An Internet Robot Tele-operating System

WANG Quanyu, CAO Haifang, LIU Xin
School of Computer Science & Technology
Beijing Institute of Technology
Beijing China
wangquanyu@bit.edu.cn, slowlycome180@hotmail.com, liuxinyhoo@yahoo.com.cn

***Abstract*— A server-decentralized internet model based on Jabber for robot tele-operation with P2P stream media transfer supplement based on JXTA is proposed. The system is composed of four components: operators, robots, transfer servers and data-keeper. The robot-controlling data/robot state data are packed with XML stanzas and delivered to the resolved robot/operator through XML streams. The locale audio/video media streams of the robot are sent directly through P2P pipes to the operators. In order to test its availability and performance, the model is implemented and instanced as remote control systems for Virtual Puma560 robot and the Hexapod Monster robot. Experiments of the systems and network tests are carried out to evaluate the instances; the results show that the systems are suitable for many kinds of robot tele-operation scenarios despite the tough network environment.**

***Keywords*—Jabber, Tele-operation, Robot simulation, P2P**

I. INTRODUCTION

The Internet is becoming more and more popular in recent years and it provides us a cheap and convenient communication channel for robot tele-operation. Thus the WWW-based RCS (remote-control system) is becoming an interesting and promising field of investigation in robotics, VR and visualization [1].

The essence of the Internet is sharing data, information and other resources. This reminds us we can not only develop our robot application via the Internet, but also share the robots and relevant resources in order to make the robots serve the people. Nowadays there are more than one million robots in use all over the world [2]. Applications of the current investigations in areas such as space and underwater robotics, remote manufacturing, remote surgery, traffic control, house cleaning, remote education and entertainment are of great interests and importance.

Although robots may have some extent of intelligence and can perform some jobs automatically, human intervention and human intelligence are still indispensable, especially in those complex situations that could not be handled by robots alone. Hence, the robots RCS is the way to provide the operators a friendly interactive environment and combine the intelligence of human beings with the execution of robots. Generally there are two popular ways to construct the Internet RCS:

1) Basic Internet protocol Based RCS [2]: This kind of Internet RCS is built up on the TCP/IP (transmission control protocol/internet protocol) [3]. As a basic technique on the worldwide web (www), the TCP/IP has arisen the possibility of using the internet for remote control. Because the protocol is so basic that too much work has to be done to resolve many internet problems, such as firewall/NAT (network address translations) traversal, connection safety, durability etc.

2) Hypertext Based RCS [5]: In this kind of RCS, usually the user interface is implemented as an HTML page with a Java applet for remote control and communication and a virtual environment written in VRML for visualization [4]. The operator (browser) has the ability to connect to a dedicated WWW server and download the user interface. An obvious shortcoming of this kind of RCS is that the only server restricts its scalability and expansibility.

In order to take full advantage of the Internet resources and to provide a scalable service to the public users, a multi-server based RCS architecture is proposed in the paper. Jabber/XMPP is chosen as the architecture's substrate with a P2P (peer to peer) stream media transfer service based on JXTA as its supplement.

Jabber is an open Instant Messaging protocol that relies on XMPP and XMPP is a generic and extensible messaging protocol based on XML [6] [7]. It is implemented here to perform a lot of the system's routine work: managing operator/robot accounts, defining the different ways that operator query remote entities for service discovery, maintaining the authentication and security, handling all communication messages sent among operators/robots and machine-based services, firewall/NAT traversal, etc.

JXTA technology is a set of open protocols that enable any connected device on the network, ranging from cell phones and wireless PDAs to PCs and servers, to communicate and collaborate in a P2P manner [8]. JXTA peers are created with a virtual network where any peer can interact with other peers and resources directly in case of multiple requests for one audio or video stream resource from different operators or if the stream resource is no longer needed to be recorded.

This paper presents the design, implementation, and evaluation of the server-distributed architecture RCS.

II. ARCHITECTURE DESIGN

A. Design Criteria

In order to make this architecture fully applicable in practice, some potential issues must be targeted:

1) Availability: Once an operator/robot goes online or offline, his state quickly becomes available or invalid.

2) Performance: Remote-control has frequent updates and they must be propagated under certain time constraints even though the operators/robots may have limited bandwidth.

3) Scalability: System is easy to deploy new servers and clients as needed to meet increased demand without disrupting existing remote-control for connected users and it can provide a single, unified, seamless environment that can support tens of thousands of concurrent operators/robots.

4) Fault Tolerant: When a server goes down, system events and remote-control connections can be redistributed to other servers.

5) Security: Security plays a critical role in our remote-control systems, and a compromise in security can result in downtime and increased operations costs.

B. Possible Application Scenarios

The architecture should be designed as a generic structure providing communication and other services and can be easily configured as the following kinds of RCS:

1) One-To-One RCS:

One operator controls one robot. The operator can make a connection to the robot. Once the connection has been established, the operator can control the robot and request for the state information of the robot. Meanwhile, the connection request from other operators will be refused by the robot.

2) One-To-Many RCS:

One operator can control more than one robot. The operator can control the robots in two different ways. In the first way the operator makes separate self-governed remote-control connections to each robot, and sends specific orders to each robot. In the other way, the robots to be controlled are considered as a robot-group, the operator sends the order to the robot-group, so there is only one connection from the operator to the robot-group, and then the orders are resolved by the robots-group into many sub-orders for each robot.

3) Many-To-One RCS:

Many operators control one robot. The operators form as an operator-group, and make a connection to the controlled robot. The operators in the operator-group can send orders to the robot alternatively, and they can also consult the orders to be sent to the robot before sending.

4) Many-To-Many RCS:

Many operators control many robots. The operators can combine the ways of operations discussed in the previous three RCS to control the robots which can be treated as a robot-group or separate robots. The operators can communicate with each other and the robots can also be configured with rules of cooperation.

C. The Network Structure

With consideration of the above issues, we designed a simple but complete network structure.

As showed below in Figure 1, the system architecture is made up of four components connected by the Internet: Operators, Robots, Transfer Servers and Data Keepers. One of the advantages of this architecture is that all of the components are not amount limited.

1) Operators/robots

Operators/Robots in this architecture are end-users who connect to the transfer server and perform as clients of it. Whenever an operator/robot is on line, registration information should be filled out and the server knows his presence.

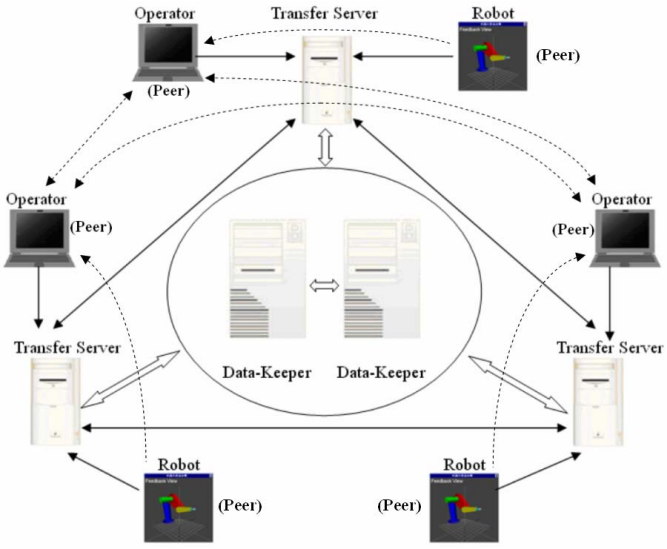


Figure 1. The network structure

Operator/robot identification in the system is the all-important problem to resolve and our solution depends on the Addressing mechanism of XMPP.

Operator/Robot@Bit.com/Remote Control

node domain resource

Figure 2. Anatomy of a typical operator/robot ID

Just as Figure. 2 shows, the ID of an operator/robot is composed of three parts:

a) Node: To the left of the "@" symbol lay the node part of the ID, naming the operator/robot to whom this OID/RID belongs. This name must be unique within the server named by the domain part.

b) Domain: After the "@" comes the domain, an identifier that names the host (Transfer Server) of this OID/RID.

c) Resource: The optional resource part, which must follow a "/" character, allows an operator/robot to further specify its location somehow. This has the effect of allowing the operator/robot to be in several remote-control connections at the same time.

2) Transfer Server

Relying on a naturally distributed structure, transfer servers are decentralized and any user can run their own Transfer Server, enabling individuals and organizations to take control of their own experience.

Each Transfer Server is identified by a network address and the architecture can be regarded as a system consists of a network of servers that inter-communicate. This pattern is familiar from messaging protocols, such as SMTP (Simple Mail Transfer Protocol), that make use of network addressing standards. Communications between any two transfer servers are OPTIONAL.

From the network's point of view, a Transfer Server do not actually host the remote-control, it's simply another user-level node of the system. Rather, it listens for and responds to the requests from operators/robots (and maybe also handles other sorts of messages, such as personal chats). The most important of these is the "request connection" request, sent by an operator that wishes to start a new remote-control connection to a robot. When it receives this request, a server makes sure that where the robot is by the domain of his RID. If they are in the same domain, the connection data will be transferred just through the Transfer Server in this specific domain. If their OID/RIDs have different domain, the operator/robot don't have to do any more than the former one, since everything will be done by Server-to-Server level, data transferring from one operator/robot to the other one when they are in different domain one actually handled by the servers of their domain.

3) Data-Keeper

As its name suggests, the Data-Keeper's role in our system architecture involves managing all the remote-control connections and operators/robots records generated by connect activity. Whenever a remote-control ends, the Data-Keeper hears about it, taking note of who is the operator, who is the robot, what server hosted it. Thereafter, operators/robots can query the Data-Keeper to learn about that remote-control connection.

Every operator/robot is allowed to run their own Data-Keeper if they wish, and in doing so create a separate remote-control network with its own set of remote-control and operators'/robots' records.

4) Peer

The operators' computers can be treated as peers whenever they are requested to relay the audio and video stream resources. Though distributed designed, the transfer servers may still be the bottleneck when many audio or video stream resources from the locale of robots. P2P is introduced here in order to: a) avoid the overload of the transfer server; b) to find a better route than the "robot-transfer server-operator" .

III. IMPLEMENTATION

On the basis of the above system architecture it is easy for us to fulfill an operator's/robot's interface of a Robot Remote Control. Here below is how the Robot RCS is implemented as an instance of the architecture with Jabber/XMPP.

A. Jabber/XMPP Data Features

Jabber/XMPP technologies provide a lot of valuable features of which we have taken advantage, such as message archiving, presence maintenance, publishing and subscription, message queues and messages expiration, etc. Among all of them, the most valuable feature is using streaming XML to exchange structured information [6].

XML is used in Jabber to define the common basic data types: message and presence. Essentially, XML is the core enabling technology within the abstraction layer, providing a common language with which operators and robots can communicate. XML allows for painless growth and expansion of the basic data types and almost infinite customization and extensibility anywhere within the data.

So our Jabber based RCS gets away with calling itself a fully XML-based technology simply because all of its technological components are themselves XML applications.

B. Packing RCS data

RCS data is transferred on the network, which contains the necessary data that exchanged between the operators and robots.

1) XML stanzas

Three XML stanzas - the actual payload messages - are defined in the remote-control system: message, presence, and info/query. All of these XML stanzas share a set of common attributes, which are to, from, id, type, and xml:lang [8].

a) Message Stanza

The message stanza is used a push mechanism from one operator/robot to another. All message stanzas operate the to attribute, which denotes the final recipient of the message. Once a server receives a message stanza, it must look up the value of the to attribute, and route the message appropriately.

b) Presence Stanza

Presence stanza is the notification part of the basic publish-subscribe mechanism; it is used to deliver information from one entity to multiple recipients. With this stanza, when an operator/robot goes online, other operators/robots will observe it.

c) Info/Query Stanza

Info/Query (IQ) is a request/response interaction, with which an operator/robot is able to request some information from another operator/robot, for example, the contact list. The type attribute plays a central role in the IQ stanza; it is used to specify the operation of the stanza, and can be one of the following: get, set, result, or error.

In the RCS, message stanzas are frequently used when an operator controls a robot. In each cycle, the operator and the robot exchange lots of data.

2) RCS data types

In the RCS, there are mainly four kinds of data to be transferred on the internet:

a) Robot modeling data: This kind of data is used to construct robot simulation model usually created by the robot owner for the operator to observe with the 3D modeling program.

b) Robot state data: The robot transfers this kind of data to the operators (maybe just observers) for them to realize the robot current state.

c) Robot controlling data: The operator(s) transfer this kind of data to send orders to the robot to inform the robot how to move.

d) Robot locale media data: This kind of data includes video and audio streams from the locale and transferred to the operators for them to have an intuitive impression about how their control data are performed and to make compare with the 3D simulation environment. This is a special kind of data and will be discussed later.

3) Packing RCS data

An easy mechanism is designed for b) and c) by attaching arbitrary properties to packets. Each property has a string name, and a value that is a primitive (int, long, float, double, bool). A piece of such data is listed below:

```
<properties >
    <property>
        <name>Joint1</name>
        <value type="double">Joint1's Angle or Displacement</value>
    <property>
</properties>
…
```

This mechanism is also suitable for transferring some of the models built from command stream, but for the binary form models, base 64 encoding scheme is employed to encode the binary form into Base 64 and transferred over the network. The Base 64 encoding is designed to represent arbitrary sequences of octets in a form that requires case sensitivity but need not be humanly readable [9].

The video and audio streams can be very large, after sampled and coded with the algorithm of H.263 [10] and G.729 [11] stored in predefined structures respectively. Then encoded as base 64 and then transferred.

C. Stream delivering

1) Text-based data

When an operator controls a robot, a set of data will be packed with message stanzas; with the OID/RID a persistent connection is established for delivering the XML data; once the connection has been opened, instead of delivering separate XML documents the operator/robot initiates a XML stream, then the data with XML stanzas are transferred through the stream.

2) Binary data

Binary data is transferred by a reliable bytestream protocol, In-Band Bytestreams (or IBB), between two end-users over a Jabber XML stream. IBB is a generic bytestream and suitable for sending small payloads, such as files that would otherwise be too cumbersome to send as stanzas defined above (such as a text file) or impossible to send (such as a small binary image file). As the data transferred, handshaking between the two end-users and an incremental counter are necessary.

3) Media stream data

Besides the IBB protocol, which may bring too much traffic to the transfer server and slow down the transfer rate of the media stream data, a P2P supplement is employed to construct direct media stream pipe between the operator and the robot. The data can be flushed to other operators (peers) who have the same media request from the robot. The peers can either use a gateway peer or act as a gateway peer, even when some of the peers and resources are behind firewalls and NATs or on different network transports. Peers can be configured to relay the data to other peers.

IV. EXPERIMENTS AND EVALUATION

With the previous work, some experimental systems have been constructed to evaluate the availability and performance of the RCS.

A. Virtual PUMA560 Remote Operation

1) RCS Implementation

The experiment was carried out with a computer connected to the Internet via a broadband connection performing as the Transfer Server and the Data Keeper; another domestic computer connected to the Internet via an ADSL dial-up connection running as a robot and an operator. Detailed information about the computers is listed below:

Transfer Server/Data Keeper:
CPU: Celeron 2.66GHz
Memory : 256M DDR 266
Operating system : Microsoft Windows XP SP2
Connection: Broad Band
Maximum Bandwidth: 100M
IP: Static IP 218.8. 193.34
Robot/Operator:
CPU: Celeron 1.7GHz
Memory: 256M DDR 266
Operating system: Microsoft Windows XP SP2
Connection: ADSL
Maximum Bandwidth: 1M
IP: Dynamically assigned

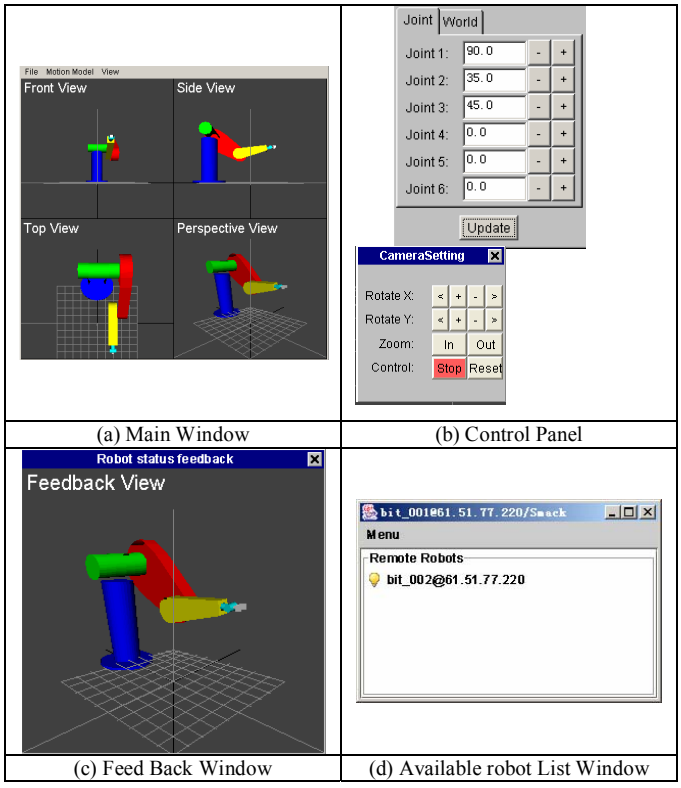
The first computer is configured as the Transfer Server and the Data Keeper with corresponding programs installed and running. Then the robot simulation program on the other computer is started and registered in the Transfer Server so that the virtual Puma robot is set in the state of *available*.

The robot to be operated in the system is a 3D fully parameterized simulation model of Puma560, which is a modeling data file stored in the client side and can be transferred to the operator if necessary. The simulation model can be changed to other robot models in order to acquire generality. When it is connected, the program pops up a window just like the Main Window in Figure. 3, showing the current position and motion of the robot.

As shown in Figure. 3, the human interface of the operator program contains four windows: the Main Window, the Control Panel, the Feedback Window and the Available robot List Window. In the Main Window, four different viewports of the simulation robot is displayed. Control Panel is designed to send control information to the robot and two modes to control the robot-Joint mode and World mode are in different tabs. Through the Control Panel, the view point of the perspective

view can also be changed. The Feed Back Window receives the robot feed-back information that indicates the angles and locations of the remote robot's joints, and then simulate it with 3D virtual reality [12] [13].

Once the operator program is in execution, after registered in the Transfer Server, it can access the system and can initiate a connection of remote-control by clicking on the virtual Puma robot in the available remote robots list. After an agreement is reached by the operator and the robot, i.e. the robot accepts the request of operation; the robot gets itself ready to be operated by the operator. The operator sends the operation commands to the robot through the control-panel. Before being sent, the commands are packed into the XML-formatted messages which can be transferred by the Transfer Server and can be translated and recognized by the virtual robot. As the robot is under control and follows the orders from the operator, it packs its current state-data into the XML-formatted message to feed back its state to the operator so that the operator program can simulate the robot's current state in the 3D visualization environment. The screen shots of the experiment are shown in Figure. 3.



(a) Main Window (b) Control Panel

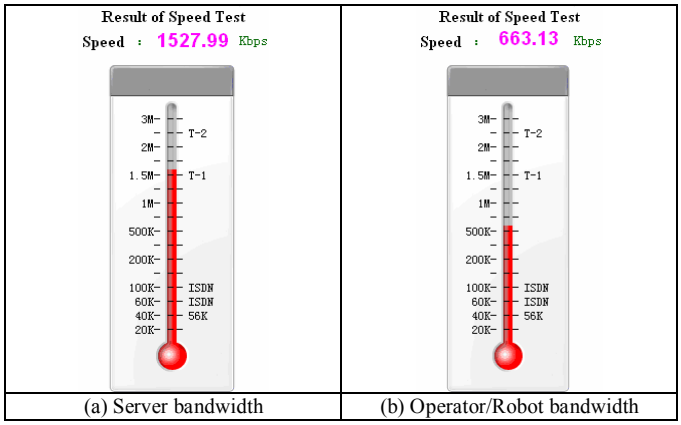
(c) Feed Back Window (d) Available robot List Window

Figure 3. Robot remote control simulation system snapshots

2) Performance

The success of the experiment has testified the availability of the RCS. Further tests are fulfilled in order to acquire the performance of the system.

Before the performance test, an Internet bandwidth test is carried out by connecting a bandwidth test website: http://www.linkwan.com/gb/broadmeter/SpeedAuto/. The results are shown in Figure. 4.



(a) Server bandwidth (b) Operator/Robot bandwidth

Figure 4. The download bandwidth of server and operator/robot

The data in TABLE I recorded the delay of the RCS and the relationship between it and the package size. The data show that the average delay increases slightly as the package size grows.

TABLE I. PERFORMANCE OF THE SYSTEM

Num of Joints	Bytes/ Package	Test Packages	Min Delay (ms)	Max Delay (ms)	Avg Delay (ms)
6	683	30	351	160	243
12	1158	31	441	170	273
18	1633	37	1302	190	419
24	2108	31	1102	200	342
30	2583	30	511	180	315
36	3058	35	631	250	320

In the Puma RCS, it takes about 200 bytes to describe the state of the 6 DOF robot arm and the control data can be managed into a pack less than 600 bytes. Theoretically the feedback from the robot is so fast that the operator can not observe any sticky display. Practically, during the operation experiment some sticky displays were found and some even very obvious due to the uncertainty of the Internet. It is not negligible when the system is applied.

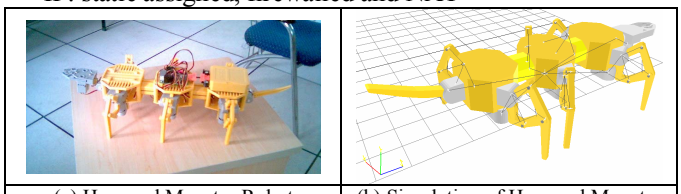
B. Hexapod Monster Operation

1) RCS Implementation

Another Hexapod Monster robot control experiment is implemented with the architecture to test the P2P stream media transfer service based on JXTA. The experiment is carried out with three computers.The server computer is the same as above.

Robot and Operator Computers:

CPU: AMD2800+
Memory: 512M DDR 266
Operating system: Microsoft Windows XP SP2
Connection: CERNET
Maximum Bandwidth: 100M
IP: static assigned, firewalled and NAT



(a) Hexapod Monster Robot (b) Simulation of Hexapod Monster

Figure 5. The photo and simulation of the Hexapod Monster robot

The Hexapod Monster is a 15 DOF robot controlled by 15 high-resolution servo motor with a serial port through which it can be connected to and controlled by a PC. In the operator's end, a simulation model of this robot is built to show the state of the remote robot. Video information is put into the window to augment the virtual environment. The photo and simulation of the Hexapod Monster robot is shown in Figure.5. The augmented reality operation interface is shown in Figure.6.

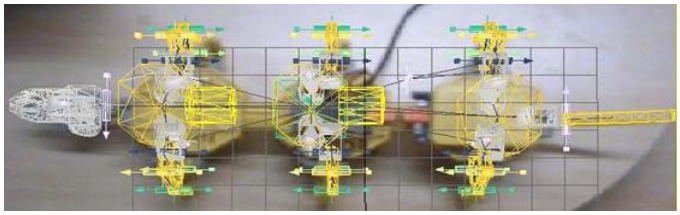


Figure 6. The Hexapod Monster in Operation

Two experiments, with and without P2P stream media transfer, were finished with the above environment.

2) Performance

The data in TABLE II show the average delay of control/state packages. Comparing with Table I, the CERNET provide a poorer quality of data transfer.

TABLE II. AVERAGE DELAY OF CONTROL/STATE PACKAGES

Control/State Packages(Bytes)	Average delay(s)
20	1.0
30	1.1
40	1.1

TABLE III shows the delay of media packages transferred by the Transfer Server and TABLE IV shows the delay of media packages transferred directly from the robot to the operator through P2P.

TABLE III. RESULTS OF AUDIO/VIDEO TRANSFER WITHOUT P2P

Sending speed(bit/s)	Reveiving speed(bit/s)	Average delay(s)
24K	11.2K	9
36K	16.0K	9.5
48K	18.4K	9.8
72K	24.0K	10.5

TABLE IV. RESULTS OF AUDIO/VIDEO TRANSFER WITH P2P

Data packages(Bytes)	Max time(ms)	Min time(ms)	Avg transfer time (ms)
6K	181	15	21
12K	188	16	24
18K	156	15	28
24K	234	15	30
30K	235	18	38

Data in tables III and IV show that by means of P2P, performance of the media transfer can be greatly improved in some specified cases. In table IV, the column "Avg transfer time" means the time from the first data packet sent to the last data packet received, including not only the "Average delay".

V. CONCLUTION

An RCS architecture based on a senior communication protocol Jabber/XMPP with P2P media transfer as its supplement is designed, implemented, instanced and evaluated. The system is firewall/NAT traversal and with it different application can be run concurrently. Experiments show that the RCS paradigm is available for many applications. There are some obvious advantages in this RCS:

1) User management is handled at the network level.

2) A cross-platform can be easily generated since Jabber can provide broad, cross-platform environment.

3) A lot of features are brought with the protocol-XMPP, for example, security and authentication, communication with foreign net, etc.

4) The architecture makes the RCS much more flexible: it provides ways to construct not only One-To-One, but also One-To-Many, Many-To-One or Many-To-Many RCS.

However, there is still a lot of work left undone and is now undergoing, for example, multi-server experiment, many-to-many RCS, scalability issues, etc.

REFERENCES

[1] Igor R. Belousov, JiaCheng Tan, Gordon J. Clapworthy, Teleoperation and Java3D Visualization of a Robot Manipulator over the World Wide Web

[2] Hartmut Ewald, Client-Server and Gateway-Systems for Remote Control, IMTC 2003 Instmentation and Mcasunmcnt Technology Conference Vail, CO, USA, 20-22 May 2003

[3] Peter X. Liu, Max Q.-H. Meng, Polley R. Liu, and Simon X. Yang, An End-to-End Transmission Architecture for the Remote Control of Robots Over IP Networks, IEEE/ASME TRANSACTIONS ON MECHATRONICS, VOL. 10, NO. 5, OCTOBER 2005

[4] Draien BrSEiC, Tool for Remote Control of Mobile Robots with Visualization by Means of Virtual Reality, 7th International Conference on Telecommunications - ConTEL 2003

[5] Xiaoli Yang, Dorina C. Petriu, Thorn E. Whalen, Emil M. Petriu, A Web-Based 3D Virtual Robot Remote Control System, CCECE 2004-CCGEI 2004, Niagara Falls, May/mai 2004.

[6] Jabber Protocols. Available at: http://www.jabber.org/protocol/

[7] Mikko Laukkanen. Extensible Messaging and Presence Protocol (XMPP). University of Helsinki Department of Computer Science.

[8] JXTA Protocol Specifications. Available at: https://jxta-spec.dev.java.net/

[9] S.Josefsson, Ed., The Base16, Base32, and Base64 Data Encodings, RFC 3548[S], July 2003; www.ietf.org/rfc/rfc3548.txt.

[10] ITU-T Recommendation H.263. Video Coding for Low Bit Rate Communication. 1998

[11] ITU-T Recommendation G.729. Coding of Speech At 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP). Geneva, 1996-03.

[12] Chen Yimin, Zhang Tao, Wang Di, He Yongyi, A robot simulation, monitoring and control system based on network and Java3D, Proceedings of the 4th World Congress on Intelligent Control and Automation. Shanghai, China, 2002 vol.1, pp.139-143

[13] BRUNO CAIADO' LUIS CORREIA' JOAO BRISSON LOPES, Three-dimensional Interactive Visualization in Java3D, 1530-1834/01 2001 IEEE