Networked Robot Control with Delay Compensation

Gustavo Hommerding Alt
Federal University of Rio Grande do Sul, Institute of Informatics
Av. Bento Gonçalves, 9500 - Bloco IV, 91501-970 Porto Alegre, RS, Brazil
alt@inf.ufrgs.br

Walter Fetter Lages
Federal University of Rio Grande do Sul, Electrical Engineering Department
Av. Osvaldo Aranha, 103, 90035-190 Porto Alegre, RS, Brazil
w.fetter@ieee.org

Abstract

This paper presents an assessment of the performance of a real-time controller using an IP network to connect sensors and actuators of a robot. The temporal behavior of network packets and controller performance are characterized and compared against those obtained using a local controller. A method is proposed to compensate network delays and recover the performance of the networked controller. The networked control system is basically composed of two nodes: the robot interface and the controller. The functionalities implemented on both nodes are based on RTAI LXRT hard real-time tasks. In order to test the proposed delay compensation method under various network load conditions, a traffic emulator was developed. The traffic emulator imposes delays to selected packets. These delays are generated accordingly to a statistical model of the network load resembling real network traffic. The traffic emulator is a program based on LXRT that processes network packets queued by Netfilter. A scheduler/dispatcher architecture is used to cope with the statistical nature of delays imposed to packets. Details of the application protocol are discussed and the paper concludes by presenting results comparing control system performance with and without delay compensation for some representative network load conditions.

1 Introduction

Traditional control system have been implemented by wiring sensors and actuators to a central controller. The development of computer network technologies has enabled the use of a computer network to link sensor, actuators and controllers, saving wiring, improving system reliability and flexibility.

Control networks typically handle small packages with real-time constraints [12]. Also, an important aspect is that the purpose of a control network is not the transmission of digital data, but the transmission of digitized control and sensor signals. Hence, usual metrics in computer networks such as amount of transferred data or data rate become of little importance for a control network. Control system related metrics are actually the important ones. Another point is that delays imposed to network packets are not generated by the physical medium, but by queues existing in the system, mainly in routers.

The network transfer rate, in general is such that the data transmission time is very short. Those queues optimize the average packet delay in typical computer network application [10]. However, for a control network it is usually more convenient to eliminate queues.

Standard computer network technologies have been adapted to control networks such as RS485 [1], IEEE-488, Ethernet and its extensions [11]. Furthermore, networks specializing in control and automation have been developed, such as CAN for vehicles and industrial automation, BACNet for building automation, Profibus [9] and Foundation Fieldbus for process control [2].

Considering the low cost and high availability of networks based on the IP protocol, they become a very suitable alternative to implementation of control networks. One of its attractive aspects is the possibility to close control loops over the Internet. However, the IP protocol was not developed to be
used on control networks, therefore it presents some problems. First, it is a best-effort protocol, which means that it makes no claims about the assured delivery of data. The end-to-end reliability problem is regarded to upper layer protocols. Typically the TCP protocol is used to implement a reliable end-to-end service. TCP builds such a reliable service by retransmitting lost or damaged packets. However, under some situations those retransmissions can make an eventual network congestion to become worse, which causes more packets to violate its timing requirements or be discarded by routers, forcing even more retransmissions.

Another important problem of IP protocol for real-time control systems is that this protocol does not make any claim about the timing of delivered messages. Any router can arbitrarily delay or even discard messages without communicating this fact to source or destination of message.

Control networks are, usually, local networks. Local networks using the IP protocol at network layer, typically use the Ethernet protocol at link and physical layers. The reasons are the same for the use of IP protocol: low cost and high availability. By its time, the Ethernet protocol is also not particularly adequate for control networks. The CSMA/CD medium access method forces a lot of uncertainty about the time needed to transmit a packet, mainly due to the possibility of packet collisions. Collisions can be almost eliminated by the use of a switched network, but even so, some collisions will persist whenever more than one packet to the same port arrives simultaneously at switch.

A solution that has been proposed to solve the timing problem associated with CSMA/CD is to implement some token-passing scheme at the application protocol or MAC sub-layer [6], in order to obtain an upper limit to the medium access time. Although those solutions solve the problems associated to CSMA/CD, they do not ensure that packets will not be discarded by network layer, not the end-to-end timing when an IP network is in use. Furthermore, changes to Ethernet protocol implies an raise of costs.

Obviously, in control network the above mentioned problems are tightly coupled, since an excessive delay is equivalent to a discarded message, due to real-time aspects.

This paper presents a analysis of the effects of delays introduced by an IP network on a control system. A method is proposed to compensate for effects of such delays, in order to recover the performance obtained by a conventional (i.e. without network) control system.

The compensation method is based on the fact that control networks characteristics are diverse from general computer networks. In a control network are transmitted small packets with regular interval and with real-time constraints [12]. Furthermore, in a control network the contents of packets in a sequence are highly correlated, since they corresponds to the digitalization of an analog signal. This fact, along with the possibility to have a good dynamic model for the system to be controlled form the base for the proposed delay compensation method.

2 Local Control

In this paper it is considered the networked control of the Janus manipulator (figure 1). This manipulator has two arms each with 8 degrees-of-freedom and a stereo vision head with two degrees-of-freedom. There is an actuator, an incremental encoder, an electromagnetic brake and an inductive sensor on each joint. The system also includes a computer that acts as the interface between the robot and the network. See figure 2.
2.1 Robot Dynamical Model

The dynamical model of a rigid manipulator with \( n \) degrees-of-freedom can be obtained by using the Lagrange-Euler formulation [3] and can be written as

\[
\tau = M(q)\ddot{q} + V(q, \dot{q}) + G(q)
\]  

(1)

where:
\( q \) = generalized coordinates vector;
\( \tau \) = generalized forces vector;
\( M(q) \) = inertia matrix;
\( V(q, \dot{q}) \) = centrifugal and Coriolis forces vector; \( e \)
\( G(q) \) = gravitational forces vector.

By defining \( x = [q^T \ \dot{q}^T]^T \) and \( u = \tau \), the model (1) can be written in state space as

\[
\dot{x} = \begin{bmatrix} 0 & I \\ -M^{-1}(q)G'(q) & -M^{-1}(q)V'(q, \dot{q}) \end{bmatrix} x + \begin{bmatrix} 0 \\ M^{-1}(q) \end{bmatrix} u
\]

(2)

with \( V'(q, \dot{q})\dot{q} = V(q) \) and \( G'(q)q = G(q) \). This model can be placed in the affine form:

\[
\dot{x} = f(x) + g(x)u
\]

(3)

2.2 Control Law

A computed torque control law [3] is used. This law is given by expression (4).

\[
u = M(q) [\ddot{q}_r + K_d(\dot{q}_r - \dot{q}) + K_p(q_r - q)]
+ V(q, \dot{q}) + G(q)
\]

(4)

where:
\( \ddot{q}_r \) = reference acceleration;
\( \dot{q}_r \) = reference velocity;
\( q_r \) = reference position;
\( K_p \) = proportional gain matrix; and
\( K_d \) = differential gain matrix.

Or, in simplified form

\[
u = h(x, q_r, \dot{q}_r, \ddot{q}_r)
\]

(5)

Figure 3 shows a block diagram of the system using this control law. The feedforward components compensate for the gravitational, Coriolis and joint interaction forces. The feedback components produce the extra torque needed to compensate for errors with respect to the manipulator nominal trajectory.

The manipulator response for the last two joints is shown in figure 4. Note the controller parameters could be tuned for a better performance, but that is not the focus of this work. The response shown in this figure reflects the performance of a controller without network introduced delays. Although this controller is not the ideal, the goal is to maintain this vary same performance even with introduction of the network in the control loop.

FIGURE 3: Computed torque control.

FIGURE 4: Robot response with local controller.
3 Networked Control

The generic structure of a networked control system is shown in figure 5. Basically, computer network technologies are used to connect sensors, actuators and controllers. The motivation for using this type of system is its lower costs and higher flexibility.

FIGURE 5: Networked control system.

The situation represented in figure 5 is one in that sensors and actuators are physically close to each other (in the same network not). However, this is not needed and is not important for the remainder of this work. Hence, there is not loss of generality.

In order to control the robot through the network, the computer acting as controller in section 2 becomes just an interface between the robot and the network, without executing any control function. Control functions are executed by another computer located in another network point. The data flow in the system is comprised of the transmission of sensor readings to the control node and the transmission of actuator signals to the robot interface.

FIGURE 6: Protocols in the networked control.

Figure 6 shows the protocols used to implement the system. Physical and link layers use the Ethernet protocol, mainly due to cost and availability issues, as discussed in section 1. The network layer uses the IP protocol, in order to enable Internet connectivity and to make it possible to control the robot from physically apart networks.

Transport layer uses the UDP protocol, instead of the most common TCP protocol. The TCP protocol is connection oriented and uses error detection and correction methods in order to provide a reliable communication channel between source and destination. The error recovery is base do retransmission of lost or damaged packets. The UDP protocol is a datagram oriented protocol. Messages are transmitted in the hope that they arrive at destination, but there is not any verification that they actually reached their destination.

At a first glance, TCP protocol seems to be more convenient, since it implements a reliable channel. However, TCP does not make any commitment with respect to message timing. In a control network, messages with large network delays are of no use, since the sensor data or control signal they carry would be outdated. Messages discarded by the application protocol due to large delay are also harmful since they used a portion of the system bandwidth which potentially caused delays to other messages. In this context, message retransmission, as performed by TCP, tends to be also harmful since retransmitted messages would suffer larger delay than regular messages, with higher chances to be discarded by the application protocol due to excessive delay. In extreme cases, retransmissions can cause network congestion, which forces messages to be discarded, just to be retransmitted and so on.

On the other hand, in a digital control system, usually sensors are sampled at a constant rate. Therefore, messages with sensor readings are transmitted at the same rate. For the control of the Janus robot a rate of 100Hz is used. At such rate, whenever a message is received with errors, it could be more convenient to wait for the next message with the sensor readings than retransmit the lost message to receive outdated readings. Also, prior sensor readings can be used to estimate the readings on the lost messages. With these considerations in mind it becomes natural to use the UDP protocol since it uses less bandwidth and is more efficient with respect to timing than TCP protocol. Note, however, that UDP does not provide commitments to respect to timing either. In this work, the problem of delays imposed to messages is dealt with at the application layer.

The application packet format includes a timestamp. Transmitted packets include a field with the time instant they were transmitted. This way the delay imposed to the packet can be measured and compensated for. However, it is important to note that delays can only be measured at reception. Hence, when computing the control law, only the delay from
the sensor signal is known. The delay to be suffered by the control signal is, obviously not known at the time the control law is computed. Therefore, slightly different methods need to be used to compensate the robot/controller delay and the controller/robot delay.

4 System Architecture

As depicted in figure 5, the networked control system used in this work is composed of two nodes: the robot interface and the controller. The functionalities of both nodes are implemented by using RTAI in LXRT mode.

4.1 Robot Interface Node

The program executing in the robot interface node is implemented by three LXRT hard real-time tasks, as shown in figure 7. The first task interacts with the robot hardware to read the joint sensors and send the readings to the controller node at a rate of 100Hz. Another task executes in a free-running (without timing) endless loop. This task waits for commands from controller node and executes them. Note that before the application of commanded voltages to robot actuators, the network delay (measured by time-stamps) is computed and compensated for, as detailed in section 8.2.

4.2 Controller Node

On the controller node side there are also three tasks: Two RTAI LXRT hard real-time tasks and a plain Linux process that implements the user interface. The purpose of the first RTAI task is to keep the time synchronization with the robot interface node, while the other implements the control algorithm.

The control algorithm is implemented by a free-running RTAI task which waits for the sensor readings, computes the delay introduced by the network and compensates for the delay using the robot dynamic model, thus obtaining an updated estimate for the sensor readings. This updated estimate is used to compute the desired voltage levels to be applied to the robot actuators by using a computed torque control law [3], given by expression (4).

5 Performance Metrics

In order to compare the performance of local and networked controllers, some metrics are used. Metrics used in this work are classified in two classes: metrics related to the network delay and metrics specifically related to the performance of the control system.

5.1 Network Delay Metrics

The characterization of delays imposed to packets under network traffic conditions can be done by using some metrics computed as functions of the time points defined by the transmission and reception of network packets, as shown in figure 8.

$$t_{ts}(k) = \text{Time of transmission of sensor signal for } k - \text{th cycle;}$$

$$t_{rs}(k) = \text{Time of reception of sensor signal for } k - \text{th cycle;}$$

$$t_{tc}(k) = \text{Time of transmission of control signal for the } k - \text{th cycle; and}$$

$$t_{rc}(k) = \text{Time of reception of control signal for } k - \text{th cycle.}$$
Based on the time-stamps obtained at those time moments, the following metrics are computed:

**Period of reception of sensor signal:** time interval between two consecutive receptions of sensor signals, given by expression (6).

\[ \tau_{rs}(k) = t_{rs}(k) - t_{rs}(k-1) \] (6)

**Jitter of period of reception of sensor signal:** variation on the period of reception of sensor signal with respect to the nominal period, \( T_s \), given by expression (7).

\[ J_{rs}(k) = \tau_{rs}(k) - T_s \] (7)

**Period of reception of control signal:** time interval between two consecutive receptions of control signal, given by expression (8).

\[ \tau_{rc}(k) = t_{rc}(k) - t_{rc}(k-1) \] (8)

**Jitter of period of reception of control signal:** variation on the period of reception of control signal with respect to the nominal period, \( T_s \), given by expression (9).

\[ J_{rc}(k) = \tau_{rc}(k) - T_s \] (9)

**Round-trip time:** period between the transmission of sensor signal and the reception of the corresponding control signal, given by expression (10).

\[ T_{rt}(k) = t_{rc}(k) - t_{rs}(k) \] (10)

**Jitter of round-trip time:** variation on the round-trip time with respect to its own average, \( T_{rt} \), given by expression (11).

\[ J_{rt}(k) = T_{rt}(k) - \bar{T}_{rt} \] (11)

### 5.2 Control System Performance Metrics

The control system performance is assessed through its transitory response to a step input, as shown in figure 9 and performance indexes that reflect the system performance. More specifically, the following indexes are used [7]:

**Angular position of joint** \( i \) \( \theta_i(k) \).

**Reference angular position of joint** \( i \) \( \theta_{ri} \).

**Maximum angular position of joint** \( i \) \( \theta_{pi} \).

**Settling time.** Time taken for the error become less than 3% of \( \theta_{ri} \).

**Raise time.** Time taken for the response achieve \( \theta_{ri} \) for the first time.

**Peak time.** Time take for the response achieve \( \theta_{pi} \).

**Overshoot:**

\[ M_p = \frac{\theta_{pi} - \theta_{ri}}{\theta_{ri} - \theta_{i}(0)} \] (12)

**ISE index:** system performance is assessed through expression (13). Due to the squared error, this index weights large errors more than small errors. A control system minimizing this index tends to present a fast reduction of a large initial error, but with a oscillatory response. Hence the system has low relative stability.

\[ ISE = \int e^2 dt = \int (\theta_{ri} - \theta_i(t))^2 dt \] (13)

**IAE index:** system performance is assessed through expression (14). A control system minimizing this index shows a reasonable dumping and satisfactory transient response. However, the selectivity of this index is not good.

\[ IAE = \int |e| dt = \int |\theta_{ri} - \theta_i(t)| dt \] (14)
ITSE index: system performance is assessed through expression (15). This index gives a small weight to initial errors and progressively large weights to error occurring later. This index has a better selectivity than ISE index.

\[ ITSE = \int t e^2 dt = \int t (\theta_r - \theta_i(t))^2 dt \] (15)

ITAE index: system performance is assessed through expression (16). Much like the ITSE index, this index also gives a small weight to initial errors and progressively large weights to error occurring later. Control systems minimizing this index have small overshoot and oscillations are well dumped. This index has a good selectivity.

\[ ITAE = \int t |e| dt = \int t |(\theta_r - \theta_i(t))| dt \] (16)

6 Traffic Emulator

In order to test the proposed delay compensation method under various network load conditions, a traffic emulator was developed. All network traffic between the robot interface and the controller node is routed through a computer executing the traffic emulator program, as shown in figure 10. The traffic emulator imposes delays to selected packets. These delays are generated accordingly to a statistical network load model resembling real network traffic.

![FIGURE 10: Filter for delay insertion.](image)

The traffic emulator is based on Linux kernel Netfilter facilities. In conjunction with iptables, the Netfilter can queue selected packets flowing through the Linux network subsystem to be processed by user-space programs. The traffic emulator is a program based on LXRT that processes the network packets queued by Netfilter. This utility is implemented by two RTAI LXRT hard-real time tasks and a Linux process that interfaces with the user. The LXRT tasks are the packet scheduler and the packet dispatcher.

The packet scheduler reads the queued packets, computes a delay for this packet using a statistical model and inserts the packet in a linked list ordered by the release time of the packet. The delay model is dependent upon the packet destination address.

Scheduled packets with expired release time are removed from the linked list and pushed back to the Linux network subsystem by the packet dispatcher. After dispatching all packets with release time expired, the dispatcher sleeps until the release time of the first packet which remained in the linked list.

This scheduler/dispatcher architecture was used due to the statistical nature of delays imposed to packets. Since random delays are generated, there is an implied packet reordering. By using a scheduler/dispatcher architecture, packet reordering can be easily handled.

7 Control without Delay Compensation

7.1 Controller and Robot Running Locally

In this section, the controller and the robot interface are implemented in a single program, running as a conventional controller, without network. Therefore, the period of sensor signal reception can be interpreted as the period of sensor sampling. Figures 11, 12, 13 and 14 show the period of sensor signals reception and associated jitter and their histograms.

![FIGURE 11: Period of sensor signal reception with local control.](image)
The period of reception of control signal is the period of actuator update. With the controller and the robot operating locally, this period is basically the period of reception of sensor signals. The very small differences are due to variations on computing speed of the processor. Figures 15, 16, 17 and 18, show the period of control signal reception and associated jitter and histograms.
FIGURE 18: Histogram of jitter of period of control signal reception with local control.

The round-trip time become the time to compute the control law. Figures 19, 20, 21 and 22 show the round-trip time, its jitter and associated histograms.

FIGURE 19: Round-trip time with local control.

FIGURE 20: Histogram of round-trip time with local control.

FIGURE 21: Jitter of round-trip time with local control.

FIGURE 22: Histogram of jitter of round-trip time with local control.

The robot response with local control was showed in figure 4. Table 1 shows some performance criteria for the local control system.

<table>
<thead>
<tr>
<th></th>
<th>Joint 1</th>
<th>Joint 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot (%)</td>
<td>43.66</td>
<td>6.19</td>
</tr>
<tr>
<td>Settling time (s)</td>
<td>1.41</td>
<td>0.27</td>
</tr>
<tr>
<td>Raise time (s)</td>
<td>0.14</td>
<td>0.015</td>
</tr>
<tr>
<td>Peak time (s)</td>
<td>0.20</td>
<td>0.02</td>
</tr>
<tr>
<td>ISE index (rad²)</td>
<td>79.771579</td>
<td>11.442537</td>
</tr>
<tr>
<td>IAE index (rad)</td>
<td>69.843623</td>
<td>11.692929</td>
</tr>
<tr>
<td>ITSE index (s rad²)</td>
<td>10.999367</td>
<td>0.162832</td>
</tr>
<tr>
<td>ITAE index (s rad)</td>
<td>57.055757</td>
<td>23.881627</td>
</tr>
</tbody>
</table>

TABLE 1: System performance with local control.

As commented in section 2, the performance obtained with the local controller is the ideal to be obtained with the method for delay compensation. Table 2 shows averages and standard deviations for some metrics for the local control.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Average (ms)</th>
<th>Standard Deviation (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{rs}$</td>
<td>9.99946958</td>
<td>0.06608492</td>
</tr>
<tr>
<td>$J_{rs}$</td>
<td>-0.00053042</td>
<td>0.06608492</td>
</tr>
<tr>
<td>$T_{rc}$</td>
<td>9.99941975</td>
<td>0.07370745</td>
</tr>
<tr>
<td>$J_{rc}$</td>
<td>-0.00058025</td>
<td>0.07370745</td>
</tr>
<tr>
<td>$T_{rt}$</td>
<td>0.09862190</td>
<td>0.00668446</td>
</tr>
<tr>
<td>$J_{rt}$</td>
<td>0</td>
<td>0.00668446</td>
</tr>
</tbody>
</table>

**TABLE 2:** Metrics for local control.

### 7.2 Controller and Robot over the Network

In this case, controller and robot are communicating over the network. Therefore, while the sensor sampling period is basically constant, varying only due to jitter introduced by the real time system, the sensor reception period varies due to delays imposed by the network. Figures 23, 24, 25 and 26 show the period of sensor signals reception and associated jitter and their histograms.

**FIGURE 23:** Period of sensor signal reception with networked control.

**FIGURE 24:** Histogram of period of sensor signal reception with networked control.

**FIGURE 25:** Jitter of period of sensor signal reception with networked control.

**FIGURE 26:** Histogram of jitter of period of sensor signal reception with networked control.

The period of control signal reception is the period for updating the actuators. When controller and robot are operating through a network, this period varies due to delays on communication from robot to controller and from controller to robot. Figures 27, 28, 29 and 30 show the period of control signal reception and associated jitter and histograms.

**FIGURE 27:** Period of control signal reception with networked control.

**FIGURE 28:** Histogram of period of control signal reception with networked control.

**FIGURE 29:** Jitter of period of control signal reception with networked control.

**FIGURE 30:** Histogram of jitter of period of control signal reception with networked control.
The round-tip time is also influenced by delays introduced by the network. Figures 31, 32, 33 and 34 show the round-trip time, its jitter and associated histograms.

**FIGURE 28:** Histogram of period of control signal reception with networked control.

**FIGURE 29:** Jitter of period of control signal reception with networked control.

**FIGURE 30:** Histogram of jitter of the period of control signal reception with networked control.

**FIGURE 31:** Round-trip time with networked control.

**FIGURE 32:** Histogram of round-trip time with networked control.

**FIGURE 33:** Jitter of round-trip time with networked control.
FIGURE 34: Histogram of jitter of round-trip time with networked control.

Figure 35 shows the robot response with networked control. Table 3 shows some performance criteria for the networked control system. Note that settling time was not measured due to very bad system performance.

FIGURE 35: Robot response with networked control.

TABLE 3: System performance with networked control.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Joint 1</th>
<th>Joint 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot (%)</td>
<td>44.93</td>
<td>6.46</td>
</tr>
<tr>
<td>Settling time (s)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Raise time (s)</td>
<td>0.16</td>
<td>0.015</td>
</tr>
<tr>
<td>Peak time (s)</td>
<td>0.25</td>
<td>0.02</td>
</tr>
<tr>
<td>ISE index (rad²)</td>
<td>1802.181821</td>
<td>686.674517</td>
</tr>
<tr>
<td>IAE index (rad)</td>
<td>1292.393529</td>
<td>760.289006</td>
</tr>
<tr>
<td>ITSE index (s rad²)</td>
<td>9765.848334</td>
<td>3404.591958</td>
</tr>
<tr>
<td>ITAE index (s rad)</td>
<td>6943.237558</td>
<td>4066.603233</td>
</tr>
</tbody>
</table>

TABLE 4: Metrics for networked control.

It can be verified, by comparing figures 4 and 35 that delays introduced by the network in the control loop lead to a severe degradation of the control system performance.

8 Delay Compensation

In order to recover the performance obtained with the local controller, even with the introduction of the network in the control loop, this paper proposes to compensate for delays imposed to packets. It is important to note that delays are introduced in the control loop in two points. There are the robot/controller delay, which is imposed to packets carrying sensor signals and the controller/robot delay, which is imposed to control signals. It is clear that both delays should be compensated for, but while the robot/controller delay can be determined by the controller (for example, using time-stamps) before computing the control law, the controller/robot delay can only be determined by the robot, hence after the computing of the control signal. So, each type of delay should be compensated for in slightly different way.

8.1 Robot/Controller Delay

Sensor readings are transmitted to controller at \( t_{ts} \), but due to network delay, they are received at \( t_{rs} > t_{ts} \), as shown in figure 8. The delay compensation is performed by obtaining an estimate of the sensor values that would be read by the time of control law computation, that means at \( t_{rs} \). This can be done by propagating the received sensor values through the robot dynamic model (3). That is

\[
\dot{x}(t_{rs}) = x(t_{ts}) + \int_{t_{ts}}^{t_{rs}} f(x) + g(x)udt \tag{17}
\]

where

\[
\dot{x}(t_{rs}) = \text{Estimate of } x \text{ at } t_{rs};
\]

\[
x(t_{ts}) = \text{value of } x \text{ at } t_{ts} \text{ (readings of the sensors)};
\]
The integral in expression 17 can be computed numerically, for example using the Runge-Kutta method [8]. It is important to note that although numerical integration are interactive procedure and hence not recommended for real-time operation, here just a few interactions will be performed to propagate $x$ from $t_s$ to $t_r$. The value of $\dot{x}(t_r)$ is then used to compute the control signal by using expression (4), as shown in figure 36. Note that, by neglecting the time taken to computed the control law implies $t_r = t_c$.

![Figure 36: Robot/controller delay compensation.](image)

FIGURE 36: Robot/controller delay compensation.

Figure 37 shows the robot performance with robot/controller delay compensation for a network load condition similar to that seen while obtaining figure 35. Table 5 shows some performance criteria for the control system with robot/controller delay compensation. As expected, the robot/controller delay compensation improves the control system performance. However, just the delays in one direction were compensated for. It can be supposed that by also compensating delays in the opposite direction, the performance can be further improved.

![Figure 37: Robot response with robot/controller delay compensation.](image)

FIGURE 37: Robot response with robot/controller delay compensation.

<table>
<thead>
<tr>
<th></th>
<th>Joint 1</th>
<th>Joint 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot (%)</td>
<td>31.23</td>
<td>16.74</td>
</tr>
<tr>
<td>Settling time (s)</td>
<td>2.15</td>
<td>1.82</td>
</tr>
<tr>
<td>Raise time (s)</td>
<td>0.105</td>
<td>0.024</td>
</tr>
<tr>
<td>Peak time (s)</td>
<td>0.13</td>
<td>0.03</td>
</tr>
<tr>
<td>ISE index (rad²)</td>
<td>60.699671</td>
<td>35.932496</td>
</tr>
<tr>
<td>IAE index (rad)</td>
<td>105.430378</td>
<td>29.450118</td>
</tr>
<tr>
<td>ITSE index (s rad²)</td>
<td>28.675426</td>
<td>21.110612</td>
</tr>
<tr>
<td>ITAE index (s rad)</td>
<td>356.038187</td>
<td>27.937111</td>
</tr>
</tbody>
</table>

TABLE 5: Control system performance with robot/controller delay compensation.

### 8.2 Controller/Robot Delay

Contrariwise to the robot/controller delay, the controller/robot delay is not known by the time of control law computation, which precludes the computation of a value for $u$ including the compensation. Of course, the controller/robot delay can be determined by the robot interface node using time-stamps. However, it does not make sense to compensate for the controller/robot delay using a method similar to the one used to compensate for the robot/controller delay, since this would be equivalent to compute the control law locally. Therefore, in order to justify the use of networked control and to explore its advantages, the additional complexity on the robot interface node should be small. This can be done by pre-compensating for the controller/robot delay while computing the control law, see figure 38.

![Figure 38: Robot/controller and controller/robot delay compensation.](image)

FIGURE 38: Robot/controller and controller/robot delay compensation.

Since the robot/controller delay is unknown to the controller node, the control signal compensated for multiple delay values can be computed and transmitted to the robot. The robot interface node can then determine the controller/robot delay and select the appropriate pre-compensated control signal. Hence, the control signal $u$ transmitted to the robot becomes a matrix $n \times m$, where $n$ is the number of joints of the robot and $m$ is the number of steps ahead for which the pre-compensated control signal is computed, accordingly to expression (18).

$$
\begin{align*}
u &= \begin{bmatrix} u_0 & u_1 & \cdots & u_m \end{bmatrix}
\end{align*}
$$

(18)
where each \( u_k \) is determined by expression (5) propagated \( k \) steps ahead, that means

\[
u_k = h(\hat{x}(t_c + kT_s), \dot{q}_r(t_c + kT_s), \ddot{q}_r(t_c + kT_s))
\]

where \( T_s \) is the control sampling period and the estimates \( \hat{x}(t_c + kT_s) \) are given by the \( k \) steps ahead propagation of the sensor readings, after its compensation for the robot/controller delay. That means

\[
\hat{x}(t_c + (k+1)T_s) = \hat{x}(t_c + kT_s) + \int_{t_c+kT_s}^{t_c+(k+1)T_s} f(x) + g(x)u_k \, dt
\]

(20)

with \( \hat{x}(t_c) = \hat{x}(t_{s0}) \). That is, expression (20) is the propagation to future of the robot/controller delay compensated through expression (17). However, when used in conjunction with expression (20), expression (17) should consider that the control signal effectively applied to the robot is not the same computed by expression (5), as done in section 8.1. Here, the control signal effectively applied is chosen among the multiple values of \( u_k \) by the robot interface node. That is, expression (17) is modified to:

\[
\hat{x}(t_{s0}) = x(t_{s0}) + \int_{t_{s0}}^{t_{ss}} f(x) + g(x)u_{k,s} \, dt
\]

(21)

where \( u_{k,s} = u(t_{s0}) \) is the control signal applied during the \( (t_{s0}, t_{ss} + T_s) \) interval.

Figure 39 shows the robot response with robot/controller and controller/robot delay compensation for a load condition similar to that seen while obtaining figures 35 and 37.

Table 6 shows some performance criteria for the control system with robot/controller and controller/robot delay compensation. Again, as anticipated, there is an improvement on the control system performance with respect to the experiments without or with partial delay compensation.

<table>
<thead>
<tr>
<th>Joint 1</th>
<th>Joint 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overshoot (%)</td>
<td>36.70</td>
</tr>
<tr>
<td>Settling time (s)</td>
<td>1.71</td>
</tr>
<tr>
<td>Raise time (s)</td>
<td>0.12</td>
</tr>
<tr>
<td>Peak time (s)</td>
<td>0.15</td>
</tr>
<tr>
<td>ISE index ((\text{rad}^2))</td>
<td>72.555937</td>
</tr>
<tr>
<td>IAE index ((\text{rad}))</td>
<td>87.3301104</td>
</tr>
<tr>
<td>ITSE index ((\text{s rad}))</td>
<td>14.4116511</td>
</tr>
<tr>
<td>ITAE index ((\text{s rad}^2))</td>
<td>214.4564410</td>
</tr>
</tbody>
</table>


9 Conclusion

This paper presented the control of a robotic manipulator through a conventional IP network, with packets subject to nondeterministic delays in its route from origin to destination. The delays imposed to packets were characterized and methods to compensate for network delays were proposed. These methods were capable of recover a large amount of the performance lost due to network delays.

Table 7 shows the performance indexes (the sum for all joints) for all strategies presented in this work. The local control is the ideal case. The introduction of the network in the control loop causes a degradation of the system performance, with the corresponding increase on indexes. If no delay compensation is used the system performance becomes inadequate, leading the system to instability. The introduction of delay compensation recovers the system performance, which approaches the one obtained with local control.

<table>
<thead>
<tr>
<th>Method</th>
<th>ISE</th>
<th>IAE</th>
<th>ITSE</th>
<th>ITAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local control</td>
<td>91.214</td>
<td>81.5365</td>
<td>11.1621</td>
<td>80.9573</td>
</tr>
<tr>
<td>Networked control</td>
<td>2488.85</td>
<td>2052.67</td>
<td>13170.43</td>
<td>11009.83</td>
</tr>
<tr>
<td>Networked control with robot/controller compensation</td>
<td>96.632</td>
<td>134.8803</td>
<td>49.786</td>
<td>383.9752</td>
</tr>
<tr>
<td>Networked control with robot/controller compensation</td>
<td>98.8265</td>
<td>96.6206</td>
<td>13.3423</td>
<td>207.8266</td>
</tr>
</tbody>
</table>

TABLE 7: System performance for different strategies.

A sequence of this work is the study of more sophisticated estimation techniques. In particular the
use of the Kalman filter [4] [5] is under study and related results should be published soon. The use of delay compensation methods proposed here in conjunction with QoS as a mean to bound the maximum delay is also a promising direction.

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 Informatics (PROADI) under Grant nr. 01/0567.0.

References


