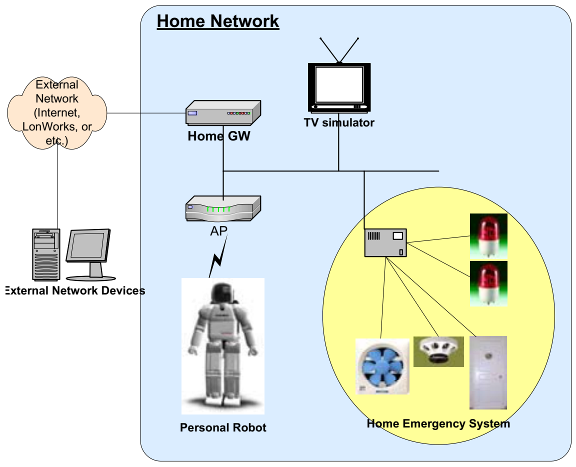

Fig. 5 Home Network and Personal Robot

Because a personal robot can control all digital home appliances and home automation devices, if a gas leakage event is happened. It will automatically emergency actions like closing gas valve, turn on air-pan, opening door, turn on warning lamp, and sending a short emergency message to host or hostess' cellular phone via CDMA SMS service. In this demo scenario, we simulated all home automation devices with animation in personal computer.

4.3 Heterogeneous Networks System

For verifying the roll of network gateway and bridge, we connected different operation system machines with a middleware through various network protocols (is shown at Fig. 6).

A Linux based PC with a Ethernet interface, a industrial PC with Ethernet and IEEE 1394, Windows based desktop PC

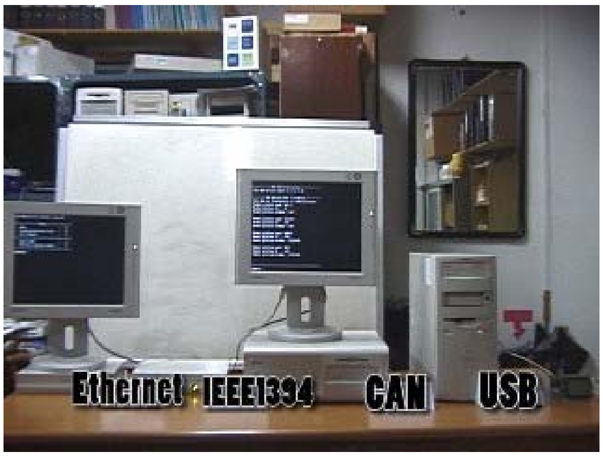

Fig. 6 Heterogeneous Networks Demo System

with IEEE 1394 and CAN interface card, Linux based PC with CAN card and USB, and a industrial controller with USB interface and a small LCD panel. All are connected linearly.

We can verify the communication through many heterogeneous network systems. Some input from the most left side machine make a respected output to the more right side industrial controller, also response with low network delay. This means our middleware is managing network route and is converting protocol appropriately.

4.4 Multimedia Service using PDA

In this demo system, we want to check the network performance on wireless protocol of middleware. Because 802.11 WLAN has a comparative low bandwidth against other wired network protocol.

We made a Windows CE Pocket PC 3.0 version middleware for embedding to Compaq Hand-held PDA. This PDA has 802.11 PCMCIA card to communicate with Samsung WLAN AP in laboratory, and this AP is connected to university intranet through 100 M Ethernet. A webcam generates a video stream and is sending this to a desktop PC through USB. A Microsoft Windows version middleware in PC sends to WLAN AP through Ethernet. A video encoder on webcam server for PC and a media player for PDA were created using Microsoft Media Player Software Development Kit.

In Fig. 6, it is shown the operation this system. A webcam (bottom of left in the left side picture) is getting images of a developer held a PDA, continuously. On PDA (the right side picture), this video stream are being played.

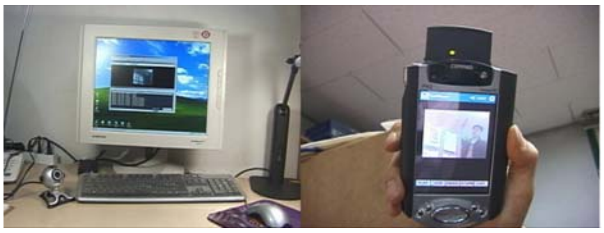

Fig. 7 Multimedia Service using a WebCam

This system was operated well but has some delay sourced WLAN bandwidth. So this application can be used a security system or a monitoring system through eyes or a extra camera on personal robot.

4.5 CDMA SMS Service

To guarantee a communication method to host or hostess of robot at outdoor, we developed a simple software module to send a short message to his or her cellular phone using SMS service through CDMA.

When a personal robot recognizes a emergency event, it will construct a message and send to home gateway using home network interface module through 802.11 WLAN. Home Gateway sends this to the SMS service server in a mobile phone service company. Then phone can get the message from company's CDMA backbone network. The Fig. 8 shown that a man puts a message manually (the left side picture) and received message in his CDMA phone (the right side picture).

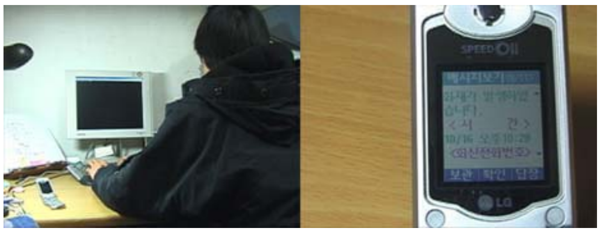

Fig. 8 CDMA SMS Service Demo

This software module is very useful and efficient to interfacing between human and robot, especially home emergency status.

5. CONCLUDION

In this paper, we discuss about the module-based personal robot system, open architecture middleware, internal operation of our middleware, and verifying the functionality by some implemented demo application system.

This middleware have been developing during three years, and integrating to some real robot platform made by universities, companies, and research facilities. At present, only one developer can't make all. Thus we must sharing jobs. The module-based personal robot structure offers a cooperative development processes. Every expert's great results can be composed through network using our middleware.

We are supporting various hardware platforms, operating systems, and network protocols, going to upgrade internal software architecture, to improve system stability and performance. This middleware offers a unified network interface to developer, so they can learn and use easily, even if they don't know about network protocols.

Now we have the main research point to the object-oriented communication in the Streaming Layer, and porting to real-time operating system using real-time system calls and other system resources.

ACKNOWLEDGMENTS

This work was supported by the Brain Korea 21 Project of Kangwon National University in 2004.

Specially thanks to Hyoung-Yuk Kim, Gun Yun, Ju-Sung Lee, and Jung-Bae Lee who contributed to this middleware at past.

REFERENCES

[1] IEEE standard for a High Performance Serial Bus "IEEE std 1394-1995, IEEE1394 std 1394a-2000"
[2] http://www.jini.org, "Jini Architecture Specification:, 2000"
[3] http://www.osgi.org, "OSGi Specification Oveview Version 1.0", 2000
[4] http://www.havi.org, "HAVI Specification Version 1.1", 2001
[5] http://www.upnp.org/resources/whitepapers.asp, "Understanding Universal Plug and Play: A White Paper", 2000
[6] Universal Serial Bus Specification revision 1.1: September 23. 1998
[7] CAN specification Part A and Part B
[8] Bluetooth SIG groups, Specification of the Bluetooth System, Ver. 1.1 Draft Oct 2000.
[9] RMI specification. http://java.sun.com/products/jdk/rmi/index.html
[10] The Common Objcet Request Broker: Architeture and Specification revision 2.3: Jun 1999
[11] COM and DCOM specifications, http://www.microsoft.com/com/resources/specs.asp