Real-Time Remote Control of a Robot Manipulator using Java and Client-Server Architecture

F. M. RAIMONDI - L. S. CIANCIMINO - M. MELLUSO
Dipartimento di Ingegneria dell'Automazione e dei Sistemi (D.I.A.S.)
Università di Palermo
Viale delle Scienze, 90128 Palermo
ITALY

Abstract: - The control of complex systems through PC networks is becoming increasingly important nowadays, both in the industries and in R&D centers. In this paper, a novel architecture is described which provides 24-h-a-day access to a remote system for remote control or supervision. The system can be controlled by any PC's integrated in a TCP/IP network using a Client Server communication. The Client Server protocol is a custom light textual protocol that allows the fast update of the plant model also using a very small bandwidth communication link. The server controls the plant through two RS232 interfaces connected to the driver of the planar manipulator with two links and two non flexible joints. The clients are connected to the server through a TCP/IP network and they are granted plant control and supervision privileges through authentication. The client is made up of a console window representing the present state of the system both in textual and in graphical form. This window also contains the buttons which allow the control of the manipulator.

Key-Words: - Remote control, Planar Manipulator, Internet, Client, Server, Real-Time, RS232, Java, communication protocol.

1 Introduction

In classical telerobotics the feedback of rich data, such as live video picture, from the physical hardware to the operator site via Internet or mobile communication links, is bandwidth limited and contains uncertain delay. Examples of this approach are Mercury project [3] and Telegarden project [4] where users were able to control the position of the robot arm and to view the scene as a series of periodically updated static images. Problems with static pictures can be avoided by using video technology. Video transfer is currently the hardest task to be performed within data transmission via Internet. The transfer of fluent video demands a high bandwidth capacity therefore such approach clearly needs a high-speed network to achieve the on-line control of the robot. The data transmission time across Internet depends heavily on the transient loading of the network, making direct teleoperation unsuitable for time critical and dangerous interactions.

In this paper we present a Client Server architecture with an innovative couple of plant model and light protocol allowing a real-time control of a Remote Manipulator by a computer on a Local Area Network or Internet.

In order to minimize data transmission, data transmission time, problem of unpredictable communication speed and network bandwidth capacity, we use a mathematical model of the robot arm in its environment. We also employ the network to transfer, using the light textual protocol, only the variations of the mathematical model state which are very small in comparison with the live video and easily transferable in every network state. So users are able to control the position of the robot arm, to view the graphical representation of the robot and its environment in real-time without add-on DSP, from every position on Internet.

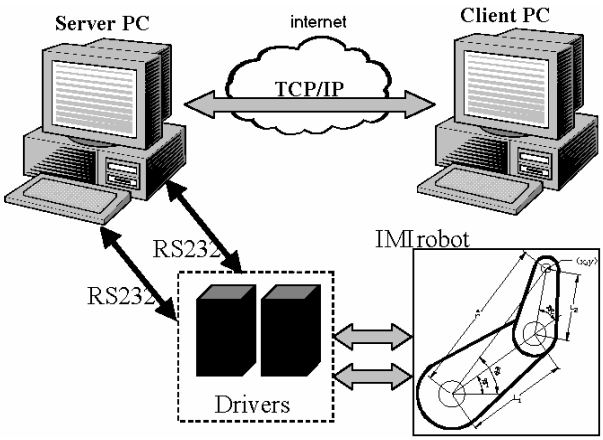

Fig.1: Remote Control System: Principle Scheme

2 Main Characteristics

The control system is made up of a standard network supporting TCP/IP protocol stack with two or more PC's and a two link planar manipulator (see fig. 1). The roles of this elements are:

1. The two link planar manipulator is the plant to be controlled. It consist of two main parts: the drivers and the arms (see fig. 2). The two arms are moved by two Brushless motors, one for every arm. The drivers implement the RS232 interfaces which interpret their input as a command or a setting instructions. The drivers give the motors the appropriate current for the movement. They also implement the local position control loop (see fig. 3) using a resolver for the feedback. All the parameters of the local Position Control (separately for the two driver) can be set via RS232 interface, by the Client part of the control system, to adjust the performance of the system. The driver also close the control loop with the server using the serial interface: the feedbak is the response of "TP5" command to the serial interface of the driver to the server.
2. The first PC is called Server because it is connected to the manipulator drivers trough two RS232 interfaces, one for every arm, and also because it runs the server part of the software of the control system. The server is the center of the control system. In fact it collects the command sent by the client in custom protocol and translates it in serial commands for the manipulator. The server checks the correctness of the manipulator state and, if correct, sends commands to the manipulator. The server grants the feedback both with respect to the plant and to the clients, closing the control loop of the system. In fact it reads the current state of the manipulator, as response of "TP5" command to the serial interface of the driver, and sends the updated state to the clients by TCP/IP connection.
3. The other PC's are called client because they run the client part of the software of the control system. They are connected to the server by a TCP/IP network and allow the user authentication through user-name and password. The client is the tool that allows the user to control the system by a graphical interface containing the graphical representation of the manipulator, the textual representation of the robot state, button setting control parameters and sending commands to the manipulator. The client also closes the control loop by TCP/IP connection. When the user tells a command the server sends a response to confirm the command and, it depends on the case, sends the update state of the manipulator.

The Client and Server programs are written in 100% pure Java. This grants the Multiplatform of the software which can run under Microsoft Windows, Unix-Like Operative Systems or in general under any operative system supporting Java (like Palm OS etc.).

Fig.2: IMI Planar Manipulator

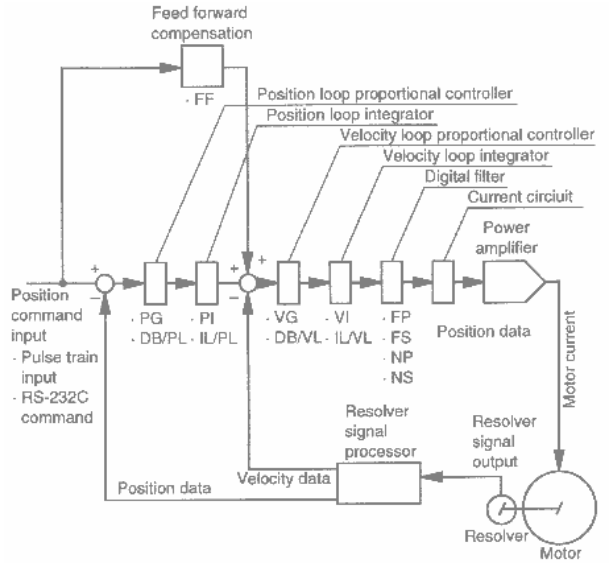

Fig.3: Position Control block diagram

3 Communication Protocols

A communication protocol is a set of key-words and grammatical rules allowing unambiguous communication between two or more entities that want to exchange information. In this application we use two protocols in order to make the client-server communication independent with respect to the server-manipulator communication. It is important to underline the difference between the two communication protocols :

The first protocol is employed between client and server (encapsulated in TCP protocols) and flows under Ethernet-like network. The second is between server and manipulator and flows in RS232 standard interfaces.

The server-manipulator (in the future S-M) protocol is a low level protocol highly dependent from the hardware drivers of the manipulator. It was developed by NSK's engineers and it is immutable.
The client-server (in the future C-S) protocol is a high level protocol developed to allow simple unambiguous communication between clients and server. This protocol is independent from S-M so we can use the same protocol to control different plants and we can change it without changing the S-M protocol. The server part executes the conversion from C-S to S-M protocols so we can update the server without changing the client part.
The information flow between client ad server in most cases consists in instructions about robot setting such as maximum acceleration or maximum velocity during movement and motion command (in the joint space).
The server examines all messages that come from client and, if the message is recognized as a valid command or setting it updates the state of the robot to check if the setting or the command is valid in the current state. In this case it will send the message to the robot using the S-M protocol.
The S-M protocol is made for serial communication. It is a textual protocol where the messages are made up by key-words representing a command or a variable and/or (it depends on the case) and information data in numeric format such as velocity, acceleration or angle.
For example, to set the maximum velocity to 1.5 r.p.s. we must send the string MV1.5 plus carriage return code (0Dh or char '\r'). The driver will give back the echo of the string plus line-feed code (0Ah or char '\n') . In case of syntax, error the command will not be recognized and the driver will give back the echo plus a question mark ('?') to notify the error. In case of correct syntax, the command will be executed and the driver will give back the echo plus a prompt (':') as notify.
Since there is one driver for every link, the C-S protocol is able to made control one link at the same time with an appropriate command. For example, to set the maximum velocity of link 1 to 1.5 r.p.s. (if the actual maximum velocity is 0.5) we will (the client) have to send the string :

```
setResolution= 1
theta1Speed++
```

the first line sets the resolution (of the incremental command like teta1Vel++) to 1.0, the second line sets the maximum velocity to 0.5+1=1.5 r.p.s. To decrease the maximum velocity simply change teta1Vel++ with teta1Vel--. When the server receives the two commands, it updates the local model, and sends the appropriate command to the driver and to all the client connected.

4 Performance

In the preceding paragraph, we overviewed the C-S and S-M protocols. In this paragraph we will study the performance and the precision of the control system. The max resolution of the S-M protocol is 0.01 degree so we can not overcome this precision.
With regard to the performance we estimate the time necessary: to send a command from client to server, to elaborate the message (by server), to update the local model, to send the command to the driver.
To simplify the compute, we can suppose null the elaboration time in the server so we can simply compute the time transmission in network and in the serial RS232 interface. As to the serial RS232 interface, since the driver needs a velocity equal to 9600 b.p.s, suppose we send 10 char we spend about 10*9/9600 = 9,3 ms.
The transmission time in network is limited only by the technology used. If we suppose a standard 100Mb/s to send 10 char we spend about 10*9/100000000= 0,0009 ms and if we have a dial-up connection with V.90 standard modem we can suppose 50kb.p.s so we spend about 10*9/50000= 1,8 ms. In this case, if we consider the Round Trip time, we spend about 2*(9,3+1,8)=22,2ms to send the message to the server, to send it to the manipulator and to receive the response.
With Regard to the unpredictable delays, the very small dimension of the TCP/IP packets let us think they will be minimal compared to the packets dimension. In conclusion we can say that the precision is the same of the robot arm and the response time is very little so we have a very real-time remote control system.
It is important to underline that we do not have to make hypotheses about the type of network; in other words we can chose an heterogeneous network containing wireless links (UMTS, IEEE 802.11 WI-fi, Bluetooth , Satellite etc) that can give many advantages. First, we can put the "plant" on a mobile support working with more motion freedom. We also can work on uncovered land with a Satellite communication. It is obvious that the performances are worse than in the traditional wire network but we have the advantages we have already described.
The use of Java languages to develop the Client-Server software control system permits to work in a multi-tasking mode; so the Server can serve many users at the same time.
The user connected to the server can have different roles such as a full control of the plant or a

supervision if it simply observes the state of the plant.

5 Hardware and Software Environment

It is important to underline the very exiguous requirement of software of the control system. It simply needs a standard PC that can run the Java Runtime Environment (JRE) version 1.3 or later. The server also needs two standard RS232 interfaces and the Java Comm API.

6 Start Up the control's software

In order to start the server part, firstly we must connect the two RS232 interfaces to the driver and the network cable to the LAN (or establish the communication for other types of link in use).
Then we turn on the IMI robot and start the server with the command at the prompt:

Java ManRoboServer <port>

where the port parameter is the server TCP/IP connection local port.

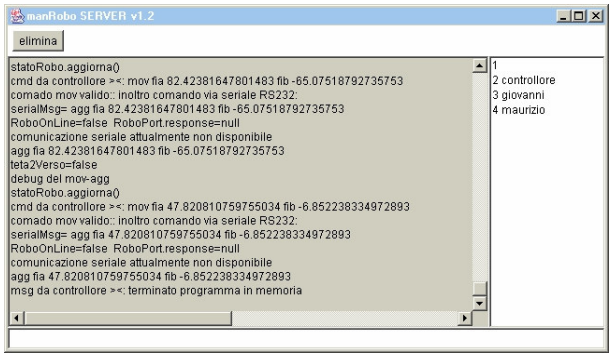

Fig.4: The Server Interface

To start the Client we employ the command at the prompt:

Java ManRoboClient <address> <port>

where the port parameter is the local port of the client TCP/IP connection and the address parameter is the server's IP address.
When the first user begins to connect to the server, the serial communication were set up and the client server application is ready for the remote control system.

7 Client Window Use

After authentication the client interface appears to the user (see fig. 5). It consist in many parts in order to simplify the plant control.

- In the upper part, there is the button menu bar that allows to send the most important commands to the server. These commands are: setting commands for important variables (movement direction and resolution, maximum speed and acceleration during the movement etc.), incremental movement commands (in the joint space) for the specified arm for an angle equal to the current resolution in the current direction.
- In the central part, there are two fields: the left one is the graphical and textual representation of the current state of the robot arm, the right one shows the information flow between the clients and the server. The graphical representation is very simple: the two arms are represented by two red lines with two red circles as a motor. The yellow circle represents the possible trajectory of the second arm supposing motionless the first. The blue box represents a hypothetical obstacle near the manipulator. The black circle represents the end-effector.
- In the bottom part, there is the text field which allows the user to send particular commands to the server or a text message to the other clients connected like in a simple chat.

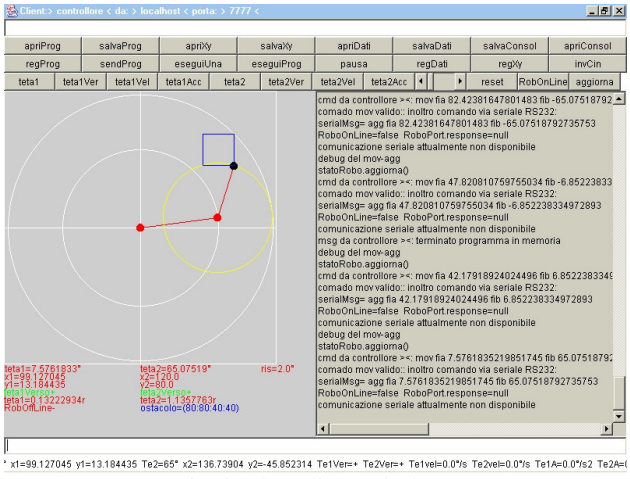

Fig.5: The Client window

When the server receives a command changing the manipulator position (and its state), it will send the manipulator state update (in broadcast mode) to the clients updating the window. In fig. 6 we can see a collection of manipulator configurations. On the left, there is the manipulator at the limit of the work space. In this position, if the client tries to move up the joint, the server checks the robot model without sending any command to the driver because the

robot can not move the arm in this direction. In this position the client can move the joint only in the direction to leave the work-space limit. On the right three canonical positions. The Home position is represented by : theta1=0 radians and theta2=0 radians. In the same figure we can see the configuration: theta1= 90 radians and theta2= 90 radians and theta1= -90 radians and theta2= -90 radians.

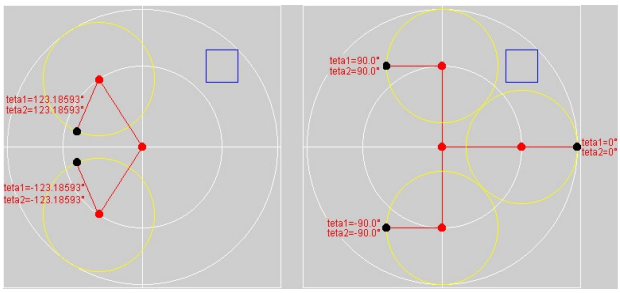

Fig.6: Work-space limit and canonical position

On fig. 7 we show the ambiguous problem obtained when we try to invert the cinematics of the robot manipulator. The (x,y) coordinates of the end-effector is the same in both cases but on the left theta2 is positive, in the right one theta2 is negative. In the first case, we do not have a collision but in the second we can have a collision with an obstacle.

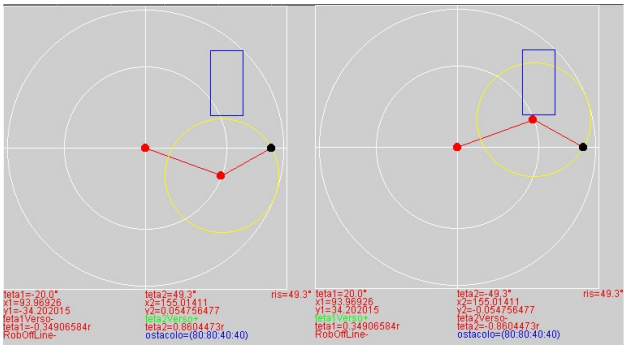
Fig.7: Ambiguous problem demonstration

8 Conclusion and Further Work

This paper present the work-in-progress of a Java Client-Server controller for a robotic manipulator. The control software accomplishes its objective of maintaining the control of the robot via the Internet (in general) connection. The client interface allows the user to move the arms with the same precision of the original system. The feedback information is provided by an update flow of the robot state. The control system is closed loop type because it is made up of three control loops between clients and server, server and drivers and, at the end, between drivers and the arm's motor.

Further developments include task-oriented manipulation (rather than step by step control) to permit the programming of the manipulator operation and, in general, to add functionality and to increase the control possibilities starting from the inversion of the mouse-graphical selected (x,y) coordinates. Other future works are: on-line estimation of the plant model parameters, web access to the collected data and to port the client window towards any Mobile Phone supporting Java 2 Micro edition (J2ME) in order to supervise and/or control the remote plant.

References:

[1] L. Sciavicco, B. Siciliano, *Robotica Industriale: modellistica e controllo di manipolatori*, McGraw-Hill libri Italia, Milano 2000.

[2] *Megatorque Motor System User's Manual*, NSK Ltd., 1995

[3] K. Goldberg; M. Maschna, Center S., et al., *Desktop Teleoperation via The WWW*, Roc. Of IEEE International Conf. On Robotics and Automation, 1995

[4] http://telegarden.aec.at

[5] J. Sanchez, S. Dormido, R. Pastor, F. Morilla, *A Java/Matlab-Based Enviroment for Remote Control System Laboratories: Illustarated Whith an Inverted Pendulum*, IEEE, 2004

[6] S. Chakrabarti, L. Wu, S. Vuong, V. C. M. Leung, *A Remote Controlled Wireless Enabled Enviroment*, University of British Columbia Vancouver, Canada, IEEE 2004

[7] R. Safaric, I. Hedrih, R. Klobucar, *Remote Controlled Robot Arm*, University of Maribor, IEEE 2003

[8] A. Malinowski, T. Konetski, B. Davis, D. Schertz, *Web-controlled Robotic Manipulator using Java and Client-Server Architecture*, Bradley University, IEEE 1999

[9] I. F. Darwin, *Java Cookbook, Solutions and Examples for Java Developers: cap. 11 Programming Serial and Parallel Ports*, O'Reilly, 2001