Hierarchical Regulation of Sensor Data Transmission for Networked Telerobots

Ángel Martínez-Tenor[†], Ana Gago-Benítez[†], Juan-Antonio Fernández-Madrigal[†], Ana Cruz-Martín[†], Rafael Asenjo[‡] and Ángeles Navarro[‡]
[†] Department of Systems Engineering and Automation
[‡] Department of Computer Architecture
Universidad de Málaga, Andalucía Tech
Málaga, Spain
Email: {angelmtenor,anagagobenitez}@gmail.com,
{jafma,anacm}@ctima.uma.es,
{asenjo,angeles}@ac.uma.es

***Abstract*—Networked telerobots carry sensors that send data, with stochastic transmission times, to a remote human operator, who must execute some real-time control task (e.g., navigation). In this paper we propose to regulate the sensory information being transmitted in order to guarantee soft real-time requirements and also optimize the quality of control, through a novel two-level hierarchical controller that both varies the amount of transmitted sensor data and dynamically reconfigures active sensors. Our controller has been implemented in a web-based teleoperator interface that is highly portable (it runs on desktop PCs, tablets, smartphones, etc.) and non-invasive, i.e., requires minimal modifications in the existing components of the system, thus being suitable for many applications. Here we present our regulation methods and the results of some experiments. They demonstrate the maximization of the transmitted data while guaranteeing the real-time requirements.**

Keywords—networked sensors, real-time communications, user interfaces.

I. INTRODUCTION

Sensors are increasingly connected to general-purpose networks, computers and operating systems. Applications based on repeatedly requesting data from them are pervasive today, e.g., robot teleoperation [1], multimedia broadcasting [2], surveillance [3], etc. In these systems, built up with off-the-shelf general-purpose components, neither the software nor the hardware provide hard real-time guarantees in the transmission times, or any mean for easy modification of the inner communication protocols or application software in order to improve those times. However, real-time guarantees may be needed for the sake of dependability, typically when control tasks must be carried out based on the received sensory data.

In many applications, the reception of less data only degrades the system performance gracefully, e.g., in telepresence, a camera may reduce the quality of the images for improving transmission times. In that case, a feasible real-time requirement would be to transmit, in each sensor polling iteration, the largest amount of data that guarantees a total transmission time (from the moment that the request is issued to the moment the next request is) equal to or shorter than a given one, τ, with a probability equal to or greater than a given one, Π. This paper deals with such soft real-time requirements.

Networked telerobots [1] are an excellent test-bed for studying that problem. They are typically mobile platforms that carry several sensors for transmitting data to a remote human operator for performing some task (e.g., surveillance, navigation, etc.). The components of the system do not allow for important modifications; besides, although it is desirable to transmit as much information as possible, the operator is able to drive the robot even when less data are received.

In previous works we have analyzed the delays existing in the transmission of sensory data from networked telerobots, analyzing some probabilistic models [4]. In this paper we use such a model as a predictor of the transmission times, and build upon it a two-level hierarchical regulation system able to automatically vary the amount of sensory data transmitted and the set of sensors that are transmitting at each time, in order to satisfy as optimally as possible the soft real-time and quality control requirements of the application. In literature, methods that regulate transmission times have been reported previously, but either not in such a combination, or for the task of teleoperation (e.g., [5], [6]). We have implemented our hierarchical controller in a web-based graphical interface for the teleoperation of robots through the Internet. Our results show the better performance of these combined methods with respect both to their separate use and to not having a controller.

The structure of the paper is as follows. In Section II we describe the components of our hierarchical controller system. In Section III we present the experiments and the results obtained. We end with some conclusions and future work.

II. OVERVIEW OF THE REGULATION SYSTEM

A. Prediction of transmission time probabilities

The inner component of our approach is a method for predicting the probability $\pi_i(\delta)$ of completing the transmission

This work has been supported by the Junta de Andalucía regional government and the European Regional Development Funds through the research projects P08-TIC-04282, P08-TIC-3500 and P11-TIC-08144.

of a given amount of data δ from the *i*-th sensor before a given time τ_i. That prediction will be used to select a value of δ that makes the actual probability of such transmission equal to or greater than a pre-defined one for that sensor, Π_i.

In a previous work [7] we have presented a powerful probabilistic model of the transmission times that considers the delays of the network and of any other software or hardware component in the transmission path. It is based on a three-parametrical log-logistic distribution —its cumulative function will provide $\pi_i(\delta)$— and on novel algorithms to estimate and assess the parameters of that distribution. In particular, here we implement in the robot on-board computer the *stateless* modeling algorithm originally reported in [8]. Unfortunately, the code in charge of estimating and assessing the parameters of the log-logistic requires numerical optimizations, which can be time and energy consuming. For addressing that problem we have used pipelining parallelization techniques compatible with the multi-core CPUs typically available in off-the-shelf computers for networked telerobots. More precisely, we leverage the Intel Threading Building Blocks (TBB) library [9], obtaining an efficient implementation, both in terms of speed-up and energy saving, even for this very fine-grained application. It has been tested on two quad-core systems: a low power Intel i7 and an ultra-low power Samsung Exynos 5. Our tests conclude that no more than 2 cores are needed (see Fig. 1), being the rest free for other parts of our algorithms [10].

B. *Fine regulation: variation of the amount of sensory data*

The first level of our hierarchical regulation system consists of a regulation algorithm for each sensor *i* in charge of deciding which amount of sensory data is requested at the next polling iteration if we want to complete that transmission in τ_i or less time with a probability Π_i or greater. Every sensor has a finite number of densities (possible amounts of requested data[1]), and we maintain, for each density, a log-logistic model.

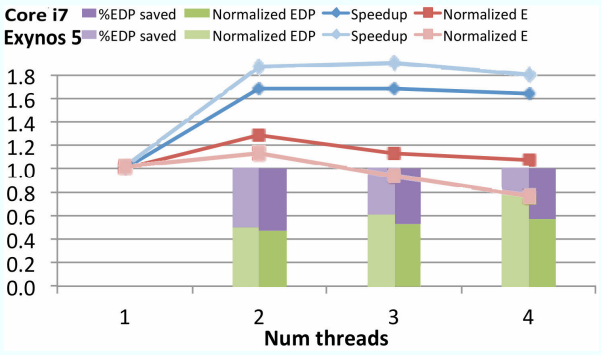

Fig. 1. Analysis of the optimal number of threads for the execution, in a multi-core CPU, of the algorithm that estimates the log-logistic model of transmission times. "Speedup" is the sequential execution time divided by the parallel execution time. EDP stands for "Energy Delay Product", and is the product of execution time in seconds times the energy consumed in Joules (the lower the better). E stands for "Energy improvement"; it is the energy measured on one core divided by the energy consumed by the corresponding number of cores (the higher the better).

[1] Without loss of generality, we consider a density as a natural number, being 0 the smallest amount of sensory data (e.g., the minimum resolution of a camera sensor) and D_i-1 the maximum, thus having D_i densities for sensor *i*.

Our density regulation algorithm is a heuristic one, similar to a classical proportional controller but with a varying constant, that tries to keep the probability of achieving the desired times τ_i close to Π_i and, simultaneously, to increase the density as much as possible. It has as input the density δ_0 that was requested in the last polling iteration, and outputs the density δ to be requested in the next. If there is not enough information in the log-logistic model for predicting $\pi_i(\delta_0)$, it outputs $\delta=\delta_0$ —which leads to eventually accumulate enough data for such a prediction; otherwise, it may happen that: i) $\pi_i(\delta_0) < \Pi_i$, ii) $\pi_i(\delta_0) > \Pi_i+\alpha$ or iii) $\pi_i(\delta_0) \in [\Pi_i , \Pi_i+\alpha]$, where $\alpha \in (0,1)$ is a small number (we use $\alpha=0.05$). In case i) the goal is not being satisfied, thus the algorithm should decrease the density, as long as $\delta_0>0$, for reducing transmission times; in case ii) the soft real-time requirement for that sensor is being satisfied, and thus it can try to increase the density, as long as $\delta_0<D_i-1$, seeking to improve operator's quality of control; finally, in case iii) we are performing well but very close to not doing so, thus it keeps the current density (it outputs $\delta=\delta_0$).

In cases i) and ii), a change of density Δδ is computed that is proportional to the "error" in probability $e=\Pi_i - \pi_i(\delta_0)$. More concretely, in case i) we change to a new density $\delta=\min(\delta_0-1, \delta_0(1-e))$. If δ=0 or if we do not have an estimated log-logistic for predicting $\pi_i(\delta)$, or if $\pi_i(\delta) \geq \Pi_i$, we return that δ; otherwise, we repeat the jump to lower densities but now starting at $\delta_0=\delta$. In case ii) we firstly set a change size $\Delta\delta=-(D_i-1-\delta_0)e$ and then calculate the new density as $\delta=\delta_0+\Delta\delta$. If Δδ=0 or if we do not have a prediction $\pi_i(\delta)$, or if $\pi_i(\delta) \geq \Pi_i$, we return that δ; otherwise, we do $\Delta\delta=\Delta\delta-1$ and repeat with $\delta=\delta_0+\Delta\delta$.

C. *Coarse regulation: activation/deactivation of sensors*

The second level in our hierarchical controller deals with several sensors at once. Each one may be regulated as described in section II.B or not, but it must have the possibility of being paused —issuing no more requests to it temporarily— or re-started; it will be paused whenever it does not satisfy its τ_i for a given time (a number of consecutive requests).

In this level we must define $n>1$ combinations of the available sensors, called *executives*, to perform the same task (e.g., robot teleoperation) with different timing requirements. For instance, a mobile robot that is teleoperated for navigation can work by sending data from a camera or from a range laser scanner, or from both sensors but less frequently, thus defining $n=3$ executives. The *j*-th executive is formally a tuple (S_j,P_j,F_j,A_j,E_j), where S_j is its set of sensors and the rest are sets that contain one value per sensor of S_j: P_j contains the pairs (τ_i, Π_i), F_j contains the maximum times allowed for not satisfying τ_i before pausing the sensor, A_j contains extra times that are added to the ones of F_j whenever the sensor is re-started, in order to bypass possible transients, and E_j contains flags that indicate whether the sensor is essential (i.e., it should not be deactivated as long as possible). We also need an ordering $O=<x_1,x_2,\ldots,x_n>$ of the executives, from the most convenient for solving the task (x_1) to the least (x_n). At any time, only one will be enabled, denoted by $x_{current}$; initially $x_{current}=x_1$.

Several events in the system trigger the execution of our coarse regulation algorithm: i) when a sensor of $x_{current}$ is paused; ii) every given constant time T_1, for checking if all sensors are transmitting; iii) every another constant time T_2 ($>T_1$), for trying to re-start previously paused sensors. These times are not critical. In case i), if the paused sensor is essential, and there is some other non-essential sensor that is transmitting, the latter is paused to facilitate the fulfillment of the requirements of the former in the future, while if there is no other non-essential sensor transmitting, the next worse executive, if any, is activated, i.e., $x_{current}=x_{current+1}$. In case ii), if all the sensors are transmitting, we change to the next better executive, if any, i.e., $x_{current}=x_{current-1}$. Finally, in case iii), if there is any paused sensor, it is re-started to try to recover it.

III. Evaluation and Experiments

Firstly we have evaluated the performance of the regulation algorithm described in section II.B separately to extract statistical conclusions (section III.A). Then, we have implemented a web-based graphical interface, designed for a human teleoperator to drive mobile robots remotely, that implements all our algorithms and collects information about the behavior of the system (section III.B).

A. Experiments with the fine regulation algorithm only

We have used in these experiments real data gathered from a webcam connected to the Internet that is requested repeatedly at different resolutions. Webcams are the most suitable sensor for this purpose because their number of densities is high (D=41 in our experiments, each one corresponding to a resolution of RGB images, from 160x120 to 640x480). They are also very common sensors in mobile robots today.

Fig. 2 shows the obtained results. Every colored point in that plane is the result of an experiment consisting in requesting the sensor a number of times. The abscissa of the point is the average density requested, weighted by the time during which each density was active, and measured as the number of bytes transmitted. The ordinate is the average percentage of time during which the transmission times were shorter than or equal to τ_1 (τ_1=0.3 secs. here). The thick dashed black line is the probability requirement Π_1 (90% here): points above or in that line satisfy the real-time pair (τ_1, Π_1), being the corresponding experiments better as the abscissa is greater.

The red dots are experiments, one per density, consisting each one of 1951 requests all with that density. Only small densities (corresponding to RGB 280x210 or less) satisfy (τ_1, Π_1). Note that we cannot guess which densities are those before doing this exhaustive exploration —and even then that would only be valid for the particular situation of network, software load, etc.—, thus choosing for example a constant density of RGB 280x210 is not a valid solution in a real scenario with this sensor. Also note that any regulation algorithm will achieve average densities and probabilities close to the red points, since all possible densities are there.

The figure also shows points corresponding to other four experiments: one that selected a uniformly random density at each request (for 972 requests), one that selected all densities in order, cyclically (1017), one with the regulator of section II-B but using a simple Bernoulli[2] predictor of probabilities (1379), and one with the same regulator and the log-logistic predictor (1705). All the experiments have been done with transmission times that statistically behave as the real ones used for the red curve. As shown, our regulation algorithm satisfies the real-time requirement, but more importantly, when the log-logistic predictor is used, requested densities never violate the real-time requirement, which happens if we use the Bernoulli predictor.

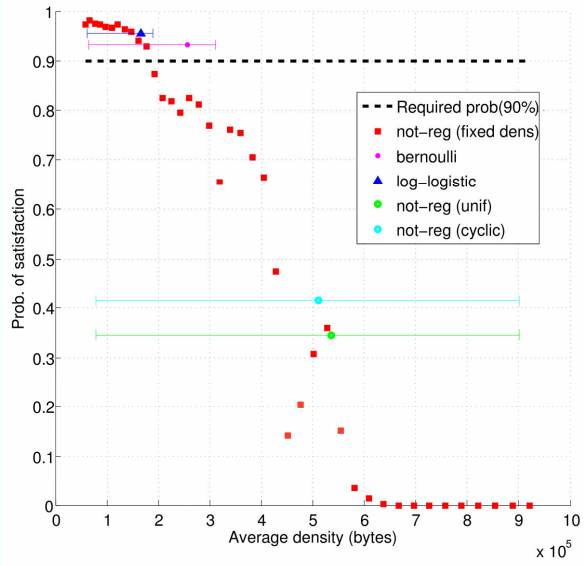

Fig. 2. Study of the regulation algorithm of section II-B with two modeling methods and w.r.t. setting a constant density in every request from a remote webcam. The right-most red square point above the probability requirement line of 90% corresponds to a webcam resolution of RGB 280x210 pixels. Horizontal ranges indicate explored densities.

B. Experiments with the complete hierarchical controller

Finally, we have implemented a highly re-configurable, web-based interface, for remotely controlling networked telerobots, that displays sensory data and implements our algorithms in a portable (it runs on desktop PCs, tablets, smartphones, etc.) and not invasive way (Fig. 3-Left/Middle). The coarse level regulator has been provided with 5 executives, all with two essential sensory widgets (frontal laser and odometer) and two non-essential ones (rear laser and frontal camera), but with different parameters to differentiate the executive with the strongest real-time requirements (x_1). We set T_1=5 secs. and T_2=10 secs.

For assessing the quality of control we have defined experiments consisting in driving, with a standard joystick, a simulated robot along a track (Fig. 3-Right). The interface automatically gathers the time spent in completing the lap, λ, and the fraction of λ that the robot has been within its limits, $o \in [0,1]$. Then it computes a measure of the quality of control,

[2] A Bernoulli model estimates $\pi_i(\delta)$ as the proportion of requests that have completed in τ_i or less time divided by the total number of requests. It does not represent well the right tail of the actual distribution, being optimistic.

the effective time λ_{eff}, by adding to λ a penalty for the time spent outside the track limits: $\lambda_{eff}=\lambda+10(1-o)\lambda$.

We have conducted 6 kinds of experiments, all with the same user, external conditions and deployment of the components of the system, in order to isolate as much as possible the effects of our algorithms from other factors: A) using the complete hierarchical controller system, B) like A but with a worse density regulator: a classical proportional controller[3], C) using only the fine regulation level, D) like C but with the classical proportional controller instead of the one of section II-B, E) without any regulation, fixing the density to the smallest one, F) like E but with the largest density. We have repeated each experiment 50 times and collected the average λ_{eff} in Table I. It is evident the advantage, on average, of using the complete controller system (A), and also the better behavior of our fine regulation level with respect to other controllers (A *vs.* B, C *vs.* D), even when no coarse regulation is used (C *vs.* D). Furthermore, we obtained better results with the complete system (A) than by fixing the density to the smallest one (E, which produces the best transmission times but minimizes the amount of information for the user).

TABLE I. QUALITY OF CONTROL (AVERAGE λ_{eff}, MSECS.)

A	*B*	*C*	*D*	*E*	*F*
55	66.9	58.4	69.1	57.5	114.7

IV. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a hierarchical controller that maximizes the information that is transmitted from remote sensors while fulfilling soft real-time requirements. It consists of two levels, one that adapts the amount of data transmitted by each sensor separately and the other that selects which sensors should transmit. We have implemented it in a web-based interface that is portable, non-invasive and highly configurable. Our results show the good performance of this combined controller according both to the fulfillment of the real-time requirements and to the quality of control.

In the future we plan to conduct more comparative studies, with a wide diversity of algorithms and also scenarios, isolating the effects of other factors such as human expertise, network congestion, computational power, etc. The final goal is to build up a system that optimizes the real-time behavior and quality of the tasks performed remotely.

REFERENCES

[1] B. Siciliano, O. Khatib, Springer Handbook of Robotics. Springer, Berlin, Germany, 2008

[2] X. Wang, H. Schulzrinne, "Comparison of adaptive internet multimedia applications", IEICE Trans. Commun. E82-B 1999, pp. 806–818.

[3] F. Porikli, F. Bremond, S.L. Dockstader, J. Ferryman, A. Hoogs, B.C. Lovell, S. Pankanti, B. Rinner, P. Tu, P.L. Venetianer, "Video surveillance: past, present, and now the future [DSP Forum]", IEEE Signal Processing Magazine, 30(3), 2013, pp. 190 – 198.

[4] A. Gago-Benítez, J.-A. Fernández-Madrigal, A. Cruz-Martín, "Marginal Probabilistic Modeling of the Delays in the Sensory Data Transmission of Networked Telerobots", Sensors 14(2), 2014, pp. 2305-2349.

[5] R. Oboe, "Web-Interfaced, Force-Reflecting Teleoperation Systems", IEEE Transactions on Industrial Electronics, 48(6), 2001.

[6] J. Wu, Q-S. Jia, K. H. Johansson, L. Shi, "Event-Based Sensor Data Scheduling: Trade-Off Between Communication Rate and Estimation Quality", IEEE Transactions on Automatic Control, 58(4), 2013.

[7] A. Gago-Benítez, J.-A. Fernández-Madrigal, A. Cruz-Martín, "Log-logistic modelling of sensory flow delays in networked telerobots", IEEE Sensors, vol. 13, no. 8, 2013.

[8] A. Gago-Benítez, J.-A. Fernández-Madrigal, A. Cruz-Martín, "Log-logistic modeling of sensory flow delays in networked telerobots", IEEE Sensors, Taipei (Taiwan), 2012.

[9] TBB 2014: Intel Corporation, "Threading building blocks," [web page] http://www.threadingbuildingblocks.org/, [Accessed on 10 Jul. 2014].

[10] R. Asenjo-Plaza, A. Navarro, J.-A. Fernández-Madrigal, A. Cruz-Martín, "On the parallelization of a three-parametric log-logistic estimation algorithm", Jornadas de Computación Empotrada, Valladolid (Spain), 2014.

Fig. 3. Left and Middle) Our interface in a real remote driving experiment, running in three different platforms: a desktop PC, a tablet and a smartphone. Right-top) Simulated track (corresponding to one of our actual labs) where the experiments of this paper have been conducted. Right-bottom) One moment of the experiments where the robot goes out of the limits of the track, which is indicated to the user changing the color of the laser scanner sensor data.

[3] The next density is computed as a constant (1.0 in our case) times the current error in probability, $\Pi_i - \pi_i(\delta_0)$, rounding to the nearest integer.