Trans. of the Society of Instrument and Control Engineers
Vol.E-4, No.1, 27/34 (2005)

Real-time Network System by Responsive Processor and Its Application to Bilateral Robot Control

Yutaka UCHIMURA*, Nobuyuki YAMASAKI**,
Takahiro YAKOH** and Kouhei OHNISHI**

This paper presents a new network device named Responsive Processor, plus its message handler, which is a basic protocol for the device. These are designed for a real-time control application such as a robot control. By using the Responsive Processor and message handler, we developed a bilateral robot system that transfers haptic impression over the network.

First, we describe the network system requirements from the view of real-time control. Then, we introduce the main features and functions of the Responsive Processor. Implementation of the message handler, which is designed to maintain real-time performance, is described in detail. Benchmarks for using the message handler with the Responsive Processor are also shown.

We also introduce a bilateral robot system, which transfers haptic impression over the network by the Responsive Processor. The robot manipulator is equipped with a newly devised transmission system, which is free from static friction. The experimental results conducted on the bilateral robot system verified real-time performance of the entire system.

Key Words: computer network, telerobotics, real time systems, protocols, force control.

1. Introduction

Widespread computer networks have already become indispensable to infrastructures. Various information services are now provided over networks, and web browsing or e-mailing have become part of daily life. Until now, most services provided via computer networks, especially through the Internet, have been based on text, graphics and motion pictures. There were few applications that took physical motions and direct interactions with the real world. However, recent research efforts in motion control and computer networks have expanded the research field beyond normal bounds and created a cutting-edge technology such as network-based control. A teleoperation system via computer networks is one of the results of synergy effect of technologies.

A network-based teleoperation provides substantial interaction with the real world. The system allows an operator to move, touch and manipulate a remote object, such as in the promising application of telesurgery [1]. For an application such as a teleoperation, real-time performance of the data communication is a high-priority requirement. Particularly when data communication path comprises a feedback loop, it directly affects the performance of the control system.

This well-known problem has prompted some remarkable research. In the field of teleoperation robotics, Ferrel showed that a kinesthetically coupled teleoperator might become unstable due to time delay [2]. Anderson and Spong used scattering theory to analyze the teleoperation with time delay and introduced an architecture, which transformed communication channel into passive system [3].

Many controller design methods deal explicitly with time delay. One of the conventional methods is the Smith Compensator Method [4], in which the time delay can be removed from the characteristic polynomial of the closed-loop system. Hence the controller can be designed without consideration of the delay-effect. However, the method assumes that the time-delay is constant and predetermined. If it is not, the stability of the system is not guaranteed.

To overcome this constraint, many other methods had been studied. One of the typical approaches is using the method based on the robust control theory, which deals with the variant time-delay factor as an unmodeled uncertainty [5]. In addition, many of the recent papers derived sufficient conditions for stability in the form of LMI form [6]. However, these robust control based methods still require quantization of the worst-case time-delay. This implies that the time delay of the network should ideally be deterministic from the view of control theory.

On the other hand, in the field of the computer net-

* Shibaura Institute of Technology, Tokyo
** Faculty of Science and Technology, Keio University, Yokohama

works, especially in the area of field bus, many studies have been conducted to eliminate uncertainty due to transmission time delay and jitters [7]. One simple solution is TDMA (Time Division Multiple Access) method. However, its transmission efficiency is not always satisfactory. Another solution is priority-based packet transfer. CAN (Control Area Network), which is a de facto standard of the automobile network, is one of the most widely used networks based on this approach [8) 9)]. Its priority-based arbitration mechanism is well organized, however latency due to the carrier sense period cannot be avoided as long as it uses a shared bus structure. These background motivated the development a novel network processor that we named Responsive Processor [10].

In the paper, we describe the Responsive Processor and also present a newly developed message handler. Moreover, a bilateral robot system, as an application of the Responsive Processor based network, is also described.

The following section introduces the main features and functions of the Responsive Processor. In Section 3, we describe a message handler and its structure, which is designed as an interface between the Responsive Processor and host PC. We also show benchmark results. In Section 4, we describe the mechanism and control structure of the network-based bilateral robot system by using the Responsive Processor system. Experimental results of haptic impression transfer are also shown.

2. Responsive Processor

2.1 Background of development

Digital feedback control systems usually require periodic execution of assigned tasks within a pre-determined sampling time. The sampling time is the dead line of such tasks as acquisition of sensors, A/D conversion, control algorithm processing, kinematics and dynamics calculation, motor control and data transfer. Among these tasks, the data transfer may pose an uncertain delay factor, when its transfer path is a computer network. Because most network devices and their media access methods do not always consider the real-time performance in terms of the dead line punctuation.

For example, Ethernet, which is one of the most widely used network methods, is not capable to guaranteeing data transfer within a determined deadline time due to its media access method based on CSMA/CD [11) 12)]. On the other hand, a priority-based media access methods such as CAN is able to send and receive a packet within dead line time since its media access method allows transmission of a higher-priority packet to take precedence over a lower-priority one. The arbitration mechanism of handling packet priority, which uses wired-OR logic, is simple and efficient. However, during the carrier sense process and when the line is in use, even the highest priority packets must wait for the transfer of prior packets. This latency is inevitable as long as the packets go through a shared network line.

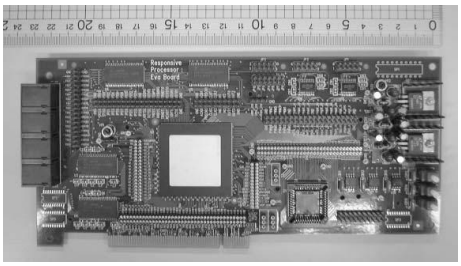

Fig. 1 Responsive Processor board with PCI bus

To resolve these restrictions, we developed the Responsive Processor, a novel network processor that integrates MPU core (SPARC) plus Responsive Links for real-time network communication. **Fig. 1** shows photo of the Responsive Processor board with PCI bus.

The Responsive Processor was originally designed for an embedded control application. Therefore, its target area ranges from the field bus of automation to the local area network and teleoperation range is within that extent for now. But its design and specification do not restrict the scale and area of application. Currently, the standardization process is now under way by ITSCJ (in Japan) and ISO/IEC JTC 1/SC 25. We expect that application range will expand to the wide area network in the future.

2.2 Responsive Link, a key device of the real-time network mechanism

The Responsive Processor is equipped with MPU, SDRAM I/F, DMAC, PCI I/F, USB I/F, a network interface and additional functions in an ASIC chip. Among them, the network interface, which we named Responsive Link, has the key role in real-time data communication.

The most significant feature of the Responsive Link is that it has two separate communication links as shown in **Fig. 2**. One of them, the Event Link, is designed for short-latency packet transmission. The other is the Data Link, which is for high-bandwidth data communication. Both of them support full-duplex up to 100 Mbps.

In order to have a clear distinction between the role of the Event Link and the Data Link, let us show an example of a network based control application. **Fig. 3** shows a bilateral robot system, which is connected via the net-

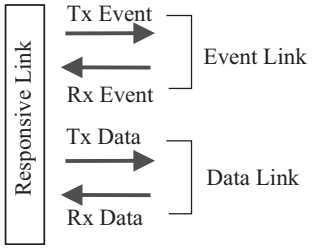

Fig. 2 Interface of the Responsive Link

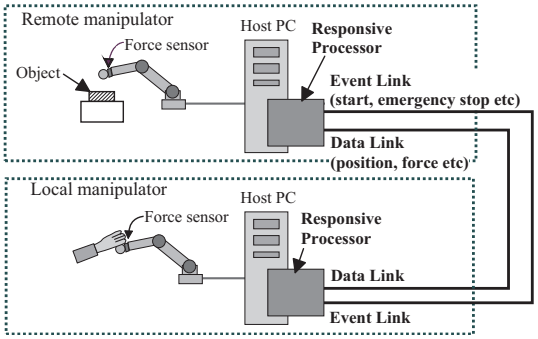

Fig. 3 System structure of the bilateral robot system

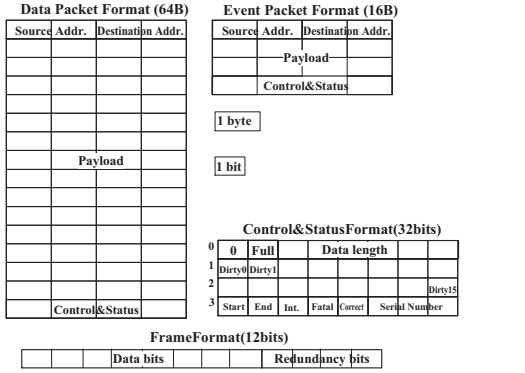

Fig. 4 Packet format

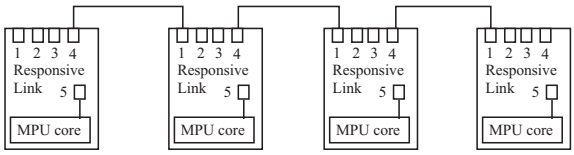

Fig. 5 Example of configuration

work by the Responsive Link. The bilateral robot system is an implementation of a teleoperation with force feedback. When an operator moves the local manipulator, the remote manipulator draws the same motion. Once the remote manipulator touch an object, the operator feels it by haptic impression.

In the case of the bilateral robot, the Event Link is used to transfer the signal, which is associated with state transition of the system, such as start, initialization, stop, change the control state and emergency stop. On the other hand, the Data Link is used to transfer the state variable data of the control, such as position, velocity and force data. Since the two links are physically isolated, an emergency stop signal on the event link, which is highly urgent, would never be delayed during the Data Link is busy.

For an application, such as transferring a clock synchronization signal, a periodic beacon sent out by a master node, the Event Link works fine, even when the Data Link is busy to transfer broadband data such as sound and video image.

2. 3 Packet format and topology

A packet used on the Event Link is called an event packet, and a packet on the Data Link is called a data packet. The size of both packets is fixed; an event packet is 16 bytes and a data packet is 64 bytes.

Fig. 4 shows the packet format. An event packet consists of 4-byte header, 8-byte payload and 4-byte trailer. Similarly a data packet consists of 4-byte header, 56-byte payload and 4-byte trailer. To maintain real-time performance, data resending should be avoided, so each byte has additional 4-bit redundancy bits that correct one-bit errors at each node by hardware.

A Responsive Processor is equipped with 5 Responsive Links, one of which is connected to the MPU core. The remaining links are used to compose point-to-point network configurations. The daisy chain like connection in **Fig. 5** is an example of possible configurations. Because shared bus topology is not a possible configuration, collision on the line does not occur. Instead, if a collision in a node, the higher-priority packet 'wins' and overtakes the lower-priority one.

2. 4 Routing table

Fig. 6 shows a routing table. The reference part of the table includes the source address and destination address together with the priority of the packet. The EE and DE bits in the referent part correspond to event packet enable and data packet enable. These bits make it possible to allocate different routes for data and event packet respectively. PE indicates priority change enable. When PE is set, the priority of the packet is changed to the value of P0, P1 bit. L0···L4 bit shows the output link number. At L0, the packet is delivered to the MPU. Setting more than two bits corresponds multicast and all of them are broadcast.

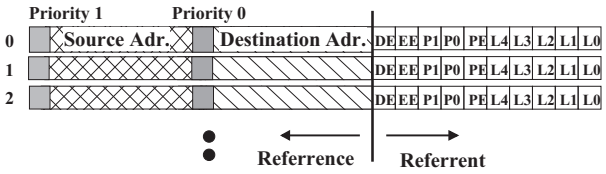

Fig. 6 Routing table

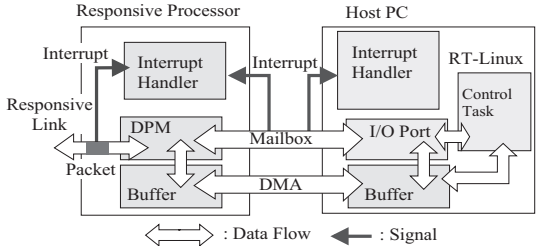

Fig. 7 Overview of the message handler structure

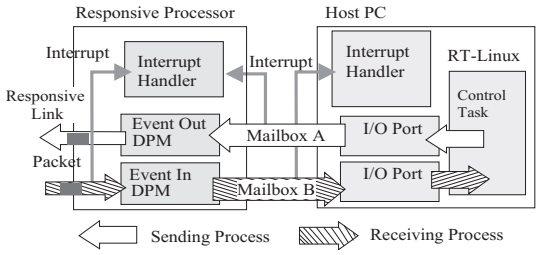

Fig. 8 Flow process of the sending and receiving an event packet

3. Message handler for the Responsive Processor

3.1 Overview

An early version of the of the Responsive Processor was designed with the objective of realizing a one-chip embedded chip; therefore there was a board equipped with A/D, D/A converters, PWM generators and pulse counters. However, the number of channels of the devices and speed of the MPU were sometimes not enough for a complicated control application. For now, a typical configuration of the Responsive Processor board consists mainly of MPU core, the Responsive Link and PCI I/F. Execution of control algorithm and interface device boards are installed on a host computer (PC).

In order to handle communication between the Responsive Processor and the host PC, we designed a message handler, taking into consideration maintaining real-time performance. **Fig. 7** depicts an overview of the message handler structure.

As previously mentioned, the Responsive Link consists of two separate communication links, and for that reason there exist two pairs of sending and receiving process. The principal design rule is that the handling of an event packet takes precedence over that of a data packet at any time. When an event packet arrives during the handling of a data packet, the data packet task should be suspended immediately and the handling process should switch to the event packet task.

For example, when a signal such as switching motion and emergency stop arrives via the event link, it takes precedence over the current data packet handling process. For this reason, we chose RT-Linux for the operating system of the host PC to improve total real-time performance (RT-Linux is a real-time extension of Linux kernel) [15].

The methods for passing data between the Responsive Processor and the host PC are dual mailboxes and DMA transfer. One of the mailboxes, which we call mailbox A, is used for passing 4-byte data from the host PC to the Responsive Processor. The other, mailbox B, is for passing data from the Responsive Processor to the host PC. These mailboxes can function as doorbells, which generate interrupt, when a message is written. The event packets are passed only by using this mailbox. On the other hand, data packets are passed by using DMA transfer (parameters and start trigger are sent via mailbox). This is because event packets require short latency but data packets require high throughput.

3.2 Event packet handling procedure

Fig. 8 illustrates the flow process of sending and receiving an event packet. Details of the procedure are described below.

(1) In the case of sending, a payload generated by a control task (thread of RT-Linux) in host PC, is written into mailbox A via I/O port register.

(2) The interrupt handler of the Responsive Processor hooks the interrupt signal, then generates a packet and sends it to the DPM (dual port memory) area.

(3) The Responsive Link sends the packet.

(4) When an event packet arrives, the Responsive Link generates an interrupt.

(5) The handler in the Responsive Link hooks it, then judges the content and writes the data to mailbox B.

(6) The real-time interrupt handler hooks the interrupt by mailbox B, then invokes the event packet receiving thread in the control task, which is allocated as high priority.

3.3 Data packet handling procedure

In the case of data packets, data buffers are allocated to improve throughput, since frequent DMA transfers of

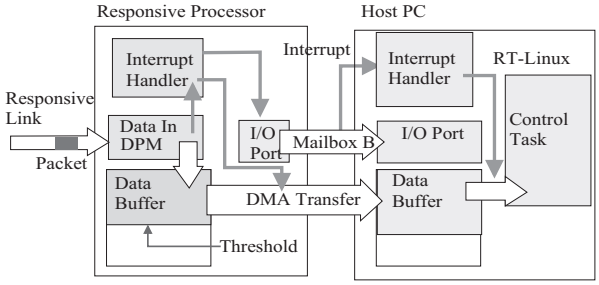

Fig. 9 Flow process of receiving data packets

Table 1 Benchmark results

	Mean (μs)	Std (μs)
Event packet	247.7506	6.3821
Data packet	663.3494	20.8979

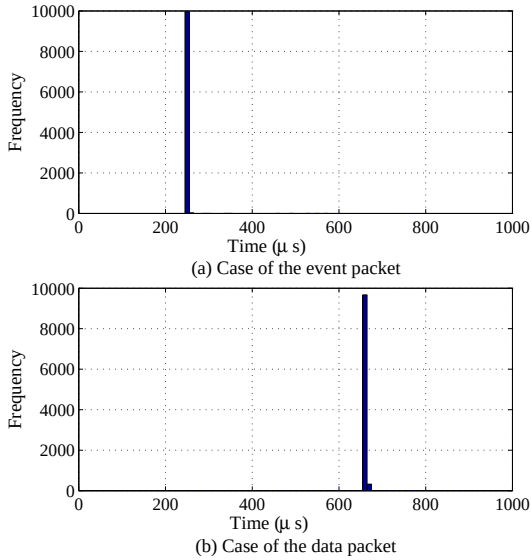

Fig. 10 Histogram of RTT

small-sized data are not efficient due to the overhead of the DMA process.

Fig. 9 illustrates the flow process of receiving data packets via the data link. Details are provided below.

(1) When a packet arrives at the Responsive Link, it is placed in the DPM.

(2) When the current packet number, which indicates the pointer in the DPM, increases and once exceeds the predetermined threshold, an interrupt occurs.

(3) The interrupt handler copies the packets to the buffer allocated in the Responsive Processor's memory.

(4) Once the size of data in the above buffer exceeds the predetermined threshold, the DMA controller transfers the data to the host PC 's buffer memory.

(5) When the DMA transfer is finished, the interrupt handler sends a message via mailbox B to the host PC indicating that the data is ready.

The sending tasks are processed in the same manner as the receiving process except the reversed direction of data flow.

In practice, the size of the buffer and its threshold size are adjustable depending on the application. If the data size is small and they requires shorter latency, the packets should be transferred directly from the DPM to the buffer in the host PC and it would be implemented in the interrupt handler. The data packets of the bilateral control are processed in this manner.

3. 4 Benchmarks

To evaluate the total performance of developed message handler with the Responsive Link, benchmarks were carried out. We set up two host PCs with a Responsive Processor for each and connected them with category 5 UTP cable. Benchmarks measure the RTT (round-trip time) from the moment when an application sent out a message to the moment when it is received. The number of times of RTT measurements is 10,000 times for each. The packet size is fixed to one packet; i.e. 16 bytes for event 64 bytes for data. But in the case of event packet, only head 4 bytes of the payload are passed to the host PC in order to evaluate maximum performance. Because the size of the mailbox is 4 bytes and overhead time due to the pre-process of interrupt handling is relatively large.

Table 1 shows the results of mean time of RTT and the standard deviation of it. Histogram of the event packet case is shown in **Fig. 10** (a) and the data packet case is shown in Fig. 10 (b).

The results show the RTT of event packets are much shorter than that of data packets. Additionally, less standard deviation corresponds to fewer jitters. This validates the superiority of the event packet in real-time performance. On the other hand, throughput of data packet is larger than event packet, and RTT is less than 1 millisecond, which is sufficiently short for a robot control application.

Theoretically, there exits a break-even point in packet transfer efficiency. Suppose the size of payload is S_p, the RTT of event packet is T_e , the size of mailbox is M, the RTT of data packet is T_d and the size of threshold of data buffer D. The event packet maintains shorter latency as long as (1) is true.

$$\frac{S_p T_e}{M} < \frac{S_p T_d}{D} \tag{1}$$

This condition supports the design philosophy of the Responsive Link, which manages the trade-off between latency and throughput.

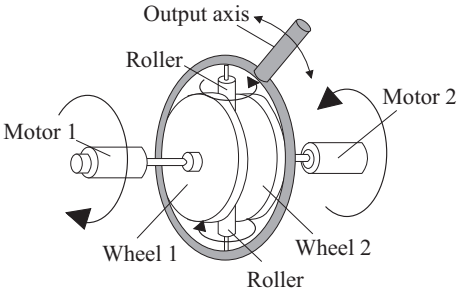

Fig. 11 Mechanical overview of the twin drive system

Table 2 Specification of the twin drive system

Moment of inertia	$2.965 \times 10^{-5}(kgm^2)$
Coefficient of viscous friction	$2.240 \times 10^{-4}(Nm/rad)$
Coulomb friction	$1.527 \times 10^{-3}(Nm)$
Servo rigidity	$544.4(Nm/rad)$

4. Application to a bilateral robot system

4.1 Overview of a bilateral robot system

To examine the performance of the developed network system when applied to a real-time control application, we developed a bilateral robot system.

As already shown in Fig.3, the bilateral robot system comprises two robot manipulators having identical mechanical structure [13]. Each robot manipulator has a multi-degree of freedom. For convenience, we call the robot manipulator which the operator maneuvers a local manipulator, and the other one a remote manipulator. At each joint, a newly devised static friction free transmission mechanism is installed.

4.2 Static friction free transmission

For the bilateral robot system, which is designed to transfer haptic impressions, static friction is a serious problem. In order to resolve the problem due to static friction, authors devised a transmission mechanism, which is named twin drive system [14].

Fig. 11 shows the mechanism of the system. The device is composed of two motors, wheels, rollers and an output axis which is connected to a robot arm. Each wheel driven by each motor rotates the rollers. The rollers sandwiched by two wheels with pressure are connected to the output axis; hence the output axis is driven by the speed difference of two motors. The rotation speed of the output axis is half of the speed difference. Consequently, it stands still while the two motors are spinning with exactly the same speed.

Theoretically, static friction does not arise because all the traction elements are always spinning. This static friction free effect greatly extends the range of motion control

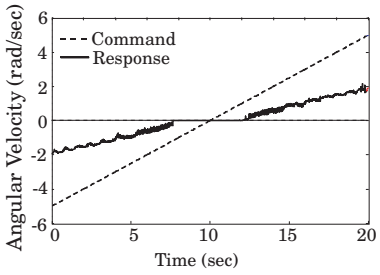

Fig. 12 Speed response of a usual motor

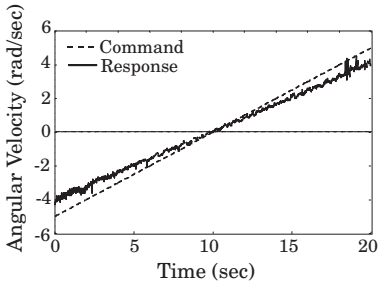

Fig. 13 Speed response of the twin drive system

applications.

Specification of the twin drive system is shown in **Table 2**. To clarify the effect of twin drive, ramp speed responses for a usual motor (rotating single motor) and the twin drive system (rotating both motors) are shown in **Fig. 12** and **Fig. 13** respectively. For the small velocity command, the usual motor stopped due to static friction while the twin drive system rotated continuously. These results verified the static friction free effect of the twin drive system.

4.3 Control system

Each manipulator has a force sensor mounted on the tip of the top link to measure interaction forces. The force sensor on the local manipulator measures the reaction force between the manipulator and operator, while the sensor on the remote side measures the reaction force between an object. A PC-based host computer controls each manipulator, and the networking by the Responsive Processor connects two PCs.

Fig. 14 shows a block diagram of the control system. This bilateral control scheme is based on modified version of the force reflecting servo type, whose robust stability against stiffness variation of the object is improved comparing to the conventional type. It is achieved by implementing force feedback loop both on the remote and local manipulator.

Notations in the figure are as follows: Z_l and Z_r are mechanical impedance models of an operator and an environment respectively, G_l and G_r are plant models of

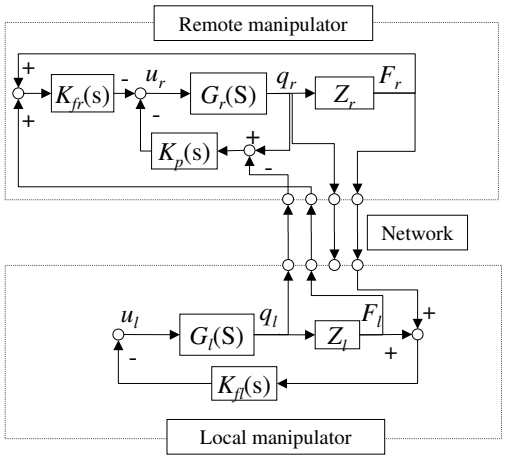

Fig. 14 Block diagram of the control system

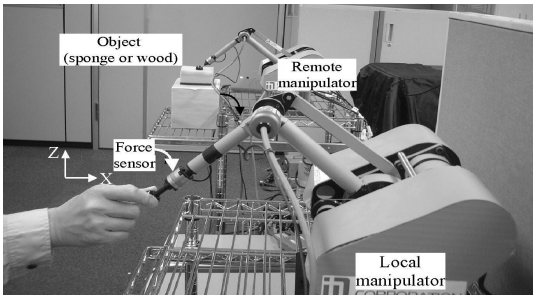

Fig. 15 Overview of the experiment

manipulators; K_p is the position controller; and K_{fl} and K_{fr} are force controllers, which are designed to achieve the target in (2).

$$\text{minimize } |F_l + F_r| \tag{2}$$

As already explained, force and position data of the manipulators are transferred using data packets while state transition signals such as an emergency stop use the event packets. The sampling time is 1 millisecond, which is short enough for the dynamic specifications of the system.

4.4 Experimental results

We set up two objects whose stiffness are different, a wood plate and a sponge block. In the experiments, an operator maneuvers the local manipulator and pushes an object several times via the remote manipulator. **Fig. 15** shows the overview of the experiment.

Fig. 16 and **Fig. 17** show the results for two different objects. Fig. 16 is the case of a sponge block and Fig. 17 is that of a wood plate. In each figure, the upper graph shows the force response in the direction of the z-axis (the gravity direction). For convenience, the sign of the local manipulator is reversed. The lower graph shows the tip position of the manipulator in the direction of the z-axis.

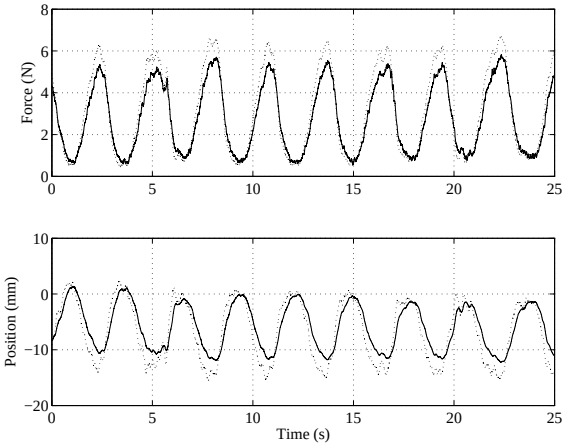

Fig. 16 Reaction force and position against sponge block (solid: remote, dotted: local)

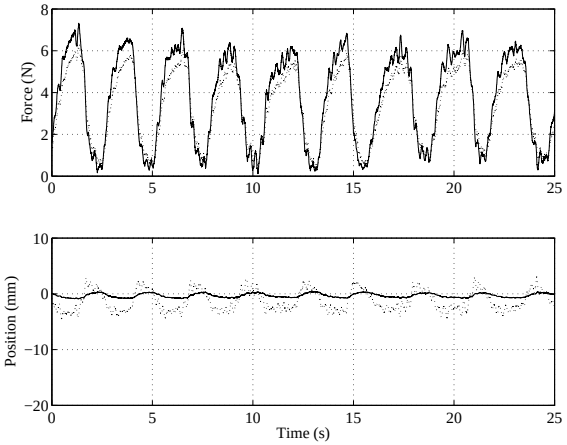

Fig. 17 Reaction force and position against wood plate (solid: remote, dotted: local)

In both cases, the force data show similar responses, in which the magnitudes reach approximately 6N. On the other hand, the peak-to-peak variation of the position value is much larger in the case of the sponge block than that of the wood plate. This implies that the operator felt the sponge was softer because the deformation was larger.

Consequently, the results verify that the system successfully transferred haptic impressions via the network and also proved that the developed networks system is applicable to a real-time control application.

5. Conclusion

In this paper, we introduced the Responsive Processor, a unique network device that accomplishes real-time networking. A newly developed real-time message handler for the Responsive Processor is also described. Benchmark results, which were carried on two separate links,

data link and event link, showed sufficient real-time performance.

A network based bilateral robot system, which transfers haptic impression via the Responsive Processor was developed as a real-time application. The robot manipulator is equipped with a newly devised transmission system, which is free from static friction. The experiment conducted on the robot system showed good performance and the results verified that the system transferred haptic impressions via developed real-time network system.

Acknowledgment

This study was performed through Special Coordination Funds of the Ministry of Education, Culture, Sports, Science and Technology of the Japanese Government.

References

1) M. C. Cavusoglu, W. Williams, F. Tendick, S.S. Sastry.,"Robotics for Telesurgery: Second Generation Berkeley/UCSF Laparoscopic Telesurgical Workstation and Looking towards the Future Applications.",Proceedings of the 39th Allerton Conference on Communication, Control and Computing, Monticello, IL, October 3-5, 2001.
2) W.R.Ferrel, "Remote manipulation with transmission delay", IEEE Transactions on Human Factors in Electronics, HFE-6:24-32, September 1965.
3) Anderson, R. J.; Spong, M. W., "Bilateral Control of Teleoperators with Time Delay," IEEE Transactions on Automatic Control, v34, n5 pp. 494-501,May 1989.
4) O.J.Smith, "A controller to overcome dead time", ISA J., vol.6, no.2,pp.28-33, Feb. 1959.
5) Gary M. H. Leung, Bruce A. Francis, Jacob Apkarian, "Bilateral Controller for Teleoperators with Time Delay via μ-Synthesis", IEEE Transactions on Robotics and Automation, vol. 11, no. 1, pp. 105–116,1997.
6) M. Jun and M. G. Safonov, "Stability Analysis of a System with Time-delayed States", IEEE American Control Conference, pp. 949-952, 2000.
7) K. Tindell and A. Burns and A. Wellings, "Analysis of Hard Real-Time Communications", Real-Time Systems ,volume 9,pp.147-171,1995.
8) Controller Area Network (CAN), http://www.can-cia.de/can/.
9) K. Tindell and H. Hansson and A. Wellings, "Analysing Real-Time Communications: Controller Area Network", IEEE Proc. 15th RealTime Systems Symposium, pp.259–263,1994.
10) N. Yamasaki, "Responsive Processor for Parallel / Distributed Real-Time Control", International Conference on Intelligent Robots and Systems, pp.1238-1244, 2001.
11) J.D. Wheels,"Process control communications: Token bus, CSMA/CD, or token ring ?", ISA Transactions vol. 32 , pp.1933-198, 1933
12) Seok-Kyu Kweon et.al,"Statistical Real-Time communication over Ethernet for Manufacturing Automation Systems", Real-Time Technology and Applications Symposium,pp.192-202,1999
13) Y.Uchimura,T.Yakoh,K.Ohnishi,"Bilateral robot system on the real time network structure",IEEE. International Conference on Advanced Motion Control, pp.63-68, 2002.
14) N.Hayashida,T.Yakoh,T.Murakami and K.Ohnishi , "A Friction Compensation in Twin Drive System", IEEE International Conference on Advanced Motion Control (AMC2000) pp187-192, 2000.
15) V. Yodaiken and M. Barabanov. "A Real-Time Linux. In Proceedings of the Linux Applications", Development and Deployment Conference(USELINUX), Anaheim, CA, January 1997.

Yutaka Uchimura (Member)

received the B.E. , M.E. degree in electrical engineering and Ph.D. degrees in integrated design engineering from Keio University, Yokohama, Japan, in 1989, 1991 and 2005 respectively. From 1991 to 2007, he was a senior researcher of Kajima Technical Research Institute, Tokyo, Japan. Since 2007, he has been with Shibaura Institute of Technology, Tokyo, Japan, where he is cuttently an Associated Professor. His main research areas are robotics, real time systems, computer networks and distributed systems.

Nobuyuki Yamasaki

received the B.S. degree in physics, and the M.S. and Ph.D. degrees in computer science from Keio University, Yokohama, Japan, in 1991, 1993, and 1996, respectively. Since October 1998, he has been with the faculty of information and computer science, Keio University, where he is currently an Associated Professor. His research interests include real-time systems, parallel/distributed systems, computer architecture, robotics, and VLSI design.

Takahiro Yakoh

received the B.E. degree in instrumentation engineering and the M.E. and Ph. D. degrees in computer science from Keio University, Yokohama, Japan, in 1989, 1991, 1995, resepectively. Since 1998 he has been with the Department of System Design Engineering, Keio University, where he is currently an Associate Professor. His main research area are communicating real-time systems, computer networks, multimedia communication, man machine interaction, and robotics.

Kouhei Ohnishi (Member)

received the B.E., M.E., and Ph.D. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1975, 1977, and 1980, respectively. Since 1980, he has been with Keio University, Yokohama, Japan. His research interests include mechatronics, motion control, and robotics.

Reprinted from Trans. of the SICE
Vol. 40 No. 2 199/206 2004