AN ASSESSMENT OF REAL-TIME ROBOT CONTROL OVER IP NETWORKS

G. H. Alt, R. S. Guerra & W. F. Lages
Federal University of Rio Grande do Sul, Electrical Engineering Department
Av. Osvaldo Aranha, 103, 90035-190 Porto Alegre, RS, BRAZIL
alt@inf.ufrgs.br, rsguerra@eletro.ufrgs.br, w.fetter@ieee.org

Abstract

This article presents an assessment of the performance of a real-time controller using an IP network to link the controller, the sensor and the actuator. This control system is part of anthropomorphic robot manipulator with two arms and a pan-and-tilt stereo vision head with total of the 19 joints. Each joint has an actuator, an incremental encoder, an electromagnetic brake and an inductive contact sensor. There are also two six-axis force sensors. In order to cope with all devices an distributed control system was implemented. The devices are all connected to an IP network and the real-time control loops are configured by software and run over the IP network. The real-time requirements of the control loops, which must run at 100Hz rate, are enforced by using RTAI in LXRT hard real-time mode. RTAI is also used for monitoring the time behavior of the network packets, in order to deal with the best-effort of assumption of the IP network. Indeed, an IP network assures that it will do its best effort to deliver each data packet, but there is no guarantee that the packet will be delivered. However, a real-time control loop must run at an accurate rate or the system performance or stability will be impacted. Therefore, sensor data must be feed to the controller at the same rate, or it will not be able to compute the control action. Hence, the ability of the IP network to deliver the packets with reliability and rate accuracy is very important. Of course, IP networks were not designed with such applications in mind, but it is very difficult to neglect the low-costs due to mass-production. The time behavior and the performance of a controller implemented over an IP network is characterized and compared with those obtained with a local controller.

1 Introduction

Control of computer networks consists in the use of control systems theory for the determination of parameters of communication networks, frequently using methods of closed-loop control. Typical problems in this line of research wireless is the control of transmission power in wireless networks [5] and the control of congestion in ATM networks [12].

On the other side, the main motivation of this work is in using the existing conventional computer networks, based on IP protocol, to implement a distributed control systems. In general, the decision to use a distributed control system based on the Internet protocol (IP) is made due to simplicity and low cost of installation and maintenance. This area t is relatively new for the control community in academy (there are a few number of academic papers dealing with this), but it is not so new in the industrial field [1] although using dedicated protocols. This peculiarity allows us to assume that there is a lot of space for developments in this area. The idea in this context is to use communication network to exchange control signals. Standard computer network technologies have been adapted for this context, such as daisy-chained RS-232, multi-drop RS485, IEEE-488 and Ethernet and its extensions, such as wireless Ethernet (IEEE-802.11). Additionally specialized networks have been developed for industrial applications, such as CAN [2] [3], Foundation Fieldbus [4] and Profibus [8].

Networks for control systems differ from general computer networks in some important aspects, such as generating frequent small packages with real-time requirements. An important aspect to consider is that the objective in a control network is not to transmit digital data, but control or sensor signals through bits. Hence, usual metrics in computer networks such as amount of data and transfer rate become secondary in a control network.

Another factor to consider is that in a control network the delays are not generated by the environment, in contrast to what could be supposed at a first glance. The transfer rate of the network, is such that the time of data transmission. The delays are generated basically by the queues in the network system. In general computer networks, these queues optimize the average delay time of packages. However, in a control network many times is more convenient to eliminate these queues [12].

Control networks must obey the two main criteria: limited delay time and guaranteed delivery. A very long delay to deliver a message can deteriorate the performance of the control system or even turn it unstable. The use of many protocols has been considered to satisfy these requirements, among them: CAN (CSMA/BA), token bus (IEEE-802.4), token ring (IEEE-802.5) and CSMA/CD (IEEE-802.3)[12]. The key point her is to determine which device in a control network has the right to access the transmission medium. Clearly, this method used to select one device has a great influence on the delay of messages.

The final goal of this work is to control a manipulator robot through the Internet, sending sensor data and control signals through the network. To cope with the variable delays, two methods will be used. Quality of Service (QoS) protocols to minimize the delays as much as possible and optimal estimation theory, such as Kalman filter, to estimate the analog signal corresponding to the data packages with unacceptable delays.

This paper analyzes the many time delays present in a control system over the Internet. The delays and their variation (jitter) occurring in a real control system are measured and their effects on the system analyzed.

While delay time in the Internet can not be accurately determined, due to its complex stochastic model [13], some attempts have been made to model the network delay [10]. However, it remains difficult to estimate the delay of packages being transmitted through a given route.

Certain enough an IP network will do its best effort to deliver the packages, but in fact there is not any guarantee about the time behavior of the package or even about the delivery itself. However, a control loop must operate at an accurate rate, or the system performance or worse, the system stability will be impacted. Therefore, the data of the sensors must be supplied to the controller at the same rate, or if it will not be able to compute the control action.

The ability of the network to deliver packages with sensor data to the controller and packages with control data to the robot at required rates is very important for the system performance and stability. Of course, IP networks were not designed with such applications in mind, but its low cost and availability due to the production scale attained, makes it difficult to not consider the use of IP networks.

In this paper, the time behavior of closed-loop control system implemented over an IP network is analyzed and contrasted with a purely local controller.

2 System Description

The system used to evaluate the delays in the close-loop control system is shown in figure 1. The system is composed by a manipulator robot and its controller and they are connected by an IP network. The control signals for the manipulator (Voltage to be applied to the motors) are computed by the controller and sent to the manipulator through the IP network. Similarly, the position of the manipulator joints (readings from incremental encoders) are sent to the controller using the network. Thus, there are two major delays due to the network in the closed-loop.

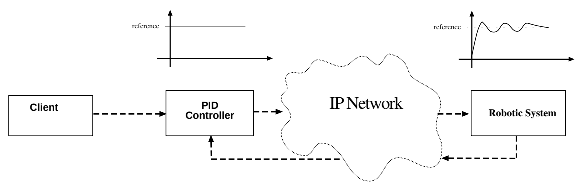

FIGURE 1: *Networked Control System*

2.1 System Hardware

The block diagram shown in figure 1 was implemented using the Janus manipulator robot, available at the Electrical Engineering, Department of the Federal University of Rio Grande do Sul, Brazil. Figure 2 shows the Janus Manipulator. It is an anthropomorphic robot with two arms and a head. Each arm has 8 joints equipped with a DC motor actuated by PWM, an incremental encoder, an electromagnetic brake and and inductive contact sensor. The head has two joints and a stereo vision system.

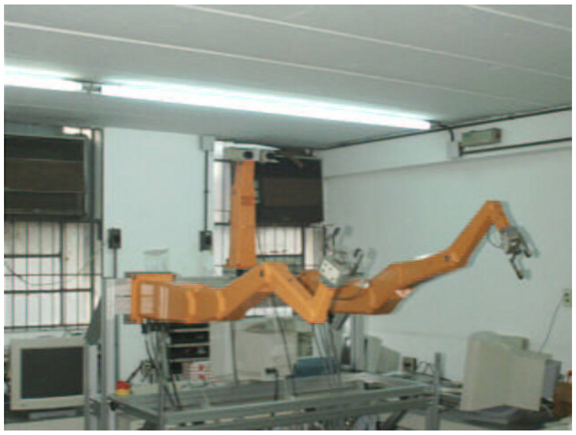

FIGURE 2: *Janus Manipulator Robot*

2.2 System Software

All the control actions of the system are performed in software. The hardware just supports the reading of the sensor data and the command of actuators. As the focus of this work is in the network introduced delays, the control law for the robot was kept as simple as possible. Thus, the robot controller uses an independent PID controller on each joint. In other words, the voltage applied to each joint motor is computed by (1):

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \qquad (1)$$

where $u(t)$ is the motor voltage, $e(t)$ is the joint position error and K_p, K_i and K_d are controller parameters to be adjusted for the desired performance.
For reasons related to control system stability, which are above the scope of this paper, the sampling rate of the joint controllers should be 100Hz. This sampling rate is enforced using RTAI in hard real-time mode. RTAI hard real-time facilities is also used to time-stamp the transmission and reception of network packages.

3 Control over IP Network

The logical network employed in this work is based on non-deterministic Ethernet at physical and link layers, IP at network layer and UDP at transport layer. The IP protocol can break up datagrams and send them through different paths over the network. The datagrams should then be reordered and reassembled at reception. Furthermore, IP does not assure data delivery, since the routers have permission to discard datagrams, without even informing sender or receiver. This is called *best effort service*, as opposed to reliable service offered by some other networks. IP trusts above layers to recover discarded frames[11] in order to implement a reliable service to applications.

This limitation does not cause problem for typical applications such as web, mail or file transfer. But as the new applications, including audio and video arise, there is an increasing demand for high transfer rates (bandwidth) and low latencies while keeping two-way communication. Hence, the question is, how to improve the level of services offered by IP and the corresponding infrastructure without adding too much complexity. There are many proposals of methods to incorporate Quality of Service on top of IP. See [9] for an up-to-date discussion of these methods.

The transport protocols most used with IP network protocol are, by far, Transmission Control Protocol (TCP) [7] and User Datagram Protocol (UDP) [6]. TCP is a full-featured transport layer protocol, with all functionalities needed to ensure a reliable end-to-end data transfer, such as flow control, error detection and recovery, etc. On the other hand, UDP is a very simple protocol without error detection and recovery. Effectively it is not much more than an interface to IP protocol.

For the real-time networked control application proposed here, UDP seems to be more adequate. The simplicity of UDP allows the application interact more directly with the network system, in order to keep the required data rate. Furthermore, as the data (sensor and control signals) should be transmitted within strict time frames, the recovery of lost or corrupted protocol packages through retransmittion could have a bad effect on the system since system bandwidth would be used to transmit a package of outdated data. Also, sensor and control signals tend to be smooth signals. Therefore, it is more effective to estimate value of the sensor or control signal correspondent to the the lost packages then to retransmit them. Figure 3 shows the stack of protocols proposed for real-time robot control.

The application programs, i. e. the programs that run above UDP use a client-server architecture. The robot systems is the server while the client perform the robot controller functions. Services available to client includes turn the robot on and off, set voltages applied to motors and return readings from sensors. The client application interfaces with user and runs the robot controller function using the PID control law, as detailed in section 2.2.

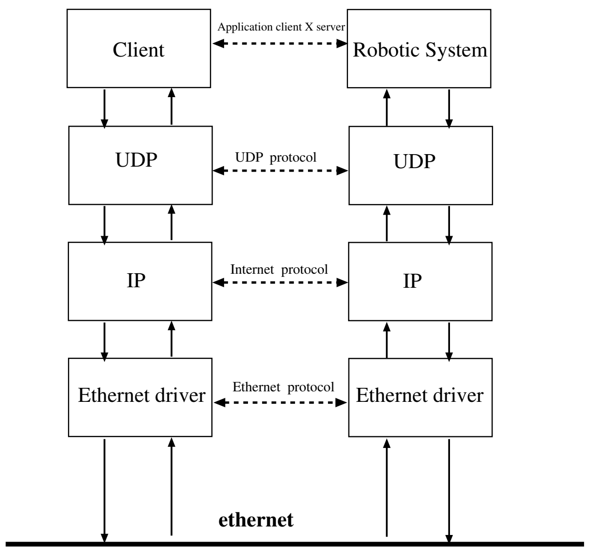

FIGURE 3: *Stack of Protocols Used for Networked Robot Control*

In a typical UDP client-server application, the client does not establish a connection with the server, since UDP is a connection less protocol. Instead, the client just sends the datagram to the server using the `sendto()` function. Similarly, server does not accept connections from client. Instead, server only calls `recvfrom()` function, that blocks until data arrive from client. The `recvfrom()`function returns the address of client, to allow server to send replies to correct client.

4 Networked Robot Control

4.1 Server

The server is implemented by a function that loops reading datagrams arriving at server port. The datagram buffering is implicit in UDP layer. Each UDP socket has a reception buffer. A call to `recvfrom()` return a datagram from the buffer in FIFO order. Hence, if many datagram arrive for socket before they can read, they are queued. Of course, the buffer has a size limit. However, for our application that is not an issue, since to maintain the sampling rate required by the control system, the server should be able to read each incoming datagram before the next one arrives.

Figure 4 shows the details of the client-server communication. There is just one server process and it uses just one socket that receives all the requests from client and sends all the answers.

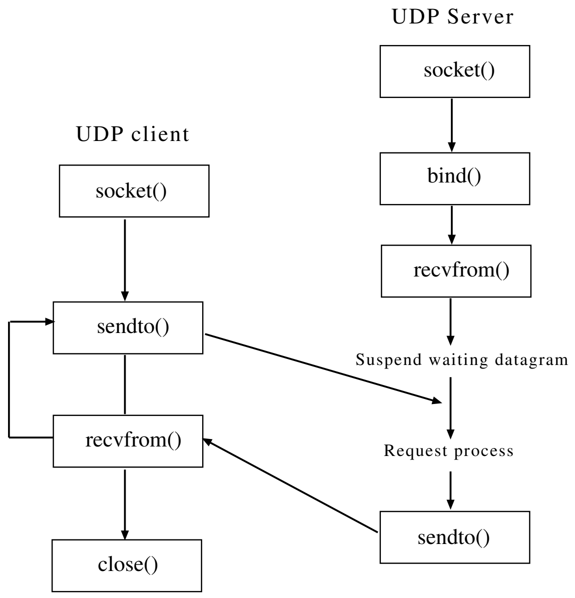

FIGURE 4: *Client-Server Networked Robot Control*

When the server program is initiated in the robot system, it creates one socket and defines the local port for the socket. This function contacts the port mapper to register the server. Then, the server waits for requests from client.

4.2 Client

The concept employed in the proposed networked control system is to perform all control operations in the client.

The client is divided in independent routines that switch on, disconnect, and send control signals to the robot and receive sensor data from robot.

The real-time robot controller, located in the client, is programmed using the threads model and allows the execution of the independent joint control. The controller thread runs in parallel with the communications thread, that sends the control signals computed by the controller thread and receive sensor data used by controller thread. These two threads communicate each other by shared variables. Both threads are hard real-time LXRT threads, running in endless loop. However, the controller thread is a periodic thread and defines the sampling rate of the system (100Hz). The controller thread synchronizes with the communicating thread through semaphores and with the server program through UDP datagrams since the `recvfrom()` function is blocking.

5 Experimental Results

Experimental result with the proposed system were obtained in four setups:

Monolithic Program PID controller and robot interface unified in a single program.

Client-Server Through Loopback Interface PID controller (client) and robot interface (server) implemented as two independent programs running in the same machine and communicating through the loopback interface.

Client-Server in a Single Machine PID controller (client) and robot interface (server) implemented as two independent programs running in the same machine and communicating through the network interface (NIC driver).

Client-Server in Different Machines PID controller (client) and robot interface (server) implemented as two independent programs running in separate machines, obviously communicating through the network.

5.1 Monolithic Program

This test case is the traditional control system implementation, without networking. Therefore it provides the bottom-line for comparing the effects of the network in the control loop. The following timing parameters were collected (see figure 5):

1. Time delay between the reading of sensor data and the writing of motor voltages, shown by the red line in figure 5. This parameter measures the time used to compute the control law give by expression (1).

2. Time delay between two consecutive writings of motor voltages, shown by the green line in figure 5. This parameter measures the effective sampling period of the controller.

3. Time delay between two consecutive readings of sensor data, shown by the blue line in figure 5. This parameter measures the effective sampling period of the controller.

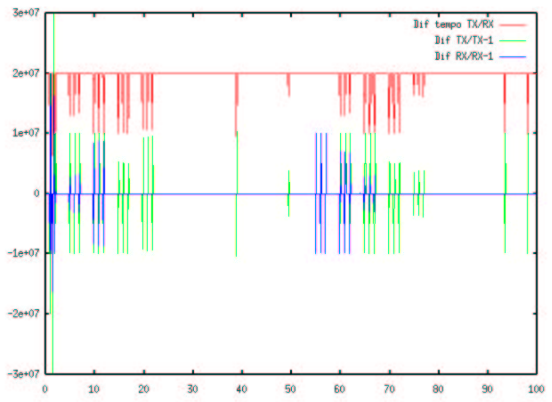

FIGURE 5: *Time Delays (ns) with Monolithic Program*

5.2 Client-Server Through Loopback Interface

This test case uses the proposed client-server architecture but the network effects are minimized by connecting the client and the server through the loopback interface. Therefore it provides what would be the faster client-server connection, enabling us to evaluate the effects of separating the controller and the robot interface while minimizing the network effects. The following timing parameters were collected (see figure 6):

1. Time delay between the reception of sensor data and the transmission of motor voltages, shown by the red line in figure 6. This parameter measures the time used to compute the control law give by expression (1). Note that this parameter is measured at the client.

2. Time delay between two consecutive transmissions of motor voltages, shown by the green line in figure 6. This parameter measures the effective sampling period of the controller. This parameter is also measured in the client

3. Time delay between the reception of the control signal and the transmission of the sensor data, shown by the blue line in figure 6. This parameter measures the time used by the robot interface to apply the voltages to motors and read sensors. This parameter is measured at the server.

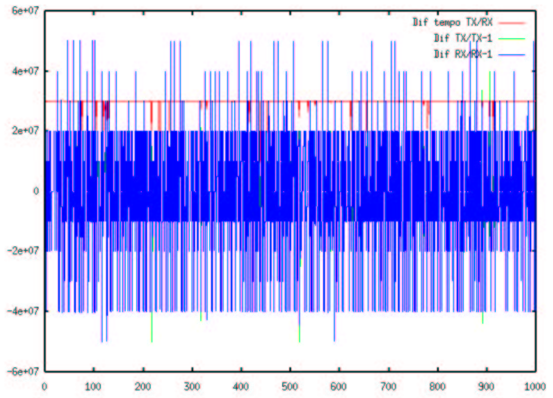

FIGURE 6: *Time Delays (ns) with Client-Server Through Loopback Interface*

5.3 Client-Server in a Single Machine

This test case uses the proposed client-server architecture but the network effects are minimized by connecting the client and the server through the NIC driver. Therefore it enables us to evaluate the effects of NIC driver while minimizing the effects of network itself. The following timing parameters were collected (see figure 7):

1. Time delay between the reception of sensor data and the transmission of motor voltages, shown by the red line in figure 7. This parameter measures the time used to compute the control law give by expression (1). Note that this parameter is measured at the client.

2. Time delay between two consecutive transmissions of motor voltages, shown by the green line in figure 7. This parameter measures the effective sampling period of the controller. This parameter is also measured in the client

3. Time delay between the reception of the control signal and the transmission of the sensor data, shown by the blue line in figure 7. This parameter measures the time used by the robot interface to apply the voltages to motors and read sensors. This parameter is measured at the server.

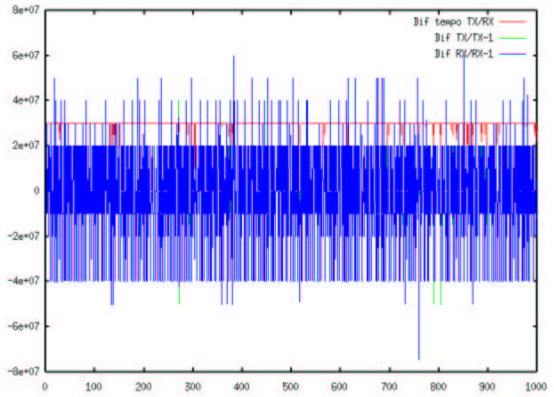

FIGURE 7: *Time Delays (ns) with Client-Server in a Single Machine*

5.4 Client-Server in Different Machines

This test case uses the proposed client-server architecture over a typical IP network, enabling us to evaluate the performance of the proposed system. The following timing parameters were collected (see figure 8):

1. Time delay between the reception of sensor data and the transmission of motor voltages, shown by the red line in figure 8. This parameter measures the time used to compute the control law give by expression (1). Note that this parameter is measured at the client.

2. Time delay between two consecutive transmissions of motor voltages, shown by the green line in figure 8. This parameter measures the effective sampling period of the controller. This parameter is also measured in the client

3. Time delay between the reception of the control signal and the transmission of the sensor data, shown by the blue line in figure 8. This parameter measures the time used by the robot interface to apply the voltages to motors and read sensors. This parameter is measured at the server.

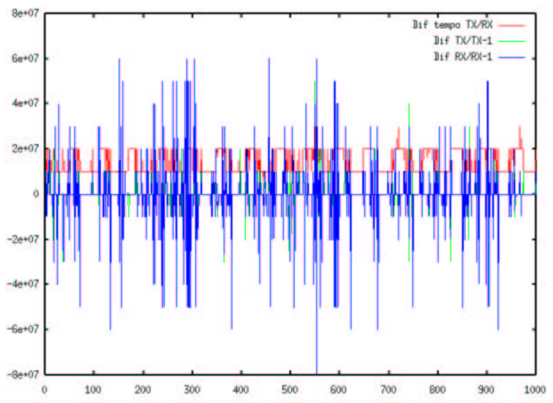

FIGURE 8: *Time Delays (ns) with Client-Server in Different Machines*

6 Conclusion

This paper presented an experimental evaluation of time delays in a networked control system using conventional IP network. RTAI hard real-time mode was used to enforce the control system sampling rate and to measure time delays and jitter.
Although the delays and jitter that were observed in experiments are acceptable, it should be noted that these figures were obtained with an unloaded network. Future work should address methods to minimize the delays introduced by the network through the use of QoS techniques. Packages loss and packages with unacceptable delay should be coped with by estimating its contents from former received packages.
Another important point to be investigated is the real impact of time delay and jitter on the control system performance, measured by traditional control systems performance criteria, such as overshoot, damping, integral of square errors and so on.

7 Acknowledgments

Authors thank the Research Support Foundation of the State of Rio Grande do Sul (FAPERGS), Brazil, for its support to this project through the Support Program for Scientific and Technological Development in Information Technology (PROADI) under Grant nº 01/0567.

References

[1] L. Bushnell. Special section - networks and control. *IEEE Control Systems Magazine*, 21(1):22–99, 2001.

[2] http://www.can.bosch.com.

[3] http://www.canopen.org.

[4] http://www.fieldbus.org.

[5] P. R. Kumar. New technological vistas for systems and control: The example of wireless networks. *IEEE Control Systems Magazine*, 21(1):24–37, 2001.

[6] J. Postel. User datagram protocol. RFC 768, ISI-USC, August 1980. available at ftp://ftp.ietf.org/rfc/rfc768.txt.

[7] J. Postel. Transmission control protocol. RFC 793, ISI-USC, September 1981. available at ftp://ftp.ietf.org/rfc/rfc793.txt.

[8] http://www.profibus.com.

[9] http://www.qosforum.com.

[10] R. Riedi, M. Course, V. Ribeiro, and R. Baraniuk. A multifractal wavelet model with application to TCP network traffic. *IEEE Trans. Inform. Theory*, pages 992–1018, 1999.

[11] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, Englewood-Cliffs, NJ, 1988.

[12] G. C. Walsh and H. Ye. Scheduling of networked control system. *IEEE Control Systems Magazine*, 21(1):57–65, 2001.

[13] Xipong Xiao and Lionel M. Ni. Internet qos: A big picture. *IEEE Network*, pages 8–18, 1999.