2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

A novel real-time control architecture for Internet-based Thin-client robot; Simulacrum-based approach

Kyung Jin Kim, Il Hong Suh, Sung Hoon Kim and Sang Rok Oh

Abstract—**There are many difficulties to remotely control a robot over Internet such as transmission time delay, limitation of bandwidth, and packet loss. In this paper, we propose a novel remote control architecture and a communication architecture for Internet-based robots, which is insensitive to the transmission time delay and packet loss. The proposed architecture guarantees enhanced remote control by providing the perfect copy of the status of remote robot at real-time. The communication protocol has a flexible packet format and various packet sizes at runtime for robust and reliable control mechanism. The architecture together with the protocol provides improved control performance. Experiments shows that the effectiveness and applicability of the internet-based robot with the proposed architecture.**

I. INTRODUCTION

Internet-based robot control system considered in this paper accomplishes given tasks with controlling the robot over Internet at real time. It consists of a remote robot, a local system which delivers effecter commands based on remote sensors and provides user interfaces or programming methodology, and a protocol through which information is exchanged between the local system and the remote robot [1].

Traditionally, remote control systems used a dedicated communication network that lacks extensibility or accessibility. However, since first Internet-based robot Cambridge coffeepot in 1994, Internet-based robots have been progressing [2].

The growth of the Internet and ubiquitous technology over the past years has extensively provided many convenient applications in daily life [3]. These trends are caused by the advantages of Internet, such as cheap price, accessibility at anywhere and no specialized hardware. Particularly, the Internet technology which is limited to passive or virtual world has been extending the application area to the real-world control along with remote control system. Also the service area for Internet-based robot becomes diverse with external sensors connected to Internet as well as server-based cognitive functions such as speech recognition and face recognition. Internet-based robot will more and more have deep impacts on human life style [4] [5].

Global market size of home service robot such as cleaning robot or entertainment robot is growing rapidly about 200% every year. According to the International Federation of Robotics, there were 1.9 million service robots in domestic in 2005. South Korea is pumping about $50 million into research every year for Internet-based service robot as one of its 10 key economic drivers from 2004. Through this project, the South Korea has been developing various service robots and spreads to people. Low price and high intelligence are key points to success the project. Because the Internet-based service robot is the solution of low price and high intelligence. But the low price robot with low cost processor has no brain to decide what to do next. Therefore, the guarantee of real-time and transmission time delay is one of the most important challenge. Thin-client robot is very interesting and important issue in Internet-based service robot field because it can be applied to various services including multimedia with action.

In spite of many advantages of Internet, remote control systems suffer from many problems caused by uncertain network traffic such as transmission time delay, limitation of bandwidth, and packet losses because of heavy traffic [1][2][6][7]. Because these drawbacks are unpredictable, the direct control of a remote robot is so difficult and challengeable research [5].

To overcome the drawbacks several control methods were proposed such as shared control, distributed control, supervisory control, and user interfaces for examples predictive display and pre-simulation [6][8][9][10][11]. Even though these architectures were effective to robots with sufficient computing power, they are not regarded as being for brainless thin-client robot. Since other researchers were interested in improving communication protocols such as TCP or UDP to minimize propagation delays or jitters in Internet [1][13][14]. Almost all the methods improve only the performance of a transport protocol, but the protocol must be optimized by the characteristics of a remote robot because the transmission packet size can be dynamically changed. The size of protocol must be flexible at run-time to reduce the transmission data size.

In this paper, we propose a novel simulacrum-based architecture for real time control for Internet-based service robot. We define the simulacrum, which is new concept proposed in this paper, as snapshot or perfect copy of the status of remote robot. Simulacrum means not communication object but the mechanism for representation of a remote robot.

Kyung Jin Kim is CEO/CTO of ROBOMATION company (url: http://www.robomation.co.kr, e-mail: jin_khim@hanmail.com).

Il Hong Suh is with College of Information and Communications, Hanyang Univ. (e-mail: ihsuh@ hanyang.ac.kr).

Sung Hoon Kim is with Electronics and Telecommunications Research Institute and is also working toward Ph.D at College of Information and Communications, Hanyang Univ.

Sang Rok Oh is IT policy advisor and project manager of Intelligent service Robot and Next Generation PC, Ministry of Information and Communication, Republic of KOREA (email: sroh@mic.go.kr, sroh@iita.re.kr)

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

The simulacrum of remote robot conceptually resides in local system and users can remote control and access sensor data using the simulacrum of a remote robot in the local system.

This paper is organized as follows. In Section II, the Simulacrum-base control architecture is shown. In Section III, The data communication protocol between the local system and the remote robot is explained. Section IV presents the experimental results and conclusion remarks follow in Section V.

II. SIMULACRUM-BASED CONTROL ARCHITECTURE

A. *Thin-Client Robot vs Thick-Client Robot*

Internet-based robot can be categorized based on their dependency on local system for task completion: thick-client robot and thin-client robot.

As shown in Fig.1, a thick-client robot can plan the sequence of actions for a given task. Moreover it has the capability to manage environmental changes reactively and autonomous as well as to perform high-level abstracted command such as go to the front of television and go to the sofa because it has sufficient computing power to process sensor data and control motor. A control mechanism of the thick-client robot is similar to open-loop control mechanism. Generally, the thick-client robot is able to make motor actions by processing sensor data such as an action to avoid obstacle. In some situation, a thick-client robot can operate even without the help of the local system.

On the other hand, a thin-client robot is not able to produce its action plans and even does not have the reactive control capability. Therefore, it depends on its local system connected over the Internet for all kinds of control, as shown in Fig 2. Generally, a thin-client robot is low price and has low level processor.

B. *Architecture Description*

The terms simulation and simulacrum have subtly different meanings. Simulation is defined as mimics the real world through mathematical modeling so that we can preview how a robot behaves in the real world. In comparison, the simulacrum is defined as "a material image, made as a representation of some deity, person, or thing," as "something having merely the form or appearance of a certain thing,

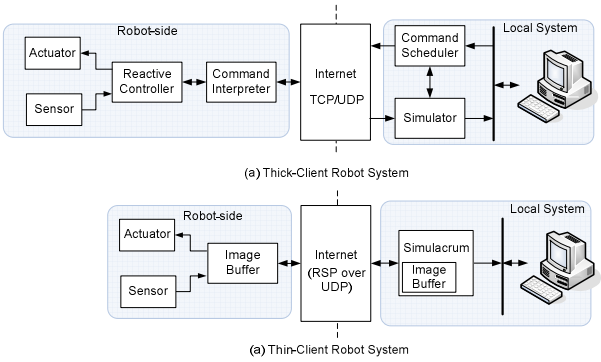


Fig. 1. Thick-Client vs. Thin-Client Robot

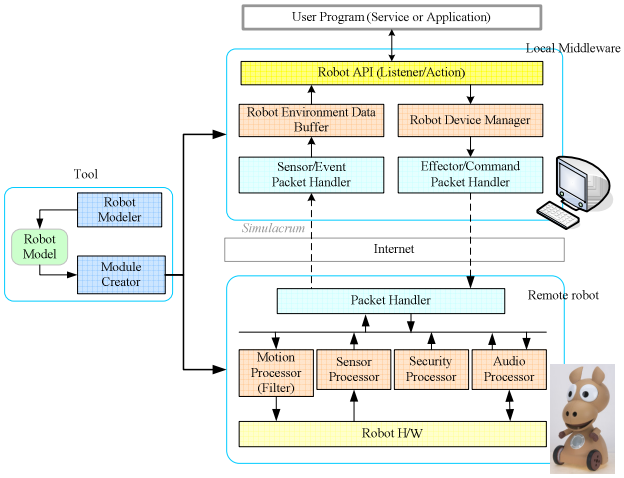


Fig. 2. Simulacrum-based remote robot control architecture

without possessing its substance or proper qualities," and as "a mere image, a specious imitation or likeness, of something" [15]. The proposed architecture uses the concept of simulacrum.

Fig. 2 shows the simulacrum-based control architecture proposed in this paper. The simulacrum is an image or the snapshot of the status of its remote robot in every moment. In fact, it works as a kind of virtual robot that receives all sensor information and transmits command to its remote robot in real-time at every moment. Note that it is not about communication of meaning but rather about the representation of the representations of robot. Moreover, the simulacrum means a real-time copy of remote robot in every instance whereas a simulation only predicts a robot. Moreover, the transferred simulacrum is the only difference of image between two systems so that the transmission overhead can be reduced.

In the remote control side thick-client robot is mainly required to use prediction technology at both local system and remote robot [6]. The simulator in the thick-client system provides environment sensor data and delivers commands to its remote robot [6]. However, it is not appropriate because its robot has no predictive function or reactive controller.

As shown in Fig. 2, the architecture consists of three sub systems: a tool, a local middleware, and a remote robot. The local middleware is composed of five modules: *Robot API, Robot Environment data buffer*, *Robot Device Manager* and two types of *downloading and uploading packet handlers*. A robot model is defined by using a GUI tool and produces the code of the *local middleware* and *packet handler* for remote robot. The *local middleware* provides the programming APIs (Application Programming Interfaces) and handles of packets.

The *module creator* in Fig. 2 generates the code of the *local middleware* and the *packet handler* using the XMI file. Though some researches using robot models can generate codes for robot control, the codes are limited to common APIs not communication protocol.

The mounted devices and capabilities of a remote robot are

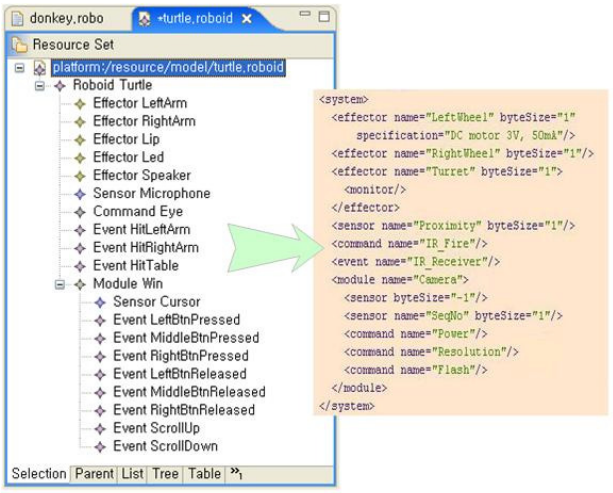


Fig. 3. Robot Profile Modeler GUI and Generated XMI

represented by robot model as shown in Fig. 3. It is automatically generated by the tool implemented as a plug-in module of eclipse. Users can easily generate robot models with this tool. XMI is used for the formal description of robot model. A model is composed of eight parts: *effector*, *sensor*, *monitor*, *command*, *module*, *event, memory command,* and *memory event*. Their detailed description is shown in TABLE I. The module activates or deactivates all defined sensors and actuators.

The *Robot API* module in the *local middleware* is a set of APIs common in thin-client robots. For event processing, it provides event processing mechanism. A user access the sensor and event data and send commands to a remote robot only using these API. Fig. 4 is example code generated

TABLE I
ROBOT MODEL ELEMENTS

Element Name	Description	Payload Size
effector	Input elements (motor, LCD, LED and etc)	user defined
Sensor	Output elements (proximity, luminance sensor and etc)	user defined
Monitor	Monitors the states of effectors	Same size of effector
Command	User commands which is must be transferred (user transfer a effector data using *Command*) Local system transfer the *Command* successively until receive ACK)	ID length + user defined
Module	Set of elements (effectors and sensors)	0
Event	A kind of sensor generated only when the condition is met. Transferred from robot to local	ID length + user defined data length
Memory Command	Same meaning with *Command* Local system transfer the *Command* only once until receive ACK	ID length + user defined data length
Memory Event	Same size of payload and meaning with *event*. Transmitted only once before notification from local	ID length + data length + user defined data length

```
public class LeftWheel_Effector()
{
    public void write(int index, int value) ;
    public Device read(int[] in);
    public int[] read();
    public int read(int index)
    public void setACTIVATED(boolean activated)
    public void setACTIVE(boolean active)
    ...
}
```

Fig. 4. Auto generated effector API

LeftWheel_Effector API.

The *Robot environment data buffer* stores sensor data transmitted from a remote robot. And, the consumer in *Robot API* is waited until the sensor data is arrived. Also, when user registers listeners for interested sensor data, the buffer fires the event to listener when it is received (eg, button pressed).

The *Robot Device Manager* temporally stores commands and the values of the effectors. The *Effector and command packet handler* prepares download packets at every '*T*' second with the data stored in the *Robot Device Manager***.** Moreover, the *Sensor and Event Packet Handler* parses and transfers the upload packet from the remote robot to the buffer. '*T*' is a transmission rate in our architecture and optimized according to the application at run-time.

The software modules in remote robot are *packet handler* and four types of processors: *Sensor***,** *Command***,** *Security,* and *Audio*. The *packet handler* interprets download packet and manage the corresponding processor. The *motion processor* activates selected effectors and executes the commands. If the transmission time delay becomes larger, several commands can be received almost simultaneously as shown in Fig. 5. Accordingly, the *motion processor* need to interpolate the data loss and correct the cumulative commands [6]. According to [6], the transmission time delay T_d for *k*th data transmission can be described as follows:

$$T_d = \sum_{i=0}^{n} [{}^{l_i}\!/_{C} + tr_i + tl_i(k) + {}^{M}\!/_{b_i}] \quad (1)$$

where l_i is the *i*th length of internet link, C is the speed of light, tr_i is the routing time of the *i*th node, tl_i is the delay caused by the *i*th node's load, M is the amount of data, and *bi* the bandwidth of the *i*th link. The main factor cause of

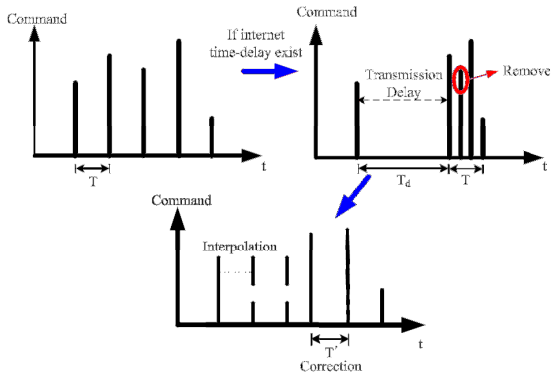


Fig. 5. The function of Motion Filter

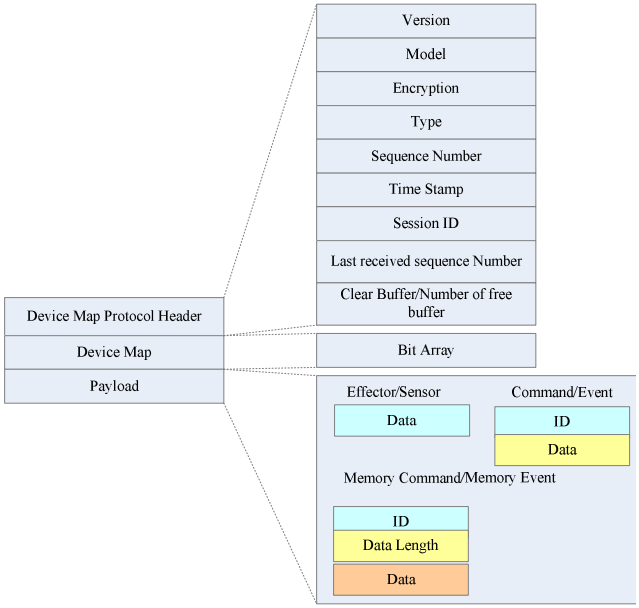


Fig. 6. Robot Simulacrum Protocol packet format

transmission time delay is the load of internet nodes. The term of tl_i makes T_d unpredictable because we cannot estimate the load of a node. The *motion processor* interpolates the missing data with the last received data. In this way, a remote robot can maintain its posture until a new command is received.

The *Sensor processor* in remote robot returns the requested sensor data. The *Security processor* manages robots ID and provides encryption capability. The *Audio processor* plays multimedia data fragmented into pieces of small size. A user can send multimedia data along with motion using this protocol.

III. RSP (Robot Simulacrum Protocol)

Fixed packet format and size are less effective and sometimes make problems to control a remote robot which has small communication packet size and short transmission rate (i.e. 40*msec* or *20msec*). RSP, the communication protocol, minimizes overheads caused by fixed packet size by varying packet size and creating the packet format dynamically according to the robot model at development time. The packet format described in Fig. 6 is automatically generated by the tool using a robot model. At running time, the payload data can be omitted when the device bit in *Device Map* has zero value. Both the local system and the remote robot transmit only the requested data always.

TCP is the most popular and reliable protocol in Internet of low bandwidth and high error rate with a retransmission mechanism to overcome the data loss [1]. A most related remote control mechanisms use TCP because of its reliability. But the reliability is obtained at the cost of delay transmission. In the case of thin-client robot, even if reliability is most important issues, it is effective to discard some extensively delayed data or interpolate than retransmit it. Though UDP has no flow control and retransmission mechanism [1], it has

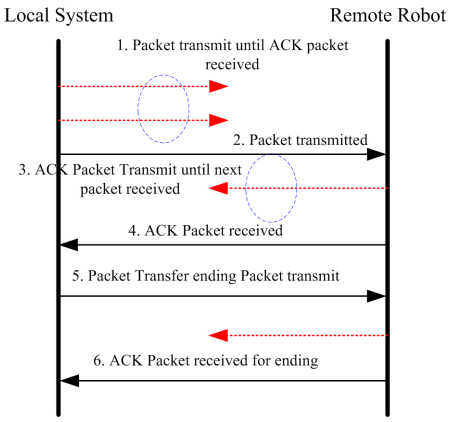


Fig. 7. Command packet retransmission mechanism

some advantages such as efficiency and bandwidth sensitivity [14]. Many Internet-based multimedia applications use UDP. RSP corresponds to upper layers than transport layer in OSI 7 layer architecture. Although RSP uses the UDP as lower layer than transport, it supports selective retransmission mechanism. Since the packet loss occasionally causes serious problem, RSP provides the ACK mechanism as shown in TABLE II (i.e, *Command, Memory Command, Memory Even*). If some device commands are defined as *Command,* then they can be retransmitted.

As shown in Fig. 6, RSP has a very simple packet structure composed of three parts: The *RSP header, Device Map* and *Payload*. RSP header represents the meta-data of the protocol. The RSP header has nine elements which are described in Table II. The retransmission mechanism is shown in Fig. 7. Local system retransmits command packets or ending packets until it receives an ACK packet such as the first sequence in Fig. 7.

The bit location and size of the *Device Map* are determined based on the devices selected by the tool as an example of Fig. 8. Each of the bits represents the device selected by a user. For example, the first of the device bits is the left wheel element of a remote robot. If user selects sixteen devices, the device bits become 16 bits. In the device bit, "1" means that the target device must be activated and its corresponding

TABLE II
RSP HEADER ELEMENTS

Element Name	Description	Size (byte)
Version	Protocol Version	1
Model ID	ID of Robot	2
Encryption	Encryption or not	1
Type	Packet data type	1
Sequence Number	Sequence number of packet starting 0	2
Time stamp	Time of packet generation	4
Session ID	Session ID for current connection	6
Last Received Sequence Number	Last received sequence number of packet	2
Clear Buffer/Number of free buffers	Bit for clear the data buffer in robot, increate by 1, robot transmit the status of buffer	1

payload contains data. If device bit has "0" value, the payload can be omitted. For example, in Fig. 8, the *ID number of IR fire command* can be omitted because the forth in device bits is "0".

Device Bits	1	1	1	0	0	0/1	0/1	0
Payload	Left wheel data							
	Right wheel data							
	Turret data							
	ID number of IR fire command							
	ACK number of IR receiver event							

Fig. 8. Down Link Packet example of Device map and Payload

IV. Experimental Results

A. Robot description

The robot, named CUBO, provide the Internet-based services such as "Music album", "Internet Radio", "News and Weather", etc. The specifications of the robot are shown in Fig. 9. It has a small body of size *20cm x 21.8cm x 30.9cm*. Its weight is about *2kg*. It has a 2-wheeled drive and two 1-DOF arms with one omni styled caster. It has an embedded processor, a wireless LAN (ASUS 54Mbps), a head with one CMOS color camera, *3* inch LCD monitor, a speaker, a microphone, a service selection button, a message button, six kinds of sensor, and a tilt unit. It has a wireless LAN system so that user can access the robot via Internet.

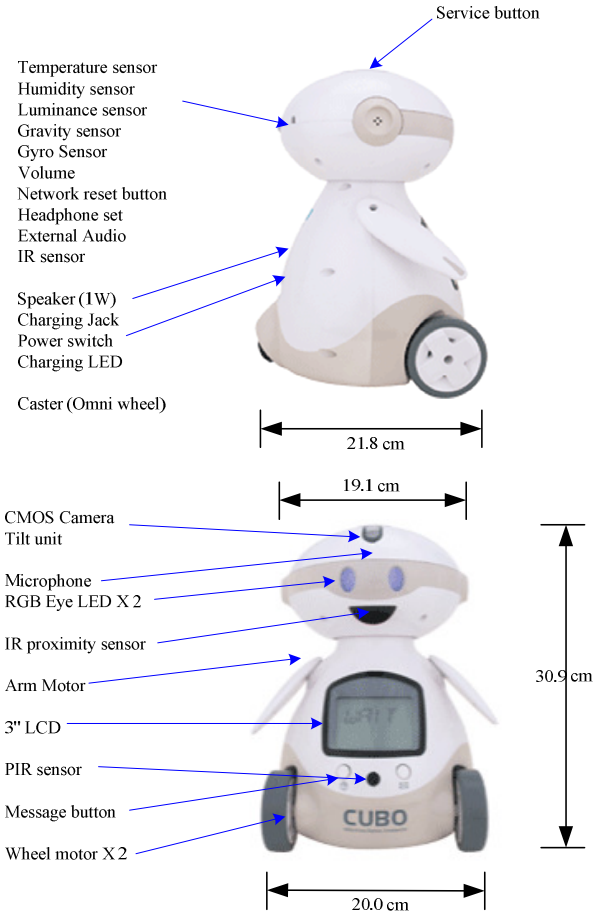


Fig. 9. Hardware description of "Cubo"

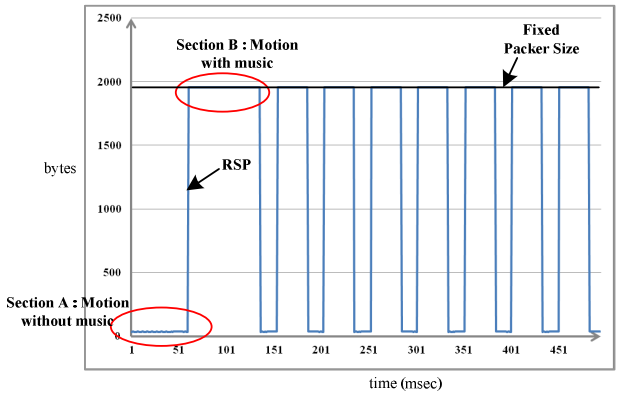
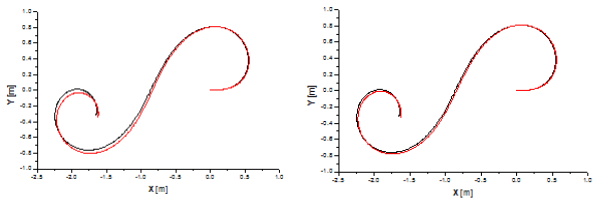


Fig. 10. Experiments of variable packet size

B. Experimental Results

To show the effectiveness of the proposed architecture, we located the robot and the local system at real Internet environment. The packet transmission rate was set as 20*msec*. The remote robot returned global position values and the local system controlled effectors or transferred multimedia data.

We performed two kinds of experiments. Firstly, to show variable packet size, simple motions composed of lip and arm movement and music were transferred to the remote robot in *2* minutes. The motions were transferred with or without music according to scenario. During the playing of music, the motion was always transferred. As shown in the Fig. 10, the packet size in section A was only 34 bytes, and in section B 1992 bytes were transferred. As shown in this experiment, RSP decreased the packet size in comparison with fixed packet size. The amount of decreasing rate was about 38.65%. Secondly, the path following was experimented in order to show the effectiveness of RSP. The experimental results are shown in Fig. 11. The local system controlled the remote robot in order to follow the path based on the transmitted position data. As shown in Fig. 11, the path error of RSP was smaller than a result by the fixed packet size. As shown in the experimental result, RSP is more robust than TCP using the fixed packet size in the transmission delay.



(a) Fixed packet size (TCP) (b) Flexible Packet size (RSP based -on UDP)

Fig. 11. Experimental results of path following

V. CONCLUSION

A simulacrum-based architecture and protocol has been proposed for thin-client robots in Internet environment. The architecture is insensitive to transmission time delay and data loss because of the variable transmission packet size and selective retransmission mechanism. In the unpredictable Internet environment, the transmission time delay and data loss can be compensated. By the RSP, the real-time control will be improved. The simulacrum-based architecture can be applied to various Internet-based service robots and the price of robot can be lowered dramatically by the proposed architecture.

REFERENCES

[1] P. Li, W. Lu, and Z. Sun, "Transport layer protocol reconfiguration for network-based robot control system", *IEEE Networking, Sensing and Control, ICNSC2005 - Proceedings 2005*, pp. 1049-1053, 2005.

[2] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop tele-operation via the World Wide Web", *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 654-659, 1995.

[3] Goradia, A., Xi, N., and Elhajj, I.H., "Internet based robots: Applications, impacts, challenges and future directions", *IEEE Workshop on Advanced Robotics and its Social Impacts,* pp. 73-78, 2005.

[4] J. Pu, M.-Q.-H. Meng, P.-R. Liu, W.-L.-C. Shek, and P.-X. Liu, "A platform and related issues for efficient interactive Internet based robotic teleoperation", *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, pp. 5571-5576, 2004.

[5] R. Oboe and P. Fiorini, "Issues on Internet-based teleoperation". *In Syroco 97*, pp. 611-617, Nantes, France, 1997

[6] K.-H. Han, S. Kim, Y.-J. Kim, S.-E. Lee and J.-H. Kim, "Implementation of Internet-based personal robot with Internet control architecture", *IEEE International Conference on Robotics and Automation*, pp. 217-222, 2001.

[7] H. Hu, L. Yu, P. Tsui, and Q. Zhou, "Internet-based robotic systems for teleoperation", *Assembly Automat. J.*, vol. 21, no. 2, pp. 143-151, 2001.

[8] A. Lasso and T. Urbancsek, "Communication architectures for web-based telerobotic systems", *Proceedings of the IEEE Mediterranean Conference on Control and Automation*, Dubrovnik, Croatia, 2001.

[9] Z. Cen, A. Goradia, M. Mutka, N. Xi, W.-k. Fung and Y. Liu, "Improving the Operation Efficiency of Supermedia Enhanced Internet Based Teleoperation via an Overlay Network", *IEEE International Conference on Robotics and Automation*, pp. 679-684, 2005.

[10] S. Munir and W.J. Book, "Control techniques and programming issues for time delayed Internet based teleoperation", *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, pp. 205-214, 2003.

[11] F.-L. Lian, J. Moyne,D. Tilbury, "Network design consideration for distributed control system**,** Control Systems Technology", *IEEE Transactions*, Volume 10, Issue 2, pp. 297-307, March 2002.

[12] I. Elhajj and Xi Ning, **"**Supermedia-enhanced Intemet-based telerobotics" *Proceedings of the IEEE*, Volume 91, Issue 3, pp.396-42I, March 2003.

[13] P.-X. Liu, M.-Q.-H. Meng and S.-X. Yang, "Data Communications for Internet Robots", *Autonomous Robots*, pp. 213–223, 2003.

[14] P.-X. Liu, M.-Q.-H. Meng, X.Ye and G. Jason, **"**An UDP-based protocol for Intemet robots" *Proceedings of the World Congress on Intelligent Control and Autumorion (WCICA),* pp. 59-65, 2002.

[15] M.Kelly, "Encyclopedia of Aesthetics", V.1, Oxford University Press, 1998.