Remote Data Transmission Middleware for Telerobotics

Jeong Seok Kang, Young Gyu Kim,*
Electronic and Telecommunication Engineering
Kangwon National University
Chuncheon, Republic of Korea
{sleeper82,playmaker07}@control.kangwon.ac.kr

Hong Seong Park*
Electronic and Telecommunication Engineering
Kangwon National University
Chuncheon, Republic of Korea
hspark@kangwon.ac.kr

Abstract—Telerobotics is the area of robotics concerned with the control of robots from a distance, using wire and wireless communications. The telerobotic system is built exploiting distributed computing system and Internet technologies. Distributed computing technologies provide several advantages to telerobotic system. In the distributed computing system, data channel management and data transmission are important because of unstable networks and firewall environments. This paper proposes remote data transmission middleware for telerobotics. The middleware provides easily transmission of data in a distance environment and automatically management of data channel. We implement the middleware and conduct empirical performance measurements.

Keywords—Data transmission; Channel management; Telerobotics; Communication middleware;

I. INTRODUCTION

Telerobotics is the area of robotics concerned with the control of robots from a distance, using wire and wireless communications [1-3]. While the physical separation may be very small, telerobotic systems are often at least conceptually split into two sites: the local site with the human operator and all elements necessary to support the system’s connection with the user, and remote site, which contains the robot and supporting sensor and control elements. To support this functionality, the telerobotic system is built exploiting distributed computing system and Internet technologies.

Distributed computing technologies provide several advantages to telerobotic system, such as dynamic and multiuser access to remote resources and arbitrary user locations. In the distributed computing system, data channel management and data transmission are important because of unstable networks and firewall environments.

Some researchers focus on the teleoperated robot systems that are accessible by the public via web browsers based on the hypertext transfer protocol (HTTP) [4-6]. The studies have the following properties. First, most telerobots are continuously accessible (online). Second, since hundreds of millions of people now have access to the Internet, mechanisms are needed to handle client authentication and contention, and etc. Some studies for data transmission in the distributed computing system were performed [7-10]. Those researches have not considered channel management in unstable networks and firewall environments.

This paper proposes remote data transmission middleware for telerobotics. The proposed middleware provides easily transmission of data in a distance environment and automatically management of data channel in unstable networks and firewall environments. We create two types of channel which are active and passive channel based on TCP/UDP, and periodically monitor those channels. And it marshals/unmarshals primitive and complex types of data for transmission.

The rest of this paper is organized as follows. Section 2 describes the architecture of remote data transmission middleware. Section 3 describes how the middleware supports channel management and data transmission. Section 4, we evaluate our middleware by empirical performance measurements. Finally, we conclude this paper in Section 5.

II. REMOTE DATA TRANSMISSION MIDDLEWARE ARCHITECTURE

The main aims of the proposed middleware are to manage data channel based on TCP and UDP and to transmit data easily. To the functions, the proposed middleware consists of remote variable manager, remote variable requester, remote variable request worker, remote variable data (un)mashaller, and network channel manager. Fig. 1 shows the proposed middleware architecture.

Remote variable manager is in charge of registration and deregistration of remote variable, and managing information of remote variable (e.g. name, remote variable type, data type, and etc.). Remote variable requester is in charge of read/write request and publish/subscribe request by remote variable type. Remote variable request worker is responsible for read/write response and subscribe response by remote variable type. Also remote variable requester and remote variable request worker can update data of local variable. Remote variable data (un)mashaller is responsible for marshaling and unmarshaling of remote variable’s data. The module marshals and unmarshals primitive and complex data types of remote variables.

This work was supported by the Industrial Foundation Technology Development Program of MOTIE/KEIT, Rep. of Korea [Development of a Tele-Service engine and Tele-Robot Systems with Multi-Lateral Feedback].

U.S. Government work not protected by U.S. copyright
variable. Network channel manager is in charge of managing active and passive data channels. The module creates and deletes channels for communication, and periodically monitors the channels.

In non-firewall environment, a data sender requests active channel creation to a data receiver. In other environment, a data receiver requests passive channel creation to a data sender in order to receive data.

III. MAIN OPERATIONS

A. Active and passive channel connection

The proposed middleware creates two types of channel which are active and passive channel in accordance with environments.

Fig. 3 shows the procedure to create a passive channel. In the passive channel, it is 2-way channel connection. The network channel manager in remote data receiver calls createChannel() of the AP channel connector. Then the connector sends CONN_REQ message to the AP channel acceptor in remote data sender. The acceptor create new passive channel object and sends CONN_RES message with information of the created channel to the AP channel connector in remote data receiver. The connector directly sends CONN_PASSIVE_REQ message to the passive channel in the remote data sender. Finally the remote data sender sends CONN_PASSIVE_RES through own passive channel.

B. Active and passive channel monitoring

In order to monitor that channels based on UDP are connected, the AP channel monitor periodically sends UDP_HELLO_REQ message with the channel information to the UDP acceptor in remote data receiver. The UDP acceptor decides that the channel is already created. If the channel is created, the channel state is updated. Then the module sends UDP_HELLO_RES message to the UDP connector. If the connector do not receive the response message, The channel object is deleted. The AP channel monitor in the remote data receiver periodically monitors the own channel states, not sending UDP_HELLO_REQ message. Fig. 4 shows the procedure to monitor UDP based channels.
C. Remote data transmission

Remote variable accessor (RVA) and sharing remote variable (SRV) are request/response model based remote data transmission mechanism. A remote variable accessor can write and read a sharing remote variable in distance.

**Writing remote variable** is to write remote variable access’s data to sharing remote variable in distance. When RVA side module calls `writeRV()`, the remote variable requester marshals the RVA’s data using the remote variable marshaller. Then, the remote variable requester gets the channel address through the remote variable manager. When the remote variable requester in RVA side sends `SRV_WRITE_REQ` message, the remote variable request worker in SRV side receives the message. The remote variable request worker finds a related SRV. If the SRV is found, the worker unmarshals the received RVA data and updates the SRV value. Then, the worker sends `SRV_WRITE_RES` message, and notifies that the SRV data is update to user module.

**Reading remote variable** is to read a SRV’s data in distance. When RVA side module call `readRV()`, the remote variable requester gets the channel address through the remote variable manager. Then, the remote variable requester sends `SRV_READ_REQ` message. After the remote variable request worker in SRV side receives the message, the worker finds a related SRV. If the SRV is found, the worker marshals the SRV data and then notifies that the SRV data is read to user module. The worker in SRV side sends `SRV_READ_RES` message, the remote variable request worker in RVA side receives the message. The worker in RVA side unmarshals the received SRV data, and then it updates RVA data. It finally notifies that RVA data is updated to the user module.

IV. IMPLEMENTATION AND EVALUATION

To evaluate the proposed middleware, we implemented the middleware in C++, and constructed the testbed as shown Fig. 6. The middleware ran on two nodes (running Windows operating system, with a 3.30GHz CPU and 4GB RAM) and those nodes are linked through 100 Mbps Ethernet in different LAN.

Fig. 6. Testbed configuration.

Based on this configuration, we measure the user round-trip data transmission time (ms) by the size of data (bytes). $N_1$ node sends data to $N_2$ node, and then $N_2$ node re-sends the data to $N_1$ node.

Fig. 7 shows the average of data transmission time in ms by the size of data in bytes. We compared our middleware with ROS [9] and OpenDDS [11] by UDP and TCP. In TCP case, we disable Nagle’ algorithm called `TCP_NODELAY`. In the case of UDP, ROS does not send the large size of data about 10,000 bytes. In addition, our middleware sends data faster than ROS and OpenDDS. Likewise, in the case of...
TCP_NODELAY, our middleware sends data faster than those middlewares.

![Fig. 7. Average of data transmission time (ms) by the size of data (bytes).](image)

V. CONCLUSION AND FUTURE WORK

In the telerobotic system which is distributed computing system, data channel management and data transmission are important because of unstable networks and firewall environments. This paper proposes remote data transmission middleware for telerobotics. The proposed middleware provides easily transmission of data in a distance environment and automatically management of data channel in those environments.

The proposed middleware is evaluated via conducting empirical performance measurements. The evaluation results show the benefits of the middleware.

In the future we plan to consider a content-based publish/subscribe model middleware.

REFERENCES


