Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Distributed Control System of Humanoid Robots based on Real-time Ethernet

Fumio Kanehiro*, Yoichi Ishiwata†, Hajime Saito‡, Kazuhiko Akachi§,
Gou Miyamori§, Takakatsu Isozumi§, Kenji Kaneko* and Hirohisa Hirukawa*
*National Institute of Advanced Industrial Science and Technology(AIST),
Tsukuba Central 2, 1-1-1 Umezono, Tukuba, Ibaraki, 305-8568 Japan
CREST, Japan Science and Technology Agency,
4-1-8 Honcho Kawaguchi, Saitama, Japan
Email: {f-kanehiro, k.kaneko, hiro.hirukawa}@aist.go.jp
†MovingEye Inc.,
Hayakawa No.3 Building 4F, 2-2 Kandatamachi, Chiyoda-ku, Tokyo, 101-0046 Japan
Email: you1@movingeye.co.jp
‡General Robotix, Inc.,
Tsukuba Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki, 305-8568 Japan
Email: h.saito@generalrobotix.com
§Kawada Industries, Inc.,
122-1 Hagadai, Haga-machi, Haga-gun, Tochigi 321-3325 Japan
Email:{kazuhiko.akachi, go.miyamori, taka.isozumi}@kawada.co.jp

***Abstract*— In this paper we realize a real-time communication on Ethernet and develop an onbody distributed control system for a humanoid robot, HRP-3P. Real-time communication on Ethernet is realized by (1) a communication method using the data link layer directly and (2) timing control using a real-time operating system ARTLinux. This enables us to reduce the cost of embedded systems and improve developmental efficiency. A CORBA implementation which works on this communication layer is also developed to increase compatibility with existing software. Finally a small-size distributed robot controller is developed for the onbody network of HRP-3P and a distributed I/O system is developed on top of this.**

I. Introduction

System configurations of self-contained robots can be divided broadly into two categories, that is, a centralized system and a distributed one as shown in Fig.1. A centralized system places a computer and an interface board which has A/D, D/A etc. at the center. Motors and sensors are all connected to the computer. A distributed system distributes I/O nodes all over the body which are connected by onbody network. Motors and sensors are connected to the nearest I/O node.

From perspective of software, the centralized system can be controlled by simple software, because only one CPU controls all I/Os. On the other hand, software for the distributed system must be more complicated. Software on I/O nodes must communicate with a central computer while doing I/O. Also a software on the central computer needs to gather information from all nodes and distribute motion commands to them while doing an whole body motion control.

From the perspective of hardware, wires between motors/sensors and the interface board become long and gather to the board on the centralized system. Therefore they are susceptible to noise and wiring near the board becomes difficult. In the case of adding a motor or a sensor, a new wire must be added from the center. Wires between motors/sensors and I/O nodes are short on the distributed system and it is easy to add a motor or a sensor so long as the capacity of the network is sufficient. The number of wires between nodes is also constant(power and network).

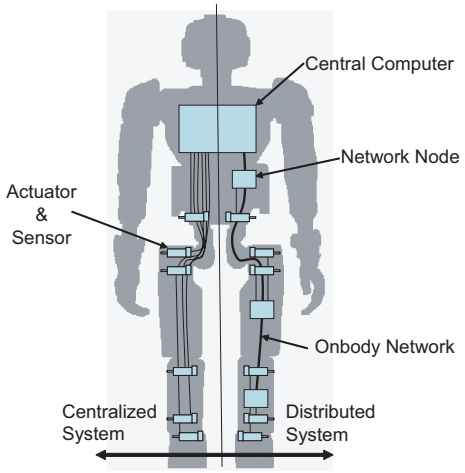


Fig. 1. Configurations of self-contained robots

This distributed system configuration is effective especially on humanoid robots. Many humanoid robots have about 30 DOF. It is expected that the number of DOF increases by introducing multi fingered robot hands. The number of sensors can also increase dramatically by introducing sensor skin. Humanoid robots have wide movable ranges of joints and for

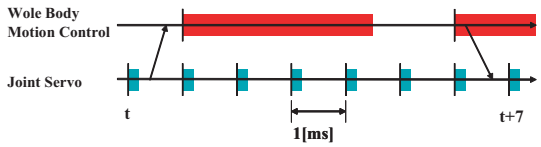


Fig. 2. Time chart of HRP-3P system

that reason it is hard to provide spaces for wires. These issues are especially serious on a small-size humanoid robot, thus some of them adopted a distributed system from the start.

We realize real-time communication on Ethernet and develop a distributed I/O system using it as an onbody network for a humanoid robot HRP-3P.

This paper is organized as follows. Section 2 explains why Ethernet is selected as an onbody network. Section 3 describes how to realize real-time communication on Ethernet. Section 4 presents a small-size controller to distribute all over the body. Section 5 explains a real-time CORBA which works on real-time Ethernet. Section 6 measures the performance of a real-time communication library. Section 7 describes a distributed I/O system for HRP-3P using the real-time CORBA. Section 8 concludes the paper.

II. Onbody Network

Requirements on an onbody network of robot changes according to specifications of a robot system. Therefore, we decided specifications of HRP-3P system at first as follows based on a humanoid robot HRP-2[1] which has a centralized system configuration.

1) The system consists of a central computer and I/O nodes. In order to decrease the number of wires, bus-connection or daisy chain is used.
2) An whole body motion control task is executed on the central computer in 5[ms] period.
3) A joint control task is executed on I/O nodes in 1[ms] period.
4) The computer and nodes communicate every 5[ms] and those communications are completed within 1[ms]. Figure 2 shows a time chart of HRP-3P system.
5) Tasks on the computer and nodes runs in sync to get sensory information at the same time and rotate motors at the same time.

Assuming the computer and all I/O nodes are connected to a single network, a requested bandwidth is estimated as follows. **Central Computer → I/O node** Motor commands and PD gains(36 DOF) are described with 16[bit] data and 16×(36+36+36)/0.001=1,728,000[bps] is required.
I/O node → Central Computer Joint angles, joint torques, motor temperatures(36 DOF), gyrometers(×3), 6 axis force sensors(×4), accelerometers(×3) are described with 16[bit] and 16×(36+36+36+6×4+3+3)/0.001=2,208,000[bps] is required.

Moreover, practically a protocol overhead is additionally required. Considering the possibility of adding motors and sensors and shortening of the control period, about 10[Mbps] will be required.

There are several networks in industry, ARCNET(Attached Resource Computer NETwork), I2C(Inter Integrated Circuit), CAN(Controller Area Network) and so on. ARCNET incorporates a token-passing protocol where media access is determined by the station with the token. When a station receives the token, it can either initiate a transmission to another station or it must pass the token to its logical neighbor. This scheme avoids collision. It also includes network configuration and error handling and its speed is up to 10[Mbps]. I2C is frequently used for communication between ICs as its name indicates. It consists of only two lines, clock and data and its speed is up to 3.4[Mbps]. I2C interface is built into many microprocessors, so it is easy to downsize network nodes. Therefore, network nodes which communicate through I2C is adopted by small-size humanoid robots that have strong constraint on size[2], [3], [4]. Speed of CAN is up to 1[Mbps] and it is comparatively slow. But it has high noise immunity and it is suitable for robots where power lines generate noise [5].

AIBO and QRIO[6] are based on an architecture called OpenR[7]. Modules are connected by an original bus called OpenR bus which consists of 10 lines including power line for motors, power line for computer and network line. Its speed is up to 12[Mbps]. A motor, a gear box and a motor control/communication board are integrated into a module called ISA(Intelligent Servo Actuator), and it works only by connecting to OpenR bus. ETL-Humanoid[8] also has an original token-passing type network and it realizes high frequency communication at 10[kHz]. An original bus can be optimized for applications, but it requires development of new interface hardware, device drivers and so on.

In addition, there are robots which adopt USB or IEEE1394 that are developed for personal computers originally.

HOAP series[9] have been adopted USB as an onbody network since its prototype [10] that is developed in HRP[11]. Its version is 1.0, so its speed is up to 12[Mbps].

Ethernet is a global standard network that is used by PCs etc. Its speed is fast enough, up to 1[Gbps]. However, it can't communicate in real-time, and meet deadlines. If real-time communication on Ethernet is possible, (1)cost of embedded systems can be reduced by using low-cost parts and (2)existing applications that use TCP/IP can be used therefore improving maintenance and developmental efficiency. Therefore, standardization of real-time communication on Ethernet have been discussed by the working group IEC/TC65/SC65C. Several specifications including PROFINET IO[12] and TCnet[13] were already proposed to this working group. Our system uses Ethernet as onbody network since real-time communication on Ethernet is realized with a small change on ARTLinux using a similar way to PROFINET IO and TCnet.

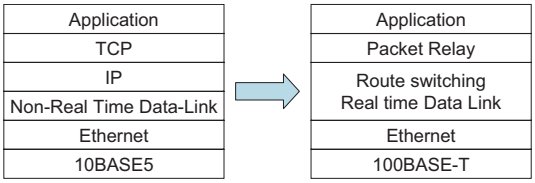


Fig. 3. Difference between conventional and real-time ethernet

III. Real-time Communication on Ethernet

A. Realization of Real-time Communication

There are three reasons real-time communication on Ethernet is considered to be difficult.

1) Ethernet incorporates CSMA/CD method. Completion time of communication can't be expected since collision is possible.
2) Processing time can't be expected since TCP/IP includes packet retransmission and buffering.
3) There is no way of controlling frame transmission time.

A frame collision in 1) occurs when using bus-connection with coaxial cable like 10BASE5 or star connection with repeater hubs. But it doesn't occurs when using point-to-point connection or star connection with twisted-pair cables or optical cables,

Unexpectable processing time in 2) can be avoided by not using TCP/IP as PROFINET IO and TCnet. That is, application data is transmitted directly in payload of Ethernet frame as shown in Fig.3. Using this method, time since an application calls data transmission system call of OS to a Ethernet device executes a transmission command becomes expectable. If transmission queue in a Ethernet device is empty, the frame is sent immediately after a transmission command is issued.

Frame transmission time in 3) need not to be controlled by network hardware, device driver and OS. It can be controlled by an application on real-time OS that decides when it calls a system call, since time between the system call and an actual transmission time is foreseeable. In this paper, ARTLinux[14] is used as a real-time OS.

Due to methods mentioned above, real-time communication on Ethernet is possible. But there are two points that must be paid attention. Since Ethernet is a protocol of physical layer and data link layer, a packet relay function of router is not available and we must implement our own relay function. If some error occurs during communication, the transmitted frame can't be received. Since an automatic retransmission in OS level causes deadline miss, an application must be developed considering that data may disappear.

B. Real-time Communication Library

A real-time communication library is developed based on methods mentioned in the previous section. This library can be used at the user-level.

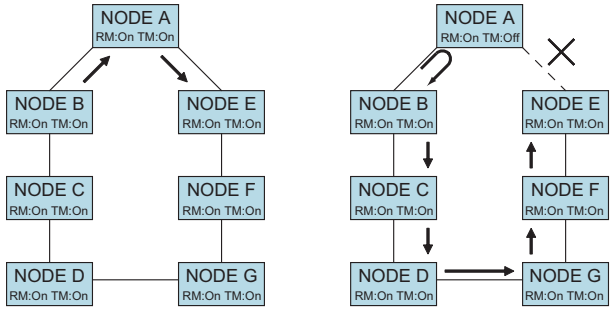


Fig. 4. Route reconfiguration(RM:Relay Mode, TM:Through Mode)

Two modes, that is, a relay mode and a pass-through mode of sockets can be set respectively.

In the relay mode, if a destination MAC address of a received frame is different from that of a received Ethernet port, the frame is re-transmitted. This function is required since functions in network layer of TCP/IP can't be used.

The through mode selects a destination port which is used by relay mode. When it is on, the other port is selected and when off, the same port is selected. Using this mode, two communication routes can be selected. Even if one of couple routes is disconnected, communication can be continued by selecting the other route. For instance, say node B and node E are communicating through a route shown in Fig.4 left. In this case the through mode of node A is on. If node A detects a disconnection of route between node A and node E, node B and node E can continue to communicate selecting a new route shown in Fig.4 right by turning the through mode of node A off. Network wires tend to suffer stress since they run through joints and it can cause disconnections. Introducing redundant routes and a route switching function, reliability of an onbody network can be improved.

C. Clock Synchronization

To realize specification 5), a function that synchronize all nodes running based on different clock is required.

In order to synchronize the clock on all nodes, a command that is based on IEEE 1588 Precision Time Protocol(PTP) is implemented. A node(called slave) is synchronized to the other node(called master) while communication. The synchronization error where a master and a slave are connected directly is smaller than where there are several nodes between them. So configuring neighboring nodes make pairs of master and slave as shown in Fig.5 right is preferred to where that one node works as a master of rest nodes as shown in Fig.5 left.

A timer interrupt is generated based on an external clock of a CPU. However its frequency is slightly different because of individual differences. If you synchronize a clock only once at the beginning, the difference is accumulated and causes a disturbance after a certain amount of execution period. So this clock synchronization function is executed at 1[s] period in the background.

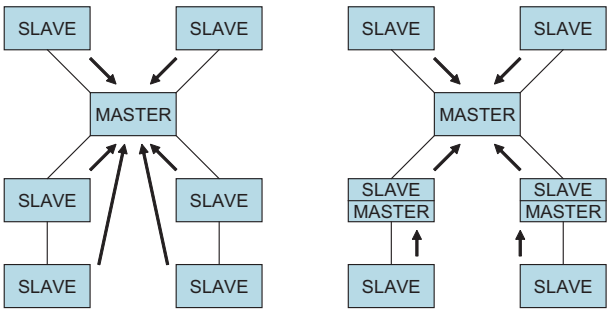


Fig. 5. Clock synchronization between nodes

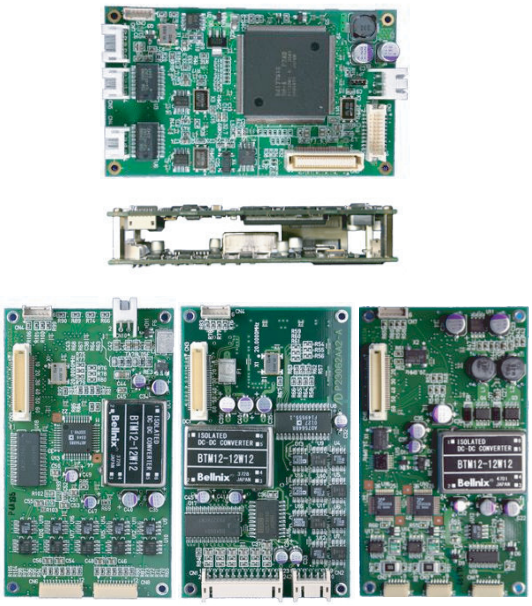


Fig. 6. CPU board, link I/O board, sensor I/O board and serial I/O board

IV. Distributed Robot Controller

In order to make a distributed I/O system using the real-time communication library, a small-size CPU board and I/O boards are developed. Figure 6 shows four kinds of developed boards. The upper row shows a CPU board which has SH4 and it is used by stacking with an I/O board as shown in the middle row. The bottom row shows a link I/O board which controls up to five joints, a sensor I/O board which measures sensors with analog outputs and a serial I/O board which has four RS422 ports from left. I/O boards are connected with a CPU board by a connector which is mounted at the upper left corner of boards and power is supplied through the connector. Brief specifications of these boards are shown in Table I.

The CPU board has two Ethernet ports to construct ring topology as mentioned above. A kernel and a root filesystem of ARTLinux are placed in flash memory and their modification can be done while running OS. Therefore, if a maintenance is required, a user can log on from a remote computer through network and modify them. The user need not to connect a serial cable and download a new files. This is possible by using Ethernet as network.

TABLE I

Specifications of CPU board and I/O boards

Common Specification	
Board size	55[mm]×95[mm]
CPU board	
CPU	32bit RISC-CPU SH4 240[MHz]
Memory	FLASH Memory 32[MB], SDRAM 32[MB]
I/O	10/100Base-T Ethernet × 2
	RS232C serial port× 1
Power	DC12[V]
Link I/O board	
AD	12[bit], ±10[V], × 10
DA	12[bit], ±10[V], × 5
Counter	24[bit], always-on with backup power supply, ×5
DIO	5[bit]
Sensor I/O board	
AD	16[bit], ±10[V], × 12
DA	12[bit], ±10[V], × 4
Serial I/O board	
AD	12[bit], ±10[V], × 4
RS422	asynchronous, up to 921.6[Kbps], ×4

Encoder counters of a link I/O board works with a power supply from a CPU board or a backup power line. So they continues to count encoder pulses even while the power supply to the CPU board is off. As long as the backup power is supplied, an origin of a joint needs not to be calibrated.

V. Performance Evaluation

In order to evaluate performance of the developed real-time communication library, an period error and a processing time of communication at a constant period is measured while changing the data size and the period. Outline of the program is shown in the following.

```
 1 periodic_send(unsigned long period, int len,
                 char *name, struct ether_addr *addr)
 2 {
 3   EtherRTSock *sock;
 4   sock = EtherRTSocket(SOCK_DGRAM, ETH_P_RT);
 5   EtherRTBind(sock, name);
 6   EtherRTConnect(sock, addr);
 7   art_enter(ART_PRIO_MAX, ART_TASK_PERIODIC,
             period);
 8   for (i = 0; i < LOOP_COUNT; ++i) {
 9     art_wait();
10     EtherRTSend(sock, buf, len, 0);
11   }
12   art_exit();
13   EtherRTClose(sock);
14 }
15
16 periodic_recv(unsigned long period,
                 char *name, struct ether_addr *addr)
17 {
18   EtherRTSock *sock;
19   sock = EtherRTSocket(SOCK_DGRAM, ETH_P_RT);
20   EtherRTBind(sock, name);
21   EtherRTConnect(sock, addr);
22   art_enter(ART_PRIO_MAX, ART_TASK_PERIODIC,
             period);
23   do{
24     art_wait();
25   }while(EtherRTRecv(sock, buf, ETH_DATA_LEN,
                        MSG_DONTWAIT) == -1);
26   for (i = 0; i < LOOP_COUNT-1; ++i) {
27     art_wait();
28     EtherRTRecv(sock, buf, ETH_DATA_LEN, 0);
29   }
30   art_exit();
31   EtherRTClose(sock);
32 }
```

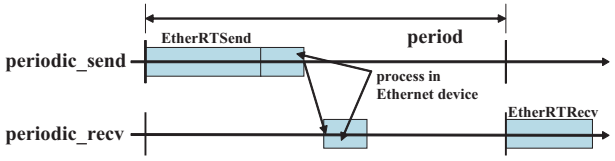


Fig. 7. Time chart of real-time communication program

Here `periodic_send()` is sending side and `periodic_recv()` is receiving side.

The sending side is executed periodically by `art_wait()` and it sends data with `EtherRTSend()` once per period.

The receiving side waits for the arrival of the first data(from line 23 to 25). Calling `EtherRTRecv()` with `MSG_DONTWAIT`, it returns -1 immediately if no data arrives. If data is found, it returns 0 and goes to the line 26. Executing the sending side and the receiving side at the same period, when `EtherRTRecv()` is called at the line 28, data must have already arrived if there is no communication error. Then `EtherRTRecv()` returns immediately. Figure 7 shows a time chart of this communication.

By this means, a real-time communication which send/receive data at the specified period is realized.

This performance evaluation is done using computers which have the following specification and developed SH4 boards, connecting them by a cross cable.

Processor	Intel Pentium 4 2.4CGHz
Chipset	Intel 865G
Memory	DDR SDRAM 400MHz 256MB * 2
Network	Intel 82559

Figure 8 show results on Pentium4 and Fig.9 do on SH4 while 100,000 times communication. The horizontal axis shows the size of data and the vertical axis does the maximum period error/processing time. Measurements is done at several periods, 1000, 500, 200, 100 and 50[μs]. Measurements of 1500[byte] at 100[μs], 1000 and 1500[byte] at 50[μs] are not done since their communications traffic exceeds 100[Mbps] limit.

In the case of Pentium4, Measurement results shows maximum errors are almost all under 2[μs] at the sending and receiving side. Measurements of 1000[byte] at 100[μs] and at 50[μs] has about 8[μs] error. It is considered that it is caused by an overhead of OS. In the case of SH4, the same thing happens in the measurement of 1000[byte] at 200[μs]. Since it takes about 200[μs] to call `EtherRTSend` and `EtherRTRecv`, measurements at 100[μs] and 50[μs] were skipped.

In the case of HRP-3P, 2,208,000$\times$0.001/8=276[byte] data must be transmitted within 1[ms]. Figure 8 and Fig.9 shows that 1500[byte] can be transmitted in 1000[μs] periods. Its maximum period error is about 7[μs] and it is small enough as compared to period.

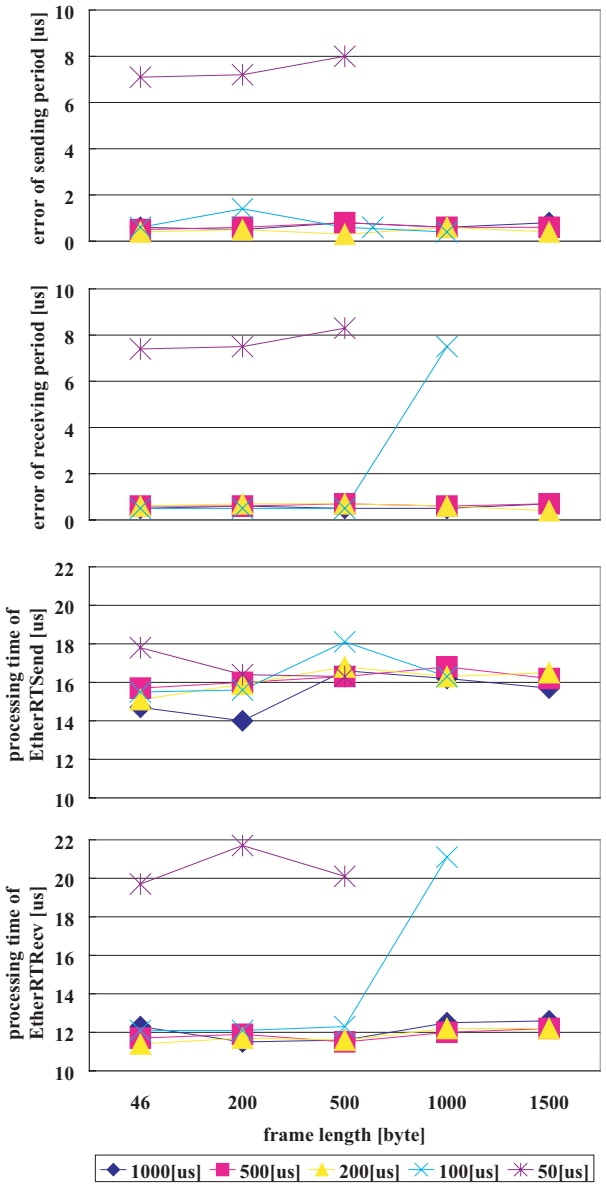


Fig. 8. Measurement result on Pentium4

VI. Real-time CORBA

In order to keep compatibility with existing software, an application layer protocol GIOP of middleware CORBA is implemented on the real-time communication protocol. Users can use this real-time CORBA with a small modification of applications.

Orbix/E[15] from IONA technologies is used as an implementation of CORBA. Orbix/E defines API called OCI(Open Communication Interface) which enable to add a new transport protocol easily(see Fig.10). Implementing these API using a new protocol, an application program can use it with a small modification.

Implementing API of OCI using the real-time communication library, a CORBA which works on real-time Ethernet was realized easily. When using CORBA on embedded

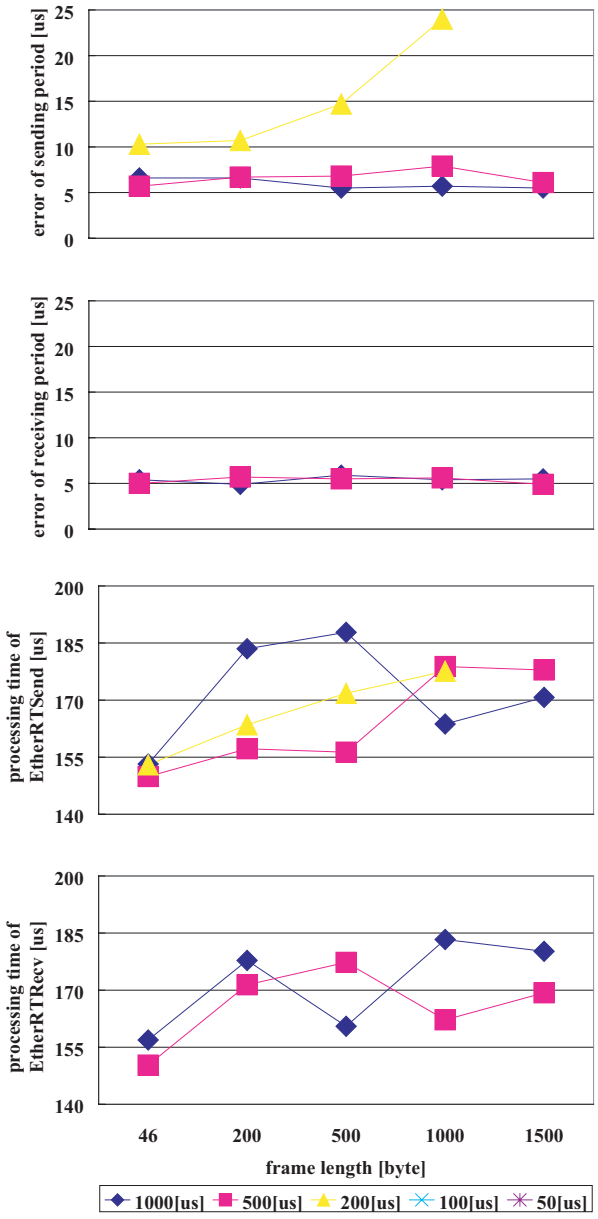


Fig. 9. Measurement result on SH4

systems, there are several problems, memory consumption, processing time and protocol overhead. But in this case, they are not serious problems and convenience of CORBA exceed those demerits. Because Orbix/E is originally developed to minimize its memory consumption and computing power requirement for embedded use the CPU(SH4@240[MHz]) and network(100[Mbps]) of this CPU board are fast enough.

VII. Humanoid Robot HRP-3P

A humanoid robot HRP-3P[16] is a prototype of HRP-3 which is under development aiming a humanoid robot which can work in an actual environment including dusty or wet workplace. HRP-3P has 36 DOF in total. And it also has many sensors to monitor its condition while working in a hazardous

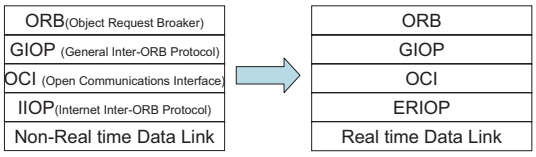
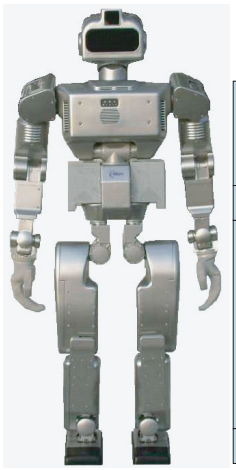


Fig. 10. Difference between conventional and real-time CORBA



Dimensions	Height	1,600[mm]
	Width	664[mm]
	Depth	363[mm]
Weight inc. batteries		65[kg]
DOF		Total 36 DOF
	Head	2 DOF
	Arm	2 x 7 DOF
	Hand	2 x 3 DOF
	Waist	2 DOF
	Leg	2 x 6 DOF
Walking Speed		Up to 2.0[km/h]

Fig. 11. Humanoid robot HRP-3P

environment. As a result, HRP-3 adopted a distributed system configuration in contrast with HRP-2 has a centralized system. Figure 11 shows the appearance and specifications of HRP-3P.

Figure 12 shows network topology and connections between nodes and motors/sensors. 13 nodes are used and they construct three rings. Dashed lines indicates cables which are not used for communication while other communication paths are normal. These rings and two PCs(for motion control and image processing) are connected to a switching hub. PCs don't have redundant connection with the hub. Because PCs and the hub are mounted in the same link and it is expected that cables between them don't suffer stress consequently.

Figure 13 shows a time chart of software for this distributed system. Software for distributed nodes consists of two threads. One of them controls motors, gets sensor outputs at 1[ms] period and sends data to a motion control PC at 5[ms] period. The other receives motion commands from the PC at 5[ms] period. In order to do I/O at precise timing, the sending thread is executed at a higher priority than the receiving one. Software for a motion control PC consists of three threads, (1)sends motion commands which is calculated at the previous period, (2)receives sensor data and (3)calculates whole body motion using sensor data. They run at 5[ms] period and their execution order is controlled by setting priorities.

VIII. Conclusions

This paper realized a real-time communication on Ethernet and developed a distributed I/O system of a humanoid robot HRP-3P using distributed robot controllers and real-time CORBA working on top of them. Contributions of this paper are summarized as follows.

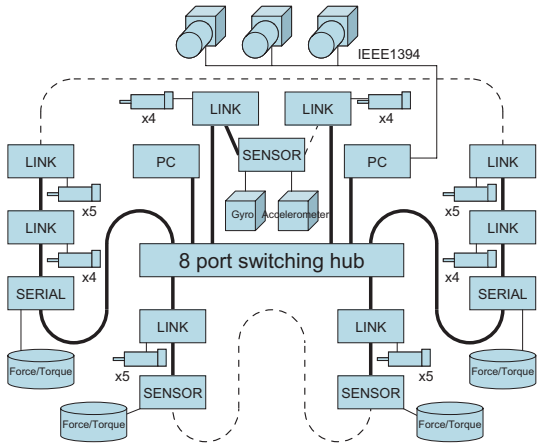


Fig. 12. Configuration of onbody network on HRP-3P

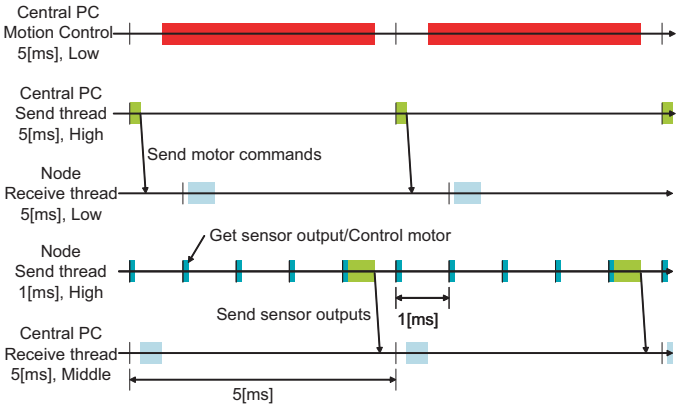


Fig. 13. Time chart of distributed I/O software

1) A real-time communication on Ethernet is realized by ARTLinux. It was enabled by (1)making a processing time predictable by not using TCP/IP and (2)controlling transmission time using a real-time OS. Using Ethernet as a real-time network, it is possible to reduce costs of embedded systems using existing low cost parts and improve development efficiency using existing software.
2) A small-size distributed robot controller is developed. A CPU board and I/O boards are separated and they can be combined according to configuration of robot.
3) A real-time CORBA which runs on real-time Ethernet is developed. It enables existing software to introduce real-time communication easily. A distributed control software for HRP-3P was developed using this. Its effectiveness has already confirmed by a basic walking experiment and a tele-operation experiment.

The distributed robot controllers are now available(only in Japan) through General Robotix[17] and a real-time Ethernet will be commercialized by MovingEye[18] after much further R&D.

REFERENCES

[1] K. KANEKO, F. KANEHIRO, S. KAJITA, M. HIRATA, K. AKACHI, and T. ISOZUMI, "Humanoid Robot HRP-2," in *Proc. of the 2004 IEEE International Conference on Robotics & Automation*, 2004, pp. 1083–1090.

[2] F. KANEHIRO, I. MIZUUCHI, K. KOYASAKO, Y. KAKIUCHI, M. INABA, and H. INOUE, "Development of a Remote-Brained Humanoid for Research on Whole Body Action," in *Proc. of the 1998 IEEE International Conference on Robotics & Automation (ICRA'98)*, 1998, pp. 1302–1307.

[3] T. Furuta, Y. Okumura, T. Tawara, and H. Kitano, "'morph': A Small-size Humanoid Platform for Behavior Coordination Research," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2001, pp. 165–171.

[4] M. Inaba, I. Mizuuchi, R. Tajima, T. Yoshikai, D. Sato, K. Nagashima, and H. Inoue, "Building Spined Muscle-Tendon Humanoid," *Robotics Research, STAR 6*, pp. 113–127, 2003.

[5] Ill-Woo Park and Jung-Yup Kim and Seo-Wook Park and Jun-Ho Oh, "Development of Humanoid Robot Platform KHR-2 (KAIST Humanoid Robot - 2)," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2004, p. 29 paper.

[6] Y. Kuroki, M. Fujita, T. Ishida, K. Nagasaka, and J. Yamaguchi, "A Small Biped Entertainment Robot Exploring Attractive Applications," in *Proc. of the 2003 IEEE International Conference on Robotics & Automation*, 2003, pp. 471–476.

[7] M. Fujita and K. Kageyama, "An Open Architecture for Robot Entertainment," in *Proc. of International Conference on Autonomous Agents*, 1997, pp. 435–442.

[8] Y. Kuniyoshi, G. Cheng, and A. Nagakubo, "Etl-humanoid: A research vehicle for open-ended action imitation," in *Int. Symp. on Robotics Research*, vol. 1, 2001, pp. 42–49.

[9] http://www.automation.fujitsu.com/group/fja/services/hoap/, "Fujitsu Automation Limited."

[10] Y. NAKAMURA and K. YAMANE, "Interactive Motion Generation of Humanoid Robots via Dynamics Filter," in *Proc. of the First IEEE-RAS International Conference on Humanoid Robots*, 2000.

[11] H. Inoue, S. Tachi, K. Tanie, K. Yokoi, S. Hirai, H. Hirukawa, K. Hirai, S. Nakayama, K. Sawada, T. Nishiyama, O. Miki, T. Itoko, H. Inaba, and M. Sudo, "HRP:Humanoid Robotics Project of MITI," in *Proc. of the First IEEE-RAS International Conference on Humanoid Robots*, 2000.

[12] http://www.profibus.com/pn/, "PROFINET."

[13] http://www3.toshiba.co.jp/sic/english/seigyo/tcnet/index.htm, "TCnet."

[14] Y. Ishiwata and T. Matsui, "Development of Linux which has Advanced Real-Time Processing Function," in *Proc. 16th Annual Conference of Robotics Society of Japan*, 1998, pp. 355–356, (in Japanese).

[15] http://www.iona.co.jp/products/orbix_e.html, "IONA Technologies."

[16] K. AKACHI, K. KANEKO, N. KANEHIRA, S. OTA, G. MIYAMORI, M. HIRATA, S. KAJITA, and F. KANEHIRO, "Development of Humanoid Robot HRP-3P," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2005, pp. 50–55.

[17] http://www.generalrobotix.com, "General Robotix, Inc."

[18] http://www.movingeye.co.jp, "MovingEye Inc."