

# Bandwidth Management for Distributed Control of Highly Articulated Robots\*

Manel Velasco, Pau Martí and Manel Frigola  
*Intelligent Robots and Systems Lab, Automatic Control Department*  
*Technical University of Catalonia*  
*Pau Gargallo, 5, 08028 Barcelona, Spain*  
{*manel.velasco,pau.marti,manel.frigola*}@upc.es

**Abstract**—An optimal bandwidth allocation policy for axis distributed control using networked control systems (NCS) is presented. First, the benefits of structuring highly articulated robots using networked control systems are discussed. Afterwards, an optimal bandwidth allocation policy for axes distributed control that allows to enhance robot tracking within the available bandwidth is introduced. It will be shown that axes control performance and thus tracking can be significantly improved by using feedback to dynamically allocate bandwidth to axis controllers as a function of the current state of each axis. Simulation results on a multiple axes robot corroborate our approach.

**Index Terms**—Bandwidth Management, Distributed Control, Performance Optimization, Highly Articulated Robots.

## I. INTRODUCTION

Control of articulated robots is usually based on a two level hierarchical communication architecture. At the lower level, for each robot axis, sensors and actuators are directly wired to the controller using a *point-to-point* configuration. Each controller performs axis control (sensing, control algorithm computation and actuation) locally. At the upper level, a master controller coordinates all axis controllers for the overall robot control (e.g., trajectory tracking) using a *master/slave* configuration over a *common bus* architecture.

Such hierarchical architectures for robot control do not suit modern automation requirements where the goal is to design flexible systems that can accomplish various tasks with small reconfiguration costs. The reconfiguration of a conventional (point-to-point) architecture to expand physical setups (e.g., adding new robot tools) and functionality (e.g., adding force feedback to an existing hand) is not an easy task. In addition, on-line task monitoring has to be performed in a two steps procedure, starting from the master controller to each axis controller, and from each axis controller to sensors and actuators, in such a way that no direct operation (control and supervision) from the master controller to sensors and actuators is allowed. In the end, in hierarchical architectures, complex robot control algorithms (e.g., arm and hand coordination) require intensive messaging, which difficults the implementation. These drawbacks result in inefficient systems and increased costs, as further discussed in [1] or [2].

\*This work is partially supported by Spanish Ministerio de Ciencia y Tecnología Projects ref. DPI2001-0822 and DPI2002-01621.

An alternative communication architecture for highly articulated robots is based on flattening the hierarchy. That is, to apply the intelligent network concept where all nodes (master controller, axis controllers, and sensors and actuators) have processing capacity and share a common bus in a completely distributed and networked feedback control system [3]. This architecture, also known as *networked control systems* (NCS), allows for the modularization of the functionality, providing standard interfaces for interchangeability and interoperability, thus suiting modern automation systems. In addition, monitoring and maintenance operation can be easily performed because all the required data can be made available to all nodes in the network. See [4] or [5] for examples of such architectures applied to robots.

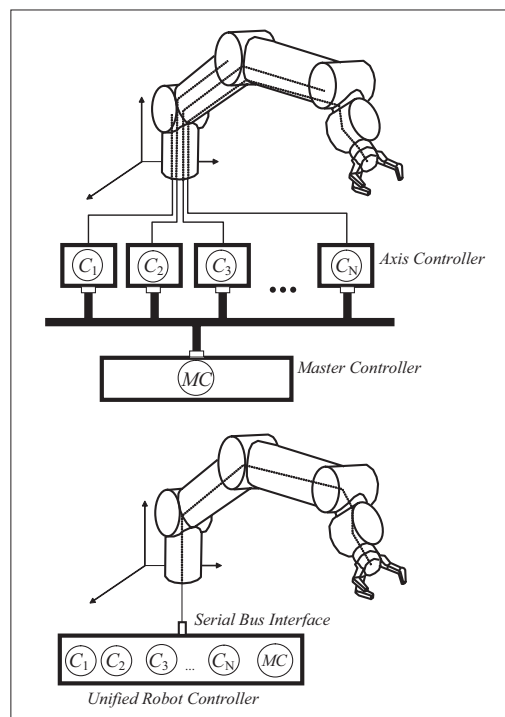


Fig. 1. Hierarchical (top) vs. networked-based architecture (bottom).

Figure 1 illustrates both types of architectures, hierarchical (top) versus networked-based (bottom). For networked-based architectures, in the case of robot control, axis

controllers ( $C_1, \dots, C_N$ ) and master controller ( $MC$ ) can be either implemented on separate nodes or centralized on a single processor-based node (as illustrated in Figure 1 bottom). In the later, all tasks can be appropriately scheduled using well known processor time scheduling techniques (see for example [6] for a review of major scheduling techniques), thus saving costs.

However, in NCS, the insertion of networks into control loops (between sensors, actuators and controllers) can make the analysis and design of control applications complex [7], because the key for achieving successful implementations is that almost no local action can be taken in isolation from the rest of the system.

This paper focuses on the problem of bandwidth allocation and robot axis control performance improvement in highly articulated robots that are based on a networked based control architecture. Most traditional bandwidth allocation techniques for control loops closed over communication networks rely on static strategies, based on *a priori* characterization of the network workload. These techniques use fixed parameters that are configured at system setup time. At run-time, the network is shared by all nodes according to the pre-established allocations regardless of the dynamics of the control applications, thus working *open loop*.

Open loop bandwidth allocation techniques work well because they guarantee a constant bandwidth share to each control loop, allowing them to meet given control performance specifications. However, a closer look at the behavior of control loops and the relation between control performance and controller execution rate indicates that open loop policies may not be optimal for bandwidth constrained systems. As previously suggested in [8], a controller may not require the assigned execution rate if the controlled system is in equilibrium. In this case, the contribution of each control job can be considered more or less useless, i.e., bandwidth is wasted. This underutilized bandwidth could be more usefully employed by other nodes (or control tasks) with higher processing demands. On the other hand, if a controlled system is affected by a perturbation and brought away from its equilibrium point, an increase in the rate of the controller will decrease system deviation and hasten system recovery, thus improving control performance.

Taking this observation as a baseline for the current work on axis control performance optimization in highly articulated robots, the research has been focused on dynamic bandwidth management techniques that allocate bandwidth to nodes at run-time based on *feedback* information from the jobs that the controllers are performing. This feedback-based approach to network management takes advantage of the fact that the performance of classically designed controllers is improved by allocating more bandwidth to control loops exactly when they need it the most, i.e. when their controlled axis is experiencing the greatest deviation from the equilibrium point (or reference signal).

As a key contribution of this paper, an optimal bandwidth

allocation policy is presented based on feedback from the robot axis dynamics, i.e., from their *state*, that maximizes overall robot control performance within the available communication bandwidth. It is argued that the optimization of axis control performance, when bandwidth is limited and allocated as a function of the state of each robot axis, can be formulated as a linear constrained optimization problem [9] whose solution provides the optimal (dynamical) bandwidth allocation policy.

## II. PROBLEM DESCRIPTION

### A. System architecture

The system considered is a highly articulated robot of  $n$  axis based on a networked control architecture (as in Figure 1 bottom). Each axis is equipped with an smart sensor and actuator with communication capabilities that can run at different rates. The sensor is capable of transmitting samples to the network and the actuator is capable of receiving control signals from the network. All control tasks that implement axes control are executed on a single CPU enabled with a real-time scheduler that can guarantee and run-time adapt different execution rates for each control task (see for example [10] or [11] for flexible scheduling techniques providing this feature). For each axis, the sensor, the controller (i.e., control task), and the actuator work on closed loop operation for the axis control over the network.

The operation of the feedback based bandwidth allocation architecture that is considered can be illustrated by the controller and bandwidth manager pseudo-code shown in Figure 2. Each controller (Figure 2(a)), with varying execution period  $h_i$ , reads from the network its axis state  $x_i$  and calculates the control signal  $u_i$  (based on  $h_i$  and  $x_i$ ), which is sent to the axis actuator over the network. Afterwards, the controller wakes up the bandwidth manager in order to trigger the run-time bandwidth re-allocation. The bandwidth manager (Figure 2(b)), taking into account the state of all axes  $\vec{x} = [x_1, \dots, x_n]$ , chooses the controller whose axis is facing the most significant error, and assigns bandwidth (and thus sampling periods  $\vec{h} = [h_1, \dots, h_n]$ ) to all controllers accordingly. Afterwards, the manager sleeps again and the scheduler continues dispatching control jobs taking into account the newly specified sampling periods.

Besides axis controllers and the bandwidth manager, the master controller is always running to coordinate the axis controllers, as well as, providing other functionalities.

### B. Problem Formulation

The problem to be solved is to determine the allocation of network bandwidth that will be assigned to each controller such that the overall control system performance is maximized.

Let (1) and (2) be the differential equations (called *state* and *output* equations, respectively) that describe the linear time invariant dynamics of each  $i^{th}$  axis,  $i = 1, \dots, n$ , where  $u_i(t)$  is the *control input* to the axis, and the vector  $x_i(t) = [x^1(t), \dots, x^n(t)]$  is the state of the axis at time

```

Axis_controller ;
begin
  while (always)
    begin
       $x_i := \text{read\_sensor}(i)$ 
       $u_i := \text{calculate\_control\_signal}(h_i, x_i)$ 
       $\text{write\_control\_signal}(u_i, i)$ 
       $\text{weake\_up}(\text{Bus\_scheduler})$ 
       $\text{sleep}(h_i)$ 
    end
  end
end

```

(a) Controller

```

Bandwidth_manager
begin
  while (always)
    begin
       $i := \text{obtain\_most\_significant\_axis\_error}(\vec{x})$ 
       $\vec{h} = \text{assign\_bandwidht}(i)$ 
       $\text{sleep}()$ 
    end
  end
end

```

(b) Bandwidth manager

Fig. 2. Pseudo-code for axis controllers and for the bandwidth manager implementing the feedback allocation management

$t$  and its elements are called *state-variables*. Full state availability is assumed.<sup>1</sup>

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) \quad t \in \mathbb{R}^+ \quad (1)$$

$$y_i(t) = C_i x_i(t) + D_i u_i(t) \quad (2)$$

If the input  $u_i(t)$  is given by a discrete state feedback control law designed according to the discretization of (1) and (2) [12], the new system dynamics (including the axis model and the control law, i.e., closed-loop system) are specified by the state-space representation given in (3), where  $L_i(h_i)$  is the control law.

$$x_{i,n+1} = \Phi_i(h_i)x_{i,n} + \Gamma_i(h_i)L_i(h_i)x_{i,n} \quad (3)$$

In (3),  $h_i$  is the sampling period, which will vary depending on the bandwidth assigned to each control loop. The relation between bandwidth and sampling period for each control loop is given by (4), where  $m_i$  is the time spent on the messaging required to perform each closed loop operation (which may include data exchange from sensor to controller and from controller to actuator, as well as the time spent to execute the controller).

$$b_i = \frac{m_i}{h_i} \quad (4)$$

By assuming  $m_i$  in (4) constant, any change on  $b_i$  will directly imply a change on  $h_i$  (and vice versa). Henceforth, either  $b_i$  or  $h_i$  will be used to denote bandwidth (or sampling period). For each control loop, it is assumed that the required messaging to carry out all sensor-controller-actuator transactions is done within each sampling period (i.e., deadline is equal to the sampling period). And it is assumed that the architecture is based on a real-time network that guarantees all the deadlines upon any bandwidth allocation among all control loops.

Due to bandwidth limitations, it is assumed that not all control loops can simultaneously run at the highest sampling frequency, providing the best possible control performance equivalent to what they would provide if each one had a dedicated network. The problem to be solved is how to assign bandwidth  $b_i$  to each control loop taking into

<sup>1</sup>For the sake of simplicity, the  $i$  subscript is omitted when not required.

account the state of each axis such as the overall control performance is optimized.

Without loss of generality, for all axes, if the equilibrium point (or reference signal) is considered to be zero, the norm of the state vector (in (1)),  $|x_i(t)|$ , is the distance that measures how far each axis is from its equilibrium point at any given time  $t > 0$ . This measure (also called *error* (5)) is defined as the feedback information that each controller, at each sample, will forward to the bandwidth manager for the run-time bandwidth allocation.

$$e_i = |x_i(t)| \quad (5)$$

For each control loop, a performance criterion  $p_i(b_i)$  that relates control performance under different bandwidth allocations,  $b_i$ , is specified. In fact, the relation between control performance (measured using standard quadratic or linear performance index) and a range of allowed periods (given by different bandwidth allocations, as in equation (4)) can be approximated by a linear relationship [13]<sup>2</sup>. Therefore, for a given control loop  $i$ , its performance criterion  $p_i(b_i)$  is approximated by a linear increasing function (6), that establishes the following relation for each control loop: *the higher the portion of allocated bandwidth (i.e., the shorter the sampling period), the better the control performance*. The  $\alpha_i$  parameter in (6) is specific for each control loop (depending on each axis) and can be determined prior to system run-time.

$$p_i(b_i) = \alpha_i b_i \quad (6)$$

This technique requires controllers capable of running with different sampling frequencies. For systems given by equation (3), controllers are designed specifying a range of sampling periods  $h_i \in [h_i^{max} \dots h_i^{min}]$  for which the closed loop requirements are met, and are allowed to execute with a run-time period that belongs to the specified range, adapting the gains accordingly (following the adaptive techniques introduced in [14]). Closed-loop stability is analyzed using the approach described in [15].

<sup>2</sup>Although this linear approximation is not an oversimplification and it covers a wide class of control systems, as it will be discussed in Section III, the optimal bandwidth allocation policy also admits performance criteria in the form of polynomials of degree less than five.

At the system level, each control loop  $c_i$  can be characterized by its bandwidth  $b_i$ , its performance criterion  $p_i$ , and its axis error  $e_i$ , represented by (7).

$$\tau_i = \{b_i, p_i, e_i\} \quad (7)$$

With this information, for a given set of  $n$  control loops,  $c_1, \dots, c_n$ , the problem is to determine the bandwidths  $b_i$ ,  $i = 1, \dots, n$ , such that all control operations are schedulable and the overall axes control system performance is maximized.

### III. OPTIMAL BANDWIDTH MANAGEMENT POLICY

#### A. Generic Formulation

The bandwidth allocation problem can be formulated as a generic constrained optimization problem (equations (8) and (9)), where the solution is a vector  $\vec{b} = [b_1, b_2, \dots, b_n]$  that maximizes the control performance delivered by the set of control loops, represented by the objective (vector) function  $g$  in (8), restricted to the utilization feasibility constraint specified in (9), where  $U_d$  is the desired global bandwidth utilization factor for the set of control loops.

$$\text{maximize} \quad g(p_i(r_i), e_i) \quad (8)$$

$$\text{subject to} \quad \sum_{i=1}^n b_i \leq U_d \quad (9)$$

The absolute maximum  $\vec{b}$  may lie either in the interior, on the boundary, or at the extreme points of the feasibility set defined by (9). A generic algorithm to find the solution can be summarized in four steps (as detailed in [9]):

*Step 1:* Search for local relative maxima in the interior of the feasibility set by solving the set of equations specified by (10), where  $\frac{\partial g}{\partial b_i}$  are the partial derivatives of  $g$  with respect to each  $b_i$ , and keep those  $\vec{b}$  that, being interior points of the feasibility set (conforming with the restriction (9)), maximize  $g$ .

$$\frac{\partial g}{\partial b_1} = 0, \quad \frac{\partial g}{\partial b_2} = 0, \quad \dots, \quad \frac{\partial g}{\partial b_n} = 0 \quad (10)$$

*Step 2:* Search for local relative maxima in the boundary of the feasibility set by solving the set of equations specified by (11), and keep those  $\vec{b}$  that maximize  $g$ .

$$\begin{aligned} \frac{\partial g([U_d - \sum_{j \leq n, j \neq 1} b_j, b_2, b_3, \dots, b_n])}{\partial b_i} &= 0, \quad i \leq n, \quad i \neq 1 \\ \frac{\partial g([b_1, U_d - \sum_{j \leq n, j \neq 2} b_j, b_3, \dots, b_n])}{\partial b_i} &= 0, \quad i \leq n, \quad i \neq 2 \\ &\vdots \\ \frac{\partial g([b_1, b_2, b_3, \dots, U_d - \sum_{j \leq n, j \neq n} b_j])}{\partial b_i} &= 0, \quad i \leq n, \quad i \neq n \end{aligned} \quad (11)$$

*Step 3:* Search for the values of the objective function  $g$  within the feasibility set extremes as specified in (12), and keep those  $\vec{b}$  that maximize  $g$ .

$$\begin{aligned} g([U_d, 0, \dots, 0]) \\ g([0, U_d, \dots, 0]) \\ \vdots \\ g([0, 0, \dots, U_d]) \end{aligned} \quad (12)$$

*Step 4:* Choose a  $\vec{b}$  among those obtained in *Step 1*, *2* and *3* that maximize  $g$ .

Depending on the objective function  $g$ , solving the optimization problem may not be feasible for run-time bandwidth allocation, because of the computational complexity, which may introduce non negligible overhead. However, in the case of control loops characterized as described in Section II-B, the optimization problem can be simplified, it is directly solvable, and the algorithm that obtains the solution can feasibly be executed at run-time (because it incurs minimum overhead), as it is explained next.

#### B. Simplification

Assuming that each controller is independent in the sense of controlling an independent axis, as it was assumed in the problem formulation (see in Section II-B), the function  $g(\cdot)$  that links all of the control performance benefits (which are given by the performance criteria  $p_i$  defined in (6)) can be considered as the sum (possibly weighted) of all individual benefits obtained by each control loop.

Each performance criterion can be weighted,  $w_i$ , in order to provide a mechanism allowing appropriate comparisons among the control loops in the system. Further, weights can be used to define the kinematics and dynamics relations among robot axes.

In addition, by defining the re-scaling of each performance criterion to account for the each axis error (defined in (5)) as  $e_i p_i$ , for the given set of  $n$  control loops, it is possible to rewrite the optimization problem as in (13) and (14).

$$\text{maximize} \quad \sum_{i=1}^n w_i e_i p_i(b_i) \quad (13)$$

$$\text{subject to} \quad \sum_{i=1}^n b_i \leq U_d \quad (14)$$

The complexity of the solution of the optimization problem stated in (13) and (14) depends on each function  $p_i(b_i)$  due to the fact that equations (10) and (11) have been simplified to the set of equations specified by (15) and (16) (because  $g$  has turned into a sum), where  $q_i = w_i e_i p_i(b_i)$ .

$$\frac{\partial q_1}{\partial b_1} = 0, \quad \frac{\partial q_2}{\partial b_2} = 0, \quad \dots, \quad \frac{\partial q_n}{\partial b_n} = 0 \quad (15)$$

$$\begin{aligned}
\frac{\partial q_1(U_d - b_2 - b_3 - \dots - b_n)}{\partial b_i} &= 0, \quad i \leq n, i \neq 1 \\
\frac{\partial q_2(U_d - b_1 - b_3 - \dots - b_n)}{\partial b_i} &= 0, \quad i \leq n, i \neq 2 \quad (16) \\
&\vdots \\
\frac{\partial q_n(U_d - b_1 - b_2 - \dots - b_{n-1})}{\partial b_i} &= 0, \quad i \leq n, i \neq n
\end{aligned}$$

If the performance criteria  $p_i$  are linear (as it was assumed in our problem formulation in Section II-B), the optimization problem becomes linear, and the solution  $\vec{b} = [b_1, b_2, \dots, b_n]$  can be found by performing a simple search (i.e., performing *Step 3*) because equations (15) and (16) corresponding to *Step 1* and 2, are not properly determined. That is, if all performance criteria  $p_i$  are linear ( $p_i = \alpha_i b_i$ ) in (15), those equations up with the set of equations defined by (17), which are not determined.

$$\begin{aligned}
\frac{\partial q_1}{\partial b_1} &= \frac{\partial w_1 e_1 \alpha_1 b_1}{\partial b_1} = w_1 e_1 \alpha_1 = 0 \\
\frac{\partial q_2}{\partial b_2} &= \frac{\partial w_2 e_2 \alpha_2 b_2}{\partial b_2} = w_2 e_2 \alpha_2 = 0 \quad (17) \\
&\vdots \\
\frac{\partial q_n}{\partial b_n} &= \frac{\partial w_n e_n \alpha_n b_n}{\partial b_n} = w_n e_n \alpha_n = 0
\end{aligned}$$

The same happens with the set of equations specified in (16). Therefore, by simply performing *Step 3* customized for the problem stated in (13) and (14), that is, by evaluating the equations listed in (18) the optimal resource allocation will be found. Note that (18) is equivalent to finding the maximum  $w_i e_i \alpha_i$ ,  $i = 1 \dots n$ .

$$\begin{aligned}
g([U_d, 0, \dots, 0]) &= q_1(U_d) + \sum_{\substack{i \neq 1 \\ i \leq n}} q_i(0) = w_1 e_1 \alpha_1 U_d \\
g([0, U_d, \dots, 0]) &= q_2(U_d) + \sum_{\substack{i \neq 2 \\ i \leq n}} q_i(0) = w_2 e_2 \alpha_2 U_d \quad (18) \\
&\vdots \\
g([0, 0, \dots, U_d]) &= q_n(U_d) + \sum_{\substack{i \neq n \\ i \leq n}} q_i(0) = w_n e_n \alpha_n U_d
\end{aligned}$$

### C. Solution

The optimal solution  $\vec{b} = [b_1, b_2, \dots, b_n]$  of the optimization problem (13) and (14) is  $\vec{b} = [0, 0, \dots, 0, b_i = U_d, 0, \dots, 0]$ ,  $i \in [1, \dots, n]$  such that  $w_i e_i \alpha_i$  is maximum  $\forall i \in [1 \dots n]$ , if each of the  $n$  control loops is described as in (7).

In terms of bandwidth allocation, the theorem states that we should assign all the available bandwidth (that is,  $U_d$ ) to the control loop with maximum  $w_i e_i p_i$ . If all of the functions  $p_i$  and all the weights  $w_i$  are the same (i.e., all the axis and control algorithms are equal), all of the available bandwidth should be assigned to the control loop with the largest error  $e_i$ . In practice it is needed to assign a minimum

rate to the rest of the control loops so that stability tests can be performed and they can continue to monitor the state of their axis. This result dictates that the control loop with the largest error should receive all of the bandwidth remaining after every control loop has received its minimum.

If the performance criteria  $p_i$  are not linear but still polynomial functions on  $b_i$  of degree less than five, an analytical solution can be found [16] following *Steps 1*, 2 and 3 by solving equations (15), (16) and (12), turning the solution into a computationally feasible algorithm for a run-time bandwidth allocation.

### D. Optimal Policy

The optimal bandwidth management policy is summarized in (19), where  $n$  is the number of control loops and  $b_j^{min}$  (corresponds to  $h_j^{max}$ ) is the guaranteed minimum utilization of the control loop  $j$ .

$$b_i = \begin{cases} U_d - \sum_{\substack{j \neq i \\ j \leq n}} b_j^{min}, & \text{if } w_i e_i p_i(r_i) \text{ is maximum} \\ b_i^{min}, & \text{otherwise} \end{cases} \quad (19)$$

The optimal policy keeps track of which control loop has maximum  $w_i e_i p_i$ . Therefore, the dynamic bandwidth allocation can be completed by a linear scan of the list of  $n$  control loops in  $O(n)$  time. Note that dynamic bandwidth allocation only occurs when the control loop with the maximum  $w_i e_i p_i$  changes, which greatly reduces the overhead.

## IV. PERFORMANCE EVALUATION

In this section the simulation results of the optimal bandwidth management policy for the networked architecture for distributed control of highly articulated robots are presented.

### A. Simulation Setup

In a first simulation, each axis is driven by a DC motor, which has been specified by the linear time-invariant state space model given by (20), where  $b$  is the damping ratio of the mechanical system,  $J$  is the moment of inertia of the rotor,  $K$  is the electro-motive force constant,  $L$  is the electric inductance,  $R$  is the electric resistance, the state variables  $\dot{\theta}$  and  $i$  are the rotational speed and electric current and the input  $u$  is the voltage.

$$\frac{d}{dt} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u \quad (20)$$

For the simulation, each DC motor has been customized equally for all axes. Each control loop implements the same parametric control law obtained by standard pole placement [12], which is parameterized on the sampling period.

All controllers and all motors have been defined to be the same because it simplifies the performance analysis. Remind that in this case,  $w_i$  and  $p_i$  are equal for all control loops, which means that the error  $e_i$  is the main driving factor in the optimal policy. However, using different motors ( $p_i$ ) or weights ( $w_i$ ) would not materially affect the results.

## B. Performance Analysis

Before evaluating the optimal policy, it will be shown that the performance criterion ( $p_i$  in 6) that relates control performance under different bandwidth allocations for the example of the DC motor can be approximated by a linear relation (corroborating the assumption made in section II-B). Figure 3 shows the cumulative error of an axis (treated as a performance criterion and calculated as the integral of the absolute value of the rotor speed,  $Cumulative\ error = \int_0^{t_e} |\dot{\theta}(t)| dt$ , where  $t_e$  is the evaluation time interval) for a broad range of sampling rates or bandwidth allocations, from  $h = 0.05 \dots 0.5$ s. As it can be seen in Figure 3, the performance criterion of the DC motor can be approximated by a linear function. In addition, the extraction of the parameter  $\alpha$  that characterizes  $p_i$  is illustrated.

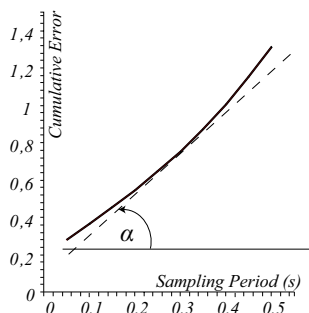


Fig. 3. Sampling periods (i.e. portions of bandwidth) vs. Error

To provide a direct comparison with traditional control system implementations, a baseline policy has been implemented, called standard, in which all control loops always share the available bandwidth equally and no dynamic resource allocation is used. The standard policy implements the “traditional” static controller and is used for examining the overall performance benefit of the optimal policy.

The control performance improvement of the optimal policy was evaluated by looking at the total cumulative error of 200 axes (i.e.,  $Cumulative\ error = \int_0^{t_e} \sum_{i=1}^{200} |\dot{\theta}(t)| dt$ , where  $t_e$  is the time each experiment lasts). Figure 4 shows the performance improvement in terms of accumulated error against the baseline, static policy. From the figure it can be seen that the optimal policy achieves better performance than the static, corroborating the theoretical optimal solution.

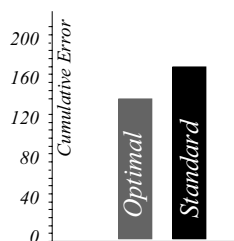


Fig. 4. Performance improvement of the optimal policy relative to the standard policy

Although in these simulations all axes have been anal-

ysed independently, the optimization approach admits expressing the kinematics and dynamics dependences among axes. This can be done by appropriately setting the weights in Equation (13), either statically (prior run-time) or dynamically (at run-time, by the master controller).

## V. CONCLUSIONS

Careful bandwidth management is the key to providing the best possible performance in highly articulated robots whose control strategy is build upon a networked control architecture. A feedback-based management model for concurrently executing networked axis controllers in a communication constrained environment that allows the system to allocate bandwidth as a function of the state of the robot axes has been presented. Based on this model, an optimal policy has been introduced such as overall control axes performance is improved. Simulation results have shown that the optimal policy outperforms the static (open-loop) policy that is traditionally used in such architectures, incurring in almost 10% less error.

## REFERENCES

- [1] Y. Zhang, M. Yim, C. Eldershaw, D. Duff and K. Roufas, "Scalable and reconfigurable configurations and locomotion gaits for chain-type modular reconfigurable robots," IEEE Symposium on Computational Intelligence in Robotics and Automation, Japan, 2003.
- [2] Z. Butler and D. Rus, "Distributed Motion Planning for Modular Robots with Unit-Compressible Modules," International Journal of Robotics Research, 22(9), 699-716, Sept. 2003.
- [3] A. Ray, "Introduction to Networking for integrated control systems," IEEE Control System Magazine, vol.9, pp. 76-79, January 1989.
- [4] Modular Robotics and Robot Locomotion. Robotics Research Center Nanyang Technological University. Web page <http://155.69.254.10/users/risc/www/mod-intro.html>.
- [5] Modular Reconfigurable Robotics. Palo Alto Research Center Incorporated. Web Page <http://www2.parc.com/spl/projects/modrobots/>
- [6] G. Buttazzo, Hard Real-Time Computing Systems, Kluwer Academic Publishers, 1997.
- [7] J. Yépez, P. Martí and J. M. Fuertes, "Control Loop Performance Analysis over Networked Control Systems," in Proc. IEEE 28th Annual Conference of the Industrial Electronics Society, Sevilla, Spain, Nov.5-8, vol. 4, 2002, pp.2880-2885.
- [8] P. Martí, G. Fohler, K. Ramamritham, and J.M. Fuertes, "Improving Quality-of-Control using Flexible Timing Constraints: Metric and Scheduling Issues," Proc. of the 23rd IEEE Real-Time System Symposium, Austin, TX, USA, December 2002.
- [9] E. K. P. Chong and S. H. Zak, An Introduction to Optimization, John Wiley and Sons, Inc, 1996.
- [10] G. Buttazzo and L. Abeni, "Adaptive Workload Management through Elastic Scheduling," Real-Time Systems, Vol. 23, No. 1-2, pp. 7-24, July-September 2002.
- [11] S. A. Brandt, S. Banachowski, C. Lin and Timothy Bisson, "Dynamic Integrated Scheduling of Hard Real-Time, Soft Real-Time and Non-Real-Time Processes," IEEE Real-Time Systems Symposium, Cancun, Mexico, December 2003, pp. 396-407.
- [12] K. J. Astrom and B. Wittenmark, "Computer-controlled systems. Third Edition", 1997.
- [13] A. Cervin, J. Eker, B. Bernhardsson and K.-E. Årzén, "Feedback-Feedforward Scheduling of Control Tasks," Journal of Real-Time Systems, vol. 23, pp. 25-53, 2002.
- [14] B. Wittenmark and K. J. Åström, "Simple Self-tuning Controllers," Unbehauen, Ed. Methods and Applications in Adaptive Control, Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin, F.R.G. vol. 24, pp. 21-29, 1980.
- [15] M. Dögruel and U. Zgner, "Stability of a Set of Matrices: A Control Theoretic Approach", in 34th Conference of Decision and Control, September 1995.
- [16] W. Gellert, S. Gottwald and M. Hellwich, The VNR Concise Encyclopedia of Mathematics, Van Nostrand Reinhold Company, 1988.