Scheduling strategy for Real-Time Distributed Systems

A. Menéndez-Leonel de Cervantes *[1], H. Benítez-Pérez [2]

[1,2] Departamento de Ingeniería de Sistemas Computacionales y
Automatización, IIMAS, Universidad Nacional Autónoma de México
Apdo. Postal 20-726. Del. A. Obregón, México D.F. CP. 01000, México.
Fax: ++52 55 5616 01 76, Tel: ++52 55 5622 36 39
*amenendezl@uxmcc2.iimas.unam.mx

ABSTRACT
The objective of this paper is to show the use of a global Scheduling Strategy based on the analysis of a Real-Time Distributed System. The use of a Networked Controlled System as a case study shows that the performance of the system depends not only on the sampling periods of its individual components, but also on the time dispersion amongst these periods. It is also shown that it is possible to have a schedulable but unstable RTDS.

Keywords: real-time, scheduler, distributed system, network control system.

RESUMEN
El objetivo de este artículo es presentar el uso de una "*Estrategia de planificación*" global basada en el análisis de un Sistema Distribuido en Tiempo-Real (RTDS). El uso de un sistema controlado por red (NCS) como caso de estudio que consiste de un helicóptero de dos rotores, muestra que el desempeño del RTDS depende no solamente de los periodos de muestreo de sus componentes individuales, sino también de las relaciones temporales entre ellos. También se muestra que es posible tener un RTDS planificadle pero inestable.

1. Introduction

It has been stated that an acceptable performance of a Networked Controlled System (NCS) depends on the sampling periods of its individual components [1]. Even though this is generally true, for Real-Time Distributed Systems (RTDS), the area of acceptable performance tends to be smaller and not precisely continuous. One of the reasons for this behavior is that the relationships amongst the periods of the RTDS components (i.e. nodes and network) have a high degree impact on the overall system performance.

It is possible to have an RTDS in which the Real-Time constraints of the tasks at every node are fulfilled, meaning that each task is scheduled in such a way that meets its deadline. In these cases the scheduling work is considerable, the system is very inflexible and there is no guarantee whatsoever that the system is going to remain stable.

The objective of this paper is to present a scheduling strategy and an analysis of the behavior and limitations of a Real-Time Distributed System (RTDS) using it.

In order to prove this statement, a case study of an RTDS acting as a Networked Controlled System (NCS) has been implemented. The case study consists of a "Quanser 2 DOF Helicopter" which is a helicopter model mounted on a fixed base with two propellers that are driven by DC motors. The front propeller controls the elevation of the helicopter nose about the pitch axis and the back propeller controls the side-to-side motions of the helicopter about the yaw axis. The pitch and yaw angles are measured using high-resolution encoders[2]. The case study also consists of a software simulator developed using Matlab, Simulink and TrueTime.

The rest of the paper has the following sections: Section 2 introduces the proposed scheduling strategy, in Section 3 the NCS case study and its implementation as an RTDS using the proposed strategy are presented. The obtained results are in Section 4 and, finally, the conclusions and future line of work are in Section 5.

2. Scheduling Strategy

The common RTDS implementation consists in several nodes and a communications network, probably running each one as an independent Real-Time component. The proposal is to have a common scheduling strategy that eventually allows evaluating the performance of an RTDS as an NCS.

The proposal is so determine a "base" period, which allows determining the operational periods of all the nodes participating in the distributed system.

Even though all the nodes can use this "base" period as their operational period, the proposal includes the option of having different periods for each node by means of the base period and a dispersion factor.

In the case study presented in the next section, the "base" period is the operational period for the controller periodic task in the corresponding controller node.

The "base" period and a so-called dispersion factor are used to calculate the actual sampling periods of the four sensor nodes according to the following equations:

$$\begin{aligned} sensor1 &= base \times (1 + dispersion) \\ sensor2 &= base \times (1 - dispersion) \\ sensor3 &= base \times (1 + (dispersion \times 1.1)) \\ sensor4 &= base \times (1 - (dispersion \times 1.1)) \end{aligned} \quad (1)$$

where *sensorN* = period for sensor N, *base* is the "base" period and 0 β *dispersion* β 0.2 is the dispersion factor.

The value of the dispersion factor means that in the tightest case, when *dispersion*=0, the four sensors have the same period (i.e. "base" times 1 + 0) as the controller. On the other extreme, when "base"=0.020, the sensors have a period 20% or 22% above or below the "base" period. Dispersion greater than 25% causes the system to have an unacceptable performance.

As part of the scheduling strategy, the RTDS scheduler node allocates a bandwidth share to every node by means of assigning a time-window when to transmit, independently from the network protocol used, which must be considered solely as the network access controller.

3. Implementation details

The case study consists of two major components, a helicopter physical and mathematical model (Subsection 3.1) and a simulated Real-Time distributed system implemented in *Matlab* with *TrueTime*[5] (Subsection 3.2).

3.1 Helicopter

While the description of the helicopter is beyond the scope of this paper, a brief description of it is provided in this subsection, for further information about it, the original documentation in [2] is recommended. The Quanser 2 DOF Helicopter experiment consists of a helicopter model mounted on a fixed base with two propellers that are driven by DC motors. The front propeller controls the elevation of the helicopter nose about the pitch axis and the back propeller controls the side to side motions of the helicopter about the yaw axis. The pitch and yaw angles are measured using high-resolution encoders[2]. The two degrees of freedom helicopter pivots about the pitch axis by angle θ and about the yaw axis by angle ψ. The pitch is defined positive when the nose of the helicopter goes up and the yaw is defined positive for a clockwise rotation. The following table (Table I) lists various lengths, masses and moment of inertias associated with the helicopter model.

A FF+LQR+I (Feed Forward + Linear Quadratic Regulator + Integrator) controller implemented in *Matlab* and *Simulink* (Figure 1) is used in the helicopter model. The FF+LQR+I regulates the pitch axis of the helicopter using an integrator in the feedback loop to improve the steady-state error. It uses a feed-forward and a proportional-integral-velocity (PIV) algorithms to regular the pitch and a PIV to control the yaw angle [2].

Variable	Description	Value	Unit
$B_{eq,p}$	Equivalent viscous damping about pitch axis.	0.800	N/V
$B_{eq,y}$	Equivalent viscous damping about yaw axis.	0.318	N/V
$J_{eq,p}$	Total moment of inertia about pitch pivot.	0.0384	$kg \cdot m^2$
$J_{eq,y}$	Total moment of inertia about yaw pivot.	0.0384	$kg \cdot m^2$
K_{pp}	Thrust torque constant acting on pitch axis from pitch motor/propeller.	0.204	N·m/V
K_{py}	Thrust torque constant acting on pitch axis from yaw motor/propeller.	0.0068	N·m/V
K_{yp}	Thrust torque constant acting on yaw axis from pitch motor/propeller.	0.0219	N·m/V
K_{yy}	Thrust torque constant acting on yaw axis from yaw motor/propeller.	0.072	N·m/V
l_{cm}	Center-of-mass length along helicopter body from pitch axis.	0.186	Cm
m_{heli}	Total moving mass of the helicopter	1.3872	kg

Table 1. Helicopter specifications and model parameters[2].

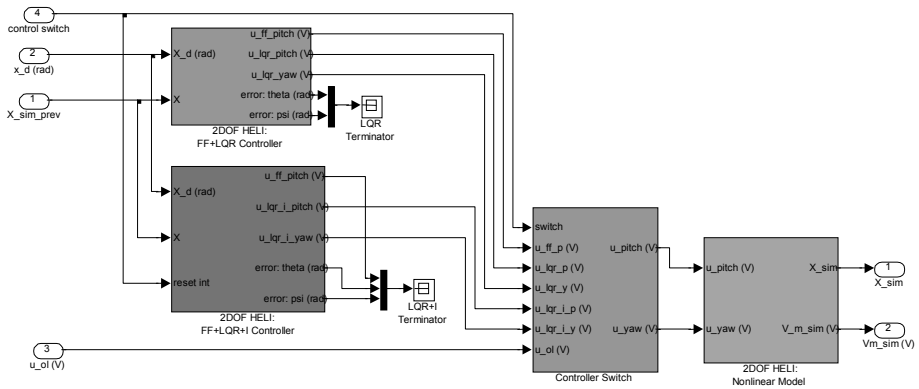

Figure 1. Helicopter Close-Loop system simulation model[2].

The linear state-space model of the helicopter is given by Equation (2), where after linearizing the nonlinear equations of motion about the quiecent point (θ 0=0, ψ0=0, θ˙0=0, ψ˙0=0) and substituting the state x = [θ, ψ, θ˙, ψ˙] is solved for x˙[2].

$$x=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{B_p}{J_{eqp}+m_{heli}l_{cm}^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{B_y}{J_{eqy}+m_{heli}l_{cm}^2} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} x+\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{pp}}{J_{eqp}+m_{heli}l_{cm}^2} & \frac{K_{py}}{J_{eqp}+m_{heli}l_{cm}^2} \\ \frac{K_{yp}}{J_{eqy}+m_{heli}l_{cm}^2} & \frac{K_{yy}}{J_{eqy}+m_{heli}l_{cm}^2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u \quad (2)$$

$$y=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x$$

The FF+LQR+I control that converges *(θ, ψ, θ˙, ψ˙) → (θd, ψd, 0, 0),* where *θd* is the desired pitch angle and *ψd* is the desired yaw angle, is defined in Equation (3), where kff is the feed-forward control gain set to: 1 V/V:

$$\begin{bmatrix} u_p \\ u_y \end{bmatrix}=\begin{bmatrix} k_{ff}\frac{m_{heli}gl_{cm}\cos\theta_d}{k_{pp}} \\ 0 \end{bmatrix}-\begin{bmatrix} k_{11}k_{12}k_{13}k_{14} \\ k_{21}k_{22}k_{23}k_{24} \end{bmatrix}\begin{bmatrix} \theta-\theta_d \\ \psi-\psi_d \\ . \\ \theta \\ . \\ \psi \end{bmatrix}- \quad (3)$$

$$\begin{bmatrix} \int_{t_0}^{t_f} k_{15}(\theta-\theta_d)+\int_{t_o}^{t_f} k_{16}(\psi-\psi_d) \\ \int_{t_0}^{t_f} k_{25}(\theta-\theta_d)+\int_{t_o}^{t_f} k_{26}(\psi-\psi_d) \end{bmatrix}$$

this FF+LQR+I equation uses the integral values of the errors in the feedback loop to improve the steady-state error, with the following control gain according to the Matlab model and the manual [2]:

$$k=\begin{bmatrix} k_{11} \ldots k_{16} \\ k_{21} \ldots k_{26} \end{bmatrix}=\begin{bmatrix} 18.9 & 1.98 & 7.48 & 1.53 & 7.03 & 0.770 \\ -2.22 & 19.4 & -0.450 & 11.9 & -0.770 & 7.03 \end{bmatrix}$$

These control outputs (up and uy) correspond to the respective “pitch” and “yaw” motor voltages. The detailed description of de model can be found in [2].

3.2 RTDS

The RTDS implementation consists of eight nodes, being everyone an independent Real-Time kernel. Four of these nodes are sensor nodes, 2 nodes are actuators, one is the controller node and the eighth node is the scheduler node. As it can be seen in Figure 2, the eight nodes that build the RTDS communicate between them trough a Real-Time “CAN” network. The model is implemented in Matlab using the tool *TrueTime* from Lund University[5], to simulate the Real-Time kernel of each node and the network. The outputs of the system can be connected to both, the simulated helicopter model or the actual helicopter, being therefore capable of calculating or actually “seeing” the RTDS performance.

The first node in the model (leftmost node in Figure 2) is the controller node, which uses the values provided by the sensors and Equation (2) to calculate the control outputs (u_p *and* u_y) that correspond to the respective “pitch” and “yaw” motor voltages.

The four sensor nodes are correspondingly: theta, psi, theta-dot and psi-dot from equation (2).

The two actuators (2 lower right nodes in Figure 2) have the control outputs u_p and u_y been the “pitch” and “yaw” motor voltages.

The last node is the scheduler node (rightmost node in Figure 2) which, as its name implies, schedules the RTDS activities. It is also responsible, as part of the scheduling strategy, to periodically assign network bandwidth to the nodes and it defines and communicates the other nodes a common operational period named “base” period.

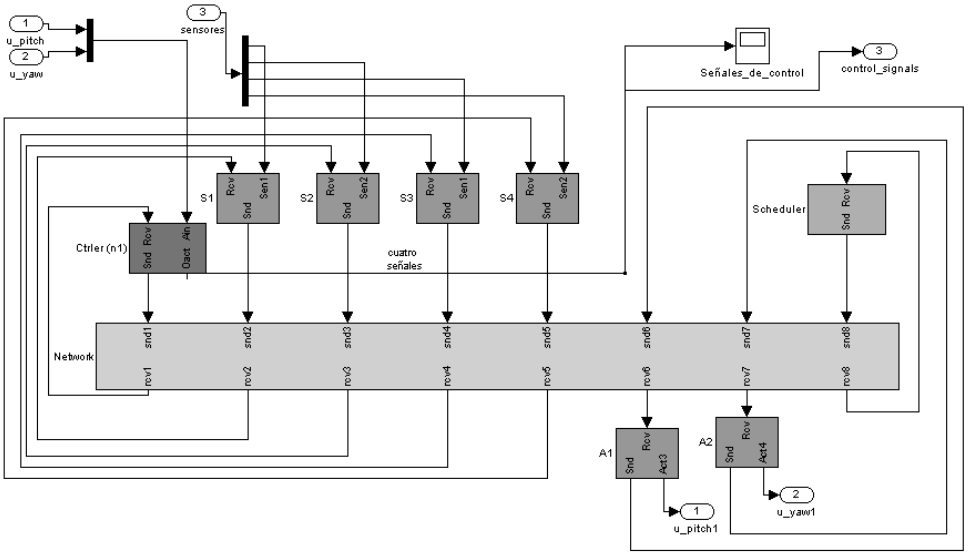

Figure 2. RTDS case study implementation.

In order to have a performance metric of the system, an error of the RTDS is calculated. The RTDS error is the difference between the obtained or real angle θ and desired angle θd (Figure 3), this error is added during the whole simulation as expressed by Equation (4):

$$quadratic\ error = \sum \frac{(\theta_t - \theta d_t)^2}{2} \qquad (4)$$

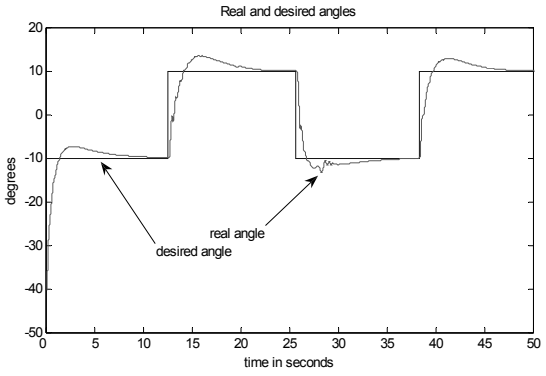

Figure 3. Real angle θ and desired angle θd.

The logarithmic chart in Figure 4 shows the *quadratic error* graphed against the "base" period. The dotted horizontal line marks the level of acceptable and unacceptable system performance. As it can be seen in this chart the function appears not to have a continuous slope as it is expected from a NCS, nevertheless the points of over sampling and under sampling periods (0.001 and 0.027 seconds respectively) can be located.

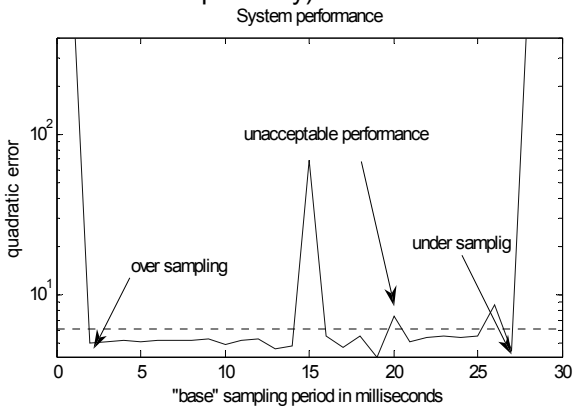

Figure 4. System performance. Sampling period (in 10^{-3} seconds) against logarithmic quadratic error.

The next consideration is the bandwidth allocated to each node of the RTDS. When all the nodes are competing to obtain the network and get bandwidth according to the network access control algorithm (i.e. CAN network priority protocol), the system trends to an unstable state as it can be inferred from Figure 5, in which the error increases trough time (i.e. The difference amongst real angle θ and desired angle θd). In this case study, the scheduler node avoids this problem by assigning each node a time window when to transmit.

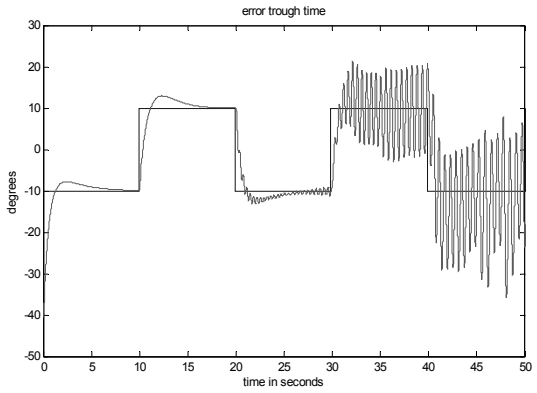

Figure 5. Error trough time (Difference amongst real angle θ and desired angle θd).

4. Results

A series of tests were conducted, having the following characteristics: the "base" period runs above 0.001 and below 0.027 seconds (the points of over and under sampling in Figure 4) with increments of 0.001 seconds. The dispersion factor runs from 0.0 to 0.2 (i.e. 0% to 20% of "base") with increments of 0.05. The total time to run each test is 50 seconds, which is time enough to determine an unacceptable performance.

The chart in Figure 6 shows the total number of transmissions (i.e. all the transmissions made by all the nodes during the test) graphed against the different "base" periods and percentages of dispersion. It can be seen that the total number of transmissions depends, as it was expected, on the "base" sampling period, being evenly distributed amongst the dispersion factors.

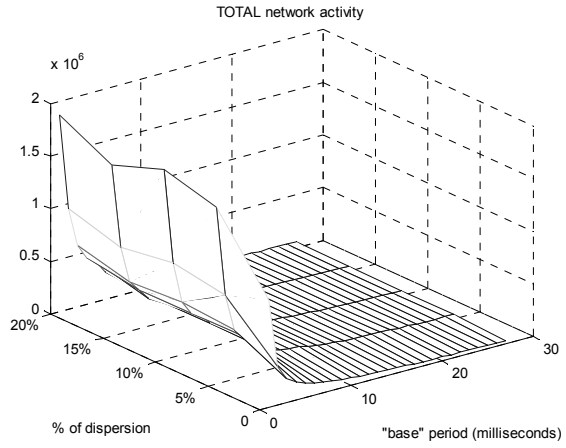

Figure 6. Total number of transmissions varying the dispersion and the "base" period.

As the total network activity, the network activity per node (i.e. number of transmissions per node) depends basically on the "base" period. As can be seen in Figure 7, the number of transmissions is inversely proportional to the "base" period, since the dispersion factor practically does not have an impact in the network activity per node, it is neglected in the chart.

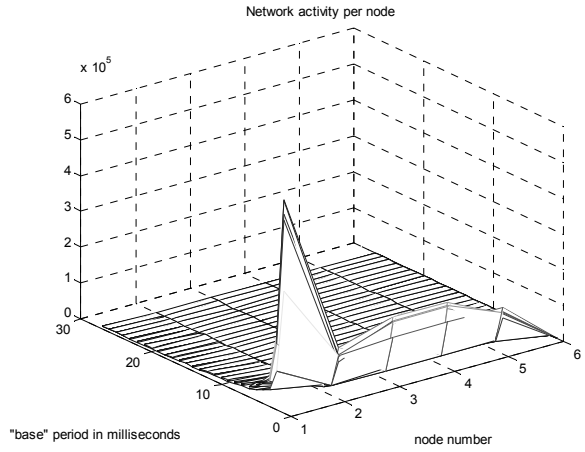

Figure 7. Relation between the total number transmissions per node and the "base" period.

The next graph (Figure 8) shows the accumulated quadratic error of several simulations, of 25 seconds each, in which the base period goes from 0.002 to 0.026 seconds, showing that the acceptable performance of the RTDS depends not only on the sampling rate, but also in the time dependencies along its components.

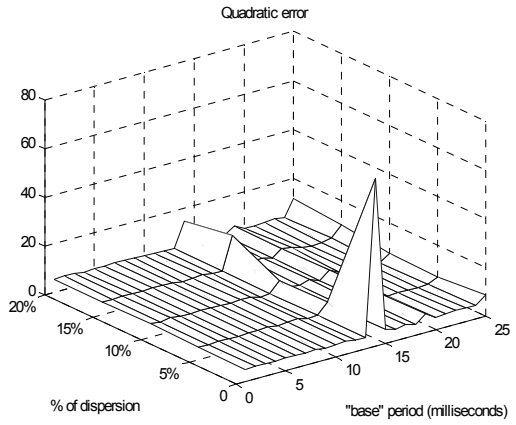

Figure 8. Quadratic error.

In the Quadratic error graph (Figure 8), some periods of operation reveal a higher than acceptable error, for instance when the "base" period is 15 milliseconds and the dispersion factor is 0 the accumulated error for 25 seconds is 80. The Quadratic error graph (Figure 8) and the "System Performance" chart (Figure 4) show that the acceptable performance and the sampling rate of an RTDS do not have a linear relation.

As it can be seen in Figure 8, the quadratic error does not present a smooth slope curve, as it is expected form a NCS, in an RTDS the boundaries for an acceptable performance do not necessarily guarantee it in all the intermediate points, proving that a stable RTDS depends not only on the individual sampling period, but also on the relations of the sampling periods of the nodes conforming the RTDS.

In Figure 9, the behavior of the system using a "base" period of 0.015 seconds can be appreciated, the system is unstable after approximately 75 seconds.

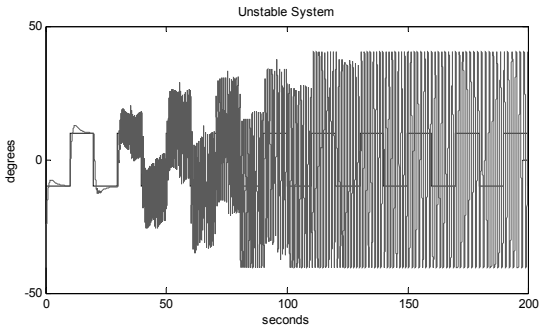

Figure 9. Unstable system
with a "base" period of 0.015 secs.

The RTDS scheduler node allocates a bandwidth share to every node by means of assigning a time-window when to transmit, taken as an heuristic solution, the controller gets 1/3 of the bandwidth, while the rest (2/3) is shared equally amongst the sensors. The Network utilization graph (Figure 10) clearly shows the network is not the constraining resource of the RTDS, since it still has spare capacity to provide. Again, emphasis should be made in the time relations among the systems components.

The graph in Figure 10 shows five horizontal lines which represent the controller (node 1) and the four sensors (nodes 2 to 5) transmissions during a one second sample. Each one of the five lines has three possible states: 0, 0.25 and 0.5, meaning respectively: not transmitting, waiting for the network and transmitting (Figure 11). It is easy to appreciate in the Network utilization graph (Figure 10) that the network throughput is not the restraining component on this RTDS, since most of the time the nodes are not transmitting (i.e. the node state is 0).

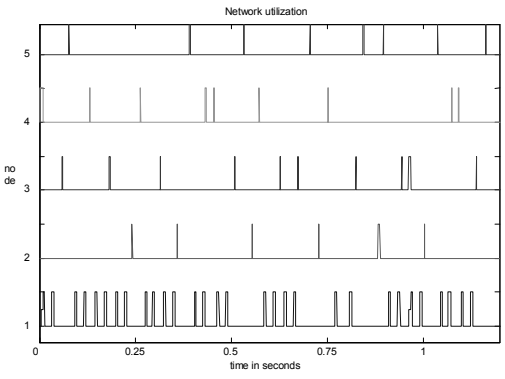

Figure 10. Network utilization.

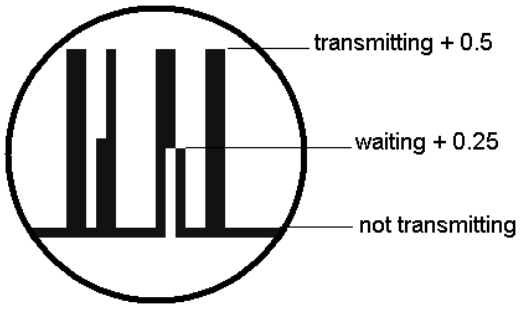

Figure 11. Network utilization zoom, the three states of a node.

In the extreme over-sampling case, the network is used 100% of the time, and in several occasions there are nodes waiting to transmit (Figure 12). This lack of network availability results in deadline losses at all the nodes, causing the information that the sensors or the controller send, if any, to be obsolete. So, as the system has to operate with practically no data, it reaches instability almost immediately.

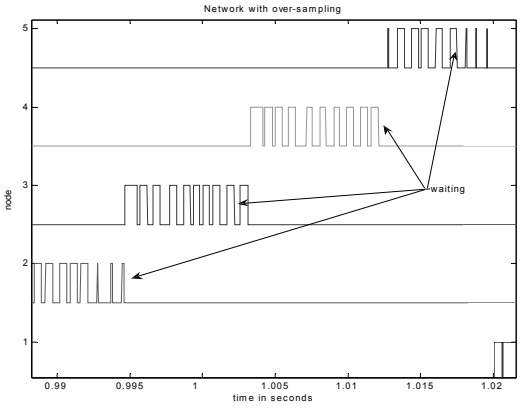

Figure 12. Network utilization with over-sampling

5. Conclusions

Since there is not a global point for validating the timely behavior for the RTDS as a whole, it is possible to have a schedulable (at component level) but unstable RTDS.

A Real-Time NCS depends not only on the operational periods of the nodes, but also in the relationships amongst those periods.

The network throughput is not necessarily the restraining resource of the RTDS, as it has been shown there are some cases in which the RTDS has not an acceptable performance even if there is a good percentage of bandwidth available.

The scheduling strategy proposed here allows evaluating the RTDS performance as a NCS by changing one single (base) parameter for the whole system.

An RTDS scheduler, even one as simple as the one proposed here, serves to increase the acceptable performance sampling range of a NCS. Further analysis needs to be done with the relations of operational or sampling periods of the nodes involved in an RTDS.

The used of scheduling composition[3] or related techniques should be further researched in order to have a distributed Real-Time system scheduler.

Hard Real-Time Distributed Systems have not robust and accurate schedulability analysis algorithms to be validated as dynamic multiprocessor systems [4].

References

[1] Feng-Li Lian, James Moyne, and Dawn Tilbury, “Network Design Consideration for Distributed Control Systems”, IEEE Transactions on Control Systems Technology, Vol. 10, No. 2, march 2002.

[2] “Quanser 2 DOD Helicopter User and Control Manual”, Quanser Inc. rev 1.0, February 10, 2006

[3] Arvind Easwaran, Insik Shin, Oleg Sokolsky, Insup Lee, ”Associative Composition of Hierarchical Real-Time Systems, Technical Report: MS-CIS-06-06”, University of Pennsylvania, Philadelphia, PA, 2006.

[4] Liu W.S. Jane, “ Real-Time Systems”, Prentice Hall, USA, 2000.

[5] M. Ohlin, D. Henriksson, A. Cervin ,“True-Time 1.5 Reference Manual”, Dep. Of Atomatic Control, Lund University, Sweden, 2007.

Authors´ Biographies

Antonio MENENDEZ-LEONEL DE CERVANTES

He is a computer science engineer from the Universidad La Salle, and obtained his PhD at the Universidad Nacional Autónoma de México (UNAM). More than 25 years of experience in the computer and technology industries, leading multi-million projects, as well as in the academic world. Devoted to research for the last years, Dr. Menendez has participated in international congresses on real-time control, convergence, hybrid systems, and computer science.

Hector BENITEZ-PEREZ

He is a full-time researcher in the IIMAS UNAM (México). He obtained his BSc in electronic engineering from the Engineering Faculty, UNAM, in 1994 and his PhD from Sheffield University, UK in 1999. His areas of interest are in real time control and fault diagnosis.