INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 19, Issue: 4, Month: August, Year: 2024
Article Number: 6606, https://doi.org/10.15837/ijccc.2024.4.6606

CCC Publications
AGORA UNIVERSITY PRESS

Resource Manager for heterogeneous processors

J. A. Aparicio-Santos, H. Benítez-Pérez, L. Alvarez-Icaza, L. Mendoza-Rodríguez

José Alberto Aparicio Santos
Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Universidad Nacional Autónoma de México
Circuito Exterior s/n, Coyoacan, cp 04510, México
alberto.aparicio@unam.mx

Héctor Benítez Pérez, Luís Mendoza Rodríguez
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Dirección General de Cómputo y de Tecnologías de Información y Comunicación
Universidad Nacional Autónoma de México
Circuito Exterior s/n, Coyoacan, cp 04510, México
hector.benitez@unam.mx, ing.luismendoza31@gmail.com

Luis Agustín Alvarez-Icaza Longoria
Instituto de Ingeniería
Universidad Nacional Autónoma de México
Circuito Exterior s/n, Coyoacan, cp 04510, México
alvar@pumas.iingen.unam.mx

Abstract

A resource administrator (RM) can distribute tasks processing time between heterogeneous processors is presented. Its design is based on analyzing of the possible forms of attention to pending tasks and in the previous knowledge about them. A model is proposed based on difference equations that quantify the resource allocation of each task and choose the best processor where the task can be executed. The scheme adapts to dynamic changes in the requirements or the number of tasks.

Keywords: Resource Manager, Constant Band Width Server, Processor Time Assignment, Diffuse control, real-time systems

1 Introduction

A resource manager (RM) distributes computational resources among several tasks or applications that must be served at specific time intervals. This is in itself a complex scheduling problem, which is even more so if it must be done over a heterogeneous multiprocessor platform [3]. The RM presented in this paper takes into account a heterogeneous multiprocessor platform, uses a Constant Bandwidth Server (CBS), where the dynamics of both are described by difference equations. The combination of scheduling algorithms and control theory makes it possible the analysis of the convergence to a fair allocation of resources among the applications of a real-time control system (RCS) involved, when starting from an arbitrary initial distribution that is not fair.

1.1 Symbols and their meanings

The various symbols used in this paper are listed in Table 1.

Table 1: Symbol list

Symbol	**Meaning**
α_i	application-dependent positive constant i
β_i	positive constant dependent on the nature of application i, $\beta_i \in \mathbb{R} : (0, \infty]$
ϵ	positive constant, represents a small number of resources assigned or removed in each iteration; $\epsilon \in \mathbb{R} : 0 < i < 1$; depends on the number of applications $\epsilon = 1/n$
δ	total number of processors $\delta \in \mathbb{N}$
λ_i	atatic priority of application i; $\lambda_i \in \mathbb{R} : [0, 1]$
Λ_i	dynamic priority of application i; $\Lambda_i = \lambda_i [f_{i,k}]\_ \in \mathbb{R} : [-1, 0]$; the operator $[]\_$ is defined in equation (1)
$\Phi_{h,k}$	rate of change of the nominal equity function; $\Phi_{h,k} = \lvert F_{h,i,k}\rvert - \lvert F_{h,i,k-1}\rvert$
$\Pi_{\mathcal{S}_i}$	return to $\bar{s}_{i,k}$ to the $\mathcal{S}_i$ domain
$\Pi_{\bar{V}_i}$	return to $\bar{v}_{i,k}$ to the $\bar{V}_i$ domain
b	processor identifier with greater capacity
$c_{si,k}$	current server budget that attends to application i; $c_{si,k} \in \mathbb{R} : 0 < c_{si,k} < Q_{si}$
C_i	nominal execution time of application i
$d_{si,k}$	deadline of the server that attends to application i; $d_{s,k} \in \mathbb{R} > 0$
$f_{i,k}$	matching function of application i at iteration k, which indicates whether the application is receiving sufficient or insufficient resources, $[f_{h,i,k}]\_ \in \mathbb{R} : [-1, 0]$ the operator $[]\_$ is defined in Equation (1)
$F_{h,i,k}$	observation or equity function of task i, executed in the h processor that measures the effect of the other tasks executed in the same processor on task i it in the k iteration; $F_{h,i,k} \in \mathbb{R} : [-1, 1]$
h	processor number $h \in \mathbb{N}$
i, j	task number identifiers; $i, \neq j$; $i, j \in \mathbb{N}$
k	iteration number; $k \in \mathbb{N}$
n	total number of tasks; $i \in \mathbb{N}$
$p_{h,k}$	remaining capacity of the h processor in the k iteration
P_b	processor with the greatest capacity
P_h	maximum processor capacity h
Q_{si}	maximum budget of the server handling application i; $Q_{si} \in \mathbb{R} > 0$
$r_{i,k}$	application arrival time i
$s_{h,i,k}$	the service level of application i at iteration k, executed on processor h; it is an internal state of application i,it depends on the virtual platform assigned to application i and can only be read/written by the same application, $s_i \in \mathbb{R} : \underline{s_i} < s_i < \overline{s_i}$
si	identifier of the server handling application i; $si \in \mathbb{N} > 0$
T_{si}	period of the server that manages application i; $T_{si} \in \mathbb{R} > 0$
$u_{T,k}$	total capacity available in the system at iteration k
U_T	total capacity of the system, the sum of the capacities of all processors $\sum_{h=1}^{\delta} \bar{P}_h = U_T$
$v_{h,i,k}$	virtual platform running on processor h, assigned to task i at, iteration k; $v_{h,i,k} \in \mathbb{R} > 0$
$\bar{v}_{h,i,k}$	size of the normalized virtual platform on processor h, assigned to task i at iteration k, $\bar{v}_{h,i,k} = v_{h,i,k}/P_h$; $\bar{v}_{h,i,k} \in \mathbb{R} : (0, 1]$
$w_{h,j}$	normalized consumption of application j assigned to processor h
$\bar{w}_i$	mormalized i application consumption

A couple of operators defined below are also used. The operator [ ]_ limits the range of its argument to $[-1, 0] \in \mathbb{R}$ and its defined as:

$$[x]\_ = \begin{cases} x & \text{si } x \leq 0 \\ 0 & \text{si } x > 0 \end{cases} \tag{1}$$

The binary operator $\rfloor x \lceil$ sends its argument to 1 or 0 as indicated by following definition:

$$\rfloor x \lceil = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases} \tag{2}$$

1.2 Background

In order to carry out a resource distribution, one of the first steps is to define the resources to be distributed and choose a mathematical abstraction to quantify them. Some authors use the definition of quality of service [8], which has multiple meanings and perspectives, and that depend specifically on each application.

Another way to quantify resources is through virtual processors [11], which divide the physical processor into multiple processors of lower power. This concept stems from the definition of a virtual private machine introduced by [12]. From this model, [4] and [7] introduce the virtual platform concept, an abstraction of the processor usage ratio employed in this paper.

It is necessary to isolate tasks temporarily to avoid effects such as resource monopolization or starvation due to lack of resources. To achieve this, the virtual platform is managed by a process called a server, where a single server can handle several subsets of tasks, which share resources, a feature that will be used later for the distribution of tasks among heterogeneous processors.

There are three general RM schemes try to meet the distribution of resources without diminishing the quality of service, which are centralized, distributed and hybrid [6].

In the centralized scheme, the RM has control over all the resources of each task, which makes it very stable. However, its numerical complexity grows geometrically with the number of tasks [4].The distributed scheme, [15], is based on local information from each task that regulates its own resource consumption by means of a local RM. This scheme's main weakness is that global resource distribution can be unstable, since there may be no convergence when allocatnig resources to each task, and therefore global optimization can never be guaranteed. The hybrid scheme takes advantage of the centralized and distributed schemes because global resources are distributed through a global RM, which requires minimum information of tasks. On the other hand, each task regulates its local resources, with an emphasis on improving its performance [6].

The scheme in Figure 1 shows the RM used in this work, which assigns the proportion of processor usage, while each application regulates its own level of service. The most important modification with respect to the scheme shown in [2], is that now we work with heterogeneous processors, which may involve several aspects, such as memory access capacity and speed, processor or bus speed, etc. [9]. In this paper, processors are considered heterogeneous when they have different processing speeds.

2 Heterogeneous processor management

As mentioned, the task scheduling problem on a homogeneous multiprocessor platform is complex, and it becomes even more difficult if a heterogeneous platform [9] is considered. Both the model presented in [7] and the extension presented in [2] were designed for homogeneous processors, i.e., with the same processing speed. To deal with heterogeneous processors consider that P_h is the capacity of the processor h and that there must exist an upper bound P_b for this capacity, i.e., $P_b > P_h, ; \forall, h$. This makes it possible to define the normalized capacity $\bar{P}_h$ as

$$\bar{P}_h = \frac{P_h}{P_b} \leq 1 \tag{3}$$

The sum of all normalized speeds results in the maximum system capacity.

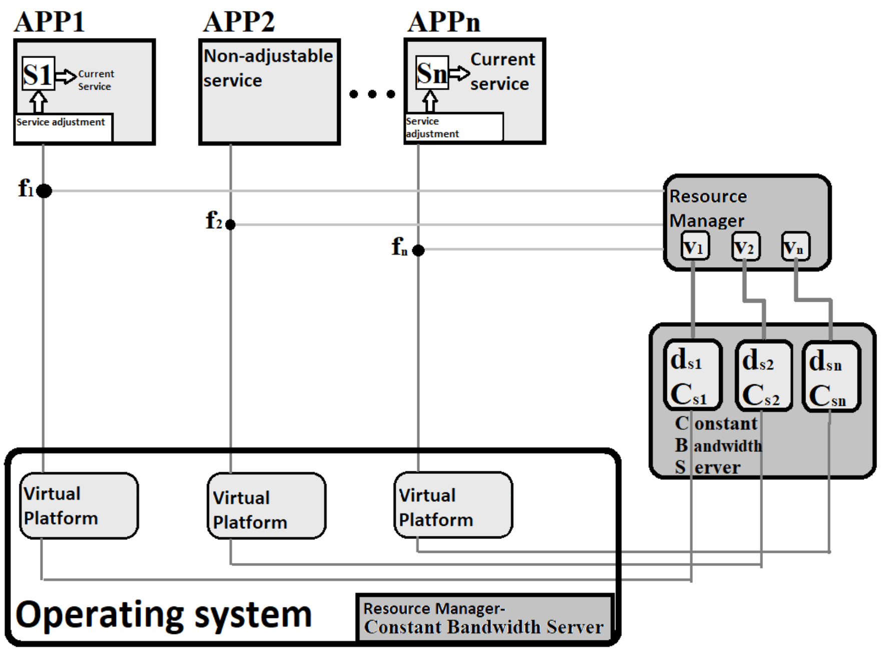


Figure 1: Scheme used in [2], considering a synchronous system with homogeneous processors. The resource managers regulate the size of virtual platforms, while the CBS is responsible for updating the consumption and deadlines of the applications.

$$U_T = \sum_{h=1}^{\delta} \bar{P}_h \tag{4}$$

where δ is the number of processors. It is clear that it is not possible to allocate resources above this bound, if a feasible operation is to be maintained. To perform the allocation between tasks, we update the fairness function calculation introduced in [2] as shown in Equation (5), which is modified so that the maximum platform that can be allocated to a task uses the normalized processor capacity, and to consider only applications that are running on the same processor as the observed application.

$$F_{h,i,k} \doteq -(\bar{P}_h - \bar{v}_{h,i,k})\lambda_i[f_{i,k}]\_ + \bar{v}_{h,i,k} \sum_{l \neq i} \lambda_j [f_{j,k}]\_ \tag{5}$$

This expression also assumes that the virtual platforms assigned to the processor must have local feasibility (dependent on the processor capacity); ϵ is a positive variable dependent on the number of applications,($\epsilon = 1/n$), in order to guarantee convergence [7].The platforms assigned to the processor must meet the following feasibility condition:

$$\sum \bar{v}_{h,i,k} \leq \bar{P}_h \tag{6}$$

where the platforms are now not normalized with respect to the number of processors but with respect to P_b, i.e. $\bar{v}_{h,i,k} = v_{h,i,k}/P_b$.

2.1 System model

The nominal equity level $F_{h,i,k}$ (Equation 11) is proposed as a dynamic indicator of the resources allocated to a task. Figure 2 shows the typical behavior of $F_{i,k}$ as a function of $v_{h,i,k}$. Positive values of $F_{h,i,k}$ indicate lack of resources in the task, while negative values indicate excess of resources. If $F_{h,i,k}$ is zero, the resource allocation is fair. Thus, a negative fairness function indicates a resource allocation that satisfies the observed task, and the more negative its value, the more slack the processor has to attend to the task.

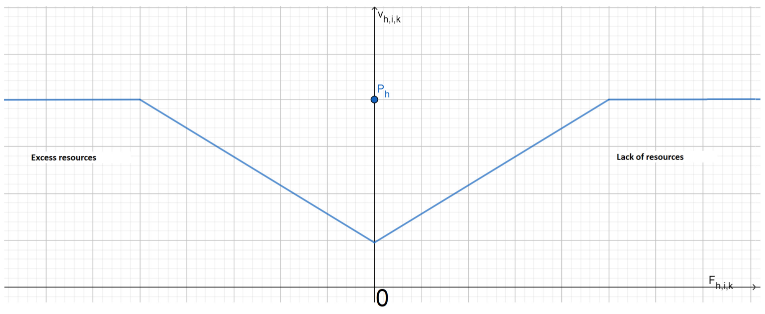


Figure 2: Behavior of the nominal fairness equation $F_{h,i,k}$ against the virtual platform $v_{h,i,k}$. $F_{h,i,k}$ positive indicates lack of resources, on the contrary if it is negative; the application must give up resources; being on a processor of capacity P_h, the allocated resources cannot exceed the capacity of that processor. $F_{h,i,k}$ zero indicates that allocation is fair. The slope of the change depends on the variable ϵ, which in turn is inversely proportional to the number of applications.

The modified equity function is used to complete the dynamic model as follows (see nomenclature in Table 1):

$$\bar{v}_{h,i,k+1} = \bar{v}_{h,i,k} + \epsilon F_{h,i,k} \tag{7}$$

$$s_{i,k+1} = s_{i,k} + \epsilon f_{i,k} \tag{8}$$

$$\begin{aligned} c_{si,k+1} = {} & c_{si,k} \rfloor (d_{si,k} - r_{i,k}) P_h \bar{v}_{h,i,k} c_{si,k} \lceil \\ & + Q_{si} \rfloor c_{si,k} - (d_{si,k} - r_{i,k}) P_h \bar{v}_{h,i,k} \lceil + Q_{si} \rfloor -c_{si,k} \lceil \end{aligned} \tag{9}$$

$$d_{si,k+1} = d_{si,k} + T_{si} \rfloor (d_{si,k} - r_{i,k}) P_h \bar{v}_{h,i,k} c_{si,k} \lceil + T_{si} \rfloor -c_{si,k} \lceil \tag{10}$$

where

$$F_{h,i,k} \doteq -(\bar{P}_h - \bar{v}_{h,i,k}) \lambda_i [f_{i,k}]\_ + \bar{v}_{h,i,k} \sum_{j \neq i} \lambda_j [f_{j,k}]\_ \tag{11}$$

$$f_{i,k} = \beta_i \frac{\bar{v}_{h,i,k}}{s_{h,i,k}} - 1 \tag{12}$$

$$P_h = \bar{P}_h P_b \tag{13}$$

$$\Lambda_{i,k} = \lambda_i F_{i,k} \tag{14}$$

Figures 3 and 4 show typical behaviors of the evolution of the allocated budget and the service level for a task, respectively. In the first case, the resource consumption is considered to occur at a constant rate and the budget is refilled when the deadline is fulfilled. In the case of Figure 4, the service level evolves until it reaches the minimum $s_{i,k_{min}}$ required to serve the task under analysis.

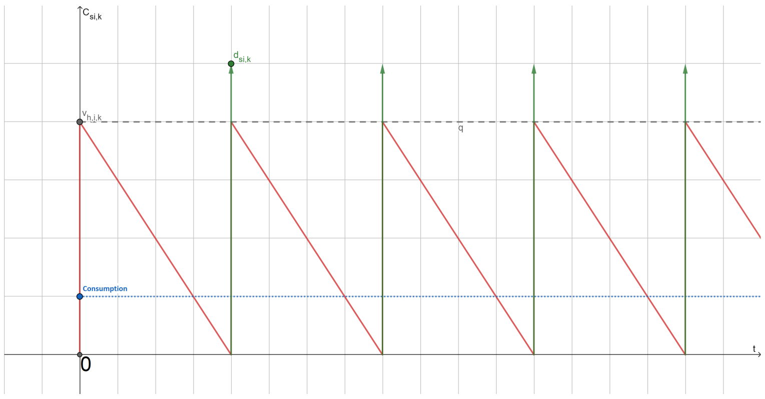


Figure 3: Behavior of the budget update, in the figure the consumption is taken as constant, the CBS is in charge of refilling the budget and postponing the deadline.

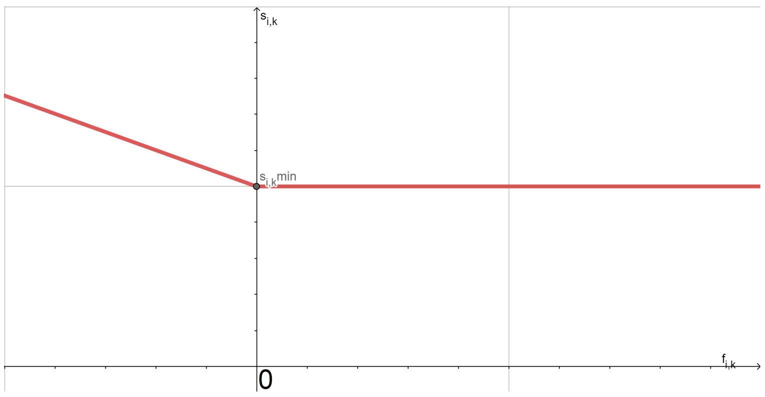


Figure 4: Behavior of service level update versus matching function $f_{i,k}$.

3 Task assignment

In general terms, the model works as follows. The P_h capacity of the h processor starts to be used to allocate resources to the tasks that are added by the virtual platforms $v_{h,i,k}$, where i points to the task number on the h processor and k to the iteration. As mentioned before, the tasks executed on processor h must not exceed the processor capacity, i.e.,

$$\sum_{i=1}^{n_h} v_{h,i,k} \leq P_h \tag{15}$$

The remaining, or still available, capacity $p_{h,k}$ on processor h at time k is:

$$p_{h,k} = P_h - \sum_{i=1}^{n_h,k} v_{h,i,k} \tag{16}$$

The remaining capacity for the whole system at the same iteration k is:

$$u_{T,k} = \sum_{h=1}^{\delta} p_{h,k} \tag{17}$$

In order to carry out the resource allocation when a new task arrives, it is assumed that its consumption is known and four scenarios are considered:

- The task is non-preemptive and it is possible to attend it.
- The task is non-preemptive and it is not possible to attend it.
- The task is preemptive and it is possible to attend it.
- The task is preemptive and it is not possible to attend it.

3.1 Non-preemptive task

If there is capacity to serve a non-preemptive task, the nominal fairness function $F_{h,i,k}$ is used to evaluate the impact of the new task on each processor. If the nominal fairness function has a negative Phi_k slope, this indicates that processor h would be granting sufficient resources to the task. The more negative this slope is, the faster the convergence to a fair distribution in the future will be on that processor. The evaluation of the fairness function is calculated by Equation (5) and the calculation of its slope as:

$$\Phi_{h,i,k} = |F_{h,i,k}| - |F_{h,i,k-1}| \tag{18}$$

For example, in case of having two processors a and b, $\Phi_{a,k} < \Phi_{b,k}$ indicates that processor a will better serve task i. Furthermore, $F_{h,i,k} < 0$ signals that task i has been allocated sufficient resources.

The update of the virtual platform is performed through Equation (7). In case there is no capacity to attend a new task, it can wait to be attended a certain number of iterations, depending on its priority λ_i, or eventually be discarded.

3.2 Preemptive task

In case there is capacity to handle the task on a single processor, it is handled in the same way as in the non-priority case. In case the entire task cannot be handled on a single processor, it can be split to be allocated according to the available capacity of the processors $u_{i,c} = p_{h,k}$, where $u_{i,c}$ is a segment c of the task i, adapted to use the available capacity of the processor h. This division will be possible if the consumption of task u_i is not greater than the total remaining capacity $u_{T,k} = \sum_{h=1}^{\delta} p_{h,k}$. The lack of capacity is handled the same way as in the previous case.

3.3 Updating of the available capacity

The total available capacity $p_{h,k}$ must be compared with the consumption $w_{h,j}$ of the new task j assigned to processor h. Both capacities must be normalized with respect to the maximum capacity of the assigned processor, P_h, therefore it is defined as follows

$$\bar{w}_{h,j} = \frac{w_{h,j}}{P_h} \tag{19}$$

Thus, the update of the available capacity is:

$$p_{h,k} = P_h - \sum_{i \neq j}^{n_h} \bar{w}_{h,j} \tag{20}$$

$$p_{h,k+1} = p_{h,k} - \bar{w}_j \tag{21}$$

where n_h is the number of tasks assigned to processor h and $w_{h,j}$ is the consumption of those tasks, all at iteration k.

3.4 Projection of available capacity

Migrating a task to another processor causes variations in the available capacities, therefore it is not only necessary to know the capacity used by the task, but also to make a projection of the capacity that will be released.The remaining capacity projection is obtained from:

$$\bar{p}_{rh,k+1} = \bar{P}_h - \sum_{i=1}^{n_h} \bar{v}_{h,i,k+1} \tag{22}$$

where n_h is the number of applications running on processor h. Note that if there is capacity on a processor after doing the task allocation, this implies that the system can be planned and the distribution may converge to a fair allocation.

3.5 Initial task allocation

An initial distribution of the tasks on the processors is needed, in this case the constant $\beta_i = \alpha_i / C_i$ is used, which depends on a constant α_i and the nominal execution time C_i of the application i. If $\beta_i < \beta_j$ implies that application i has higher consumption than application j. Based on these comparisons, an initial distribution can be made, where applications with lower $beta_i$ will be assigned to processors with higher capacity.

4 Fuzzy modeling

An RM such as the extended Chasparis [2] incorporates nonlinear terms, the presence of which makes the design of control schemes difficult. One possibility to deal with these nonlinearities, is to take advantage of the ability of artificial neural networks and fuzzy systems to produce models of nonlinear systems that approximate with arbitrary accuracy the nonlinear terms, if they have enough hidden neurons and training data or fuzzy rules, respectively [5].

Here we propose to use a Takagi-Sugeno type fuzzy model [16] which is described by a set of rules **IF – THEN**. This fuzzy RM will be able to approximate the nonlinearities of the extended Chasparis model by a set of linear systems, where for each element it is also possible to design a specific controller. As in any fuzzy system, the current state of the model has a membership level assigned for each element of the set of nonlinear systems. Thus, the fuzzy model-controller combination can, from any arbitrary initial distribution of resources, be brought to a fair distribution, without losing the closed-loop stability of the approximate system.

In addition to the fuzzy approach, it is proposed to use a distributed RM, which is achieved by distributing the extended Chasparis model in three subsystems: i) service adjustment; ii) virtual platform distribution; iii) deadline and budget update (see Figure 5). These subsystems can be treated as three independent systems to design the control schemes that guarantee their closed-loop stability. In all cases, the membership functions used will be triangular in shape.

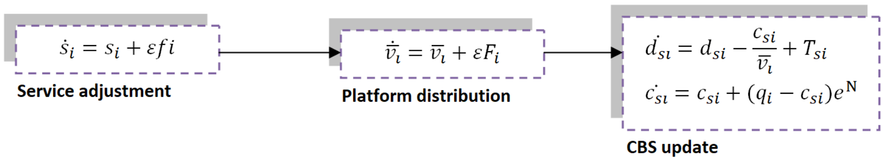

$\dot{s}_i = s_i + \varepsilon f i$
Service adjustment
$\dot{\bar{v}}_i = \bar{v}_i + \varepsilon F_i$
Platform distribution
$\dot{d}_{si} = d_{si} - \frac{c_{si}}{\bar{v}_i} + T_{si}$
$\dot{c}_{si} = c_{si} + (q_i - c_{si})e^N$
CBS update

Figure 5: The system is divided into three subsystems, which are linked by a minimum of information.

4.1 Virtual platform distribution

For the distribution of the virtual platform it is necessary to know the value of the matching function f_i of each application, in order to calculate the nominal equity function F_i; substituting Equation (5) into (7), we obtain:

$$\bar{v}_{i+1} = \bar{v}_i + \epsilon \left\{ -(\bar{P}_h - \bar{v}_{h,i,k})\lambda_i[f_{i,k}]\_ + \bar{v}_{h,i,k} \sum_{l \neq i} \lambda_j [f_{j,k}]\_ \right\} \tag{23}$$

The variable premise is defined as

$$z_{v_i} = \epsilon \left(-(\bar{P}_h - \bar{v}_{h,i,k})\lambda_i[f_{i,k}]\_ + \bar{v}_{h,i,k} \sum_{l \neq i} \lambda_j [f_{j,k}]\_ \right) \tag{24}$$

and membership functions

$$M_{vi_1} = 1 - \frac{z_{v_i} - z_{vi_{max}}}{z_{vi_{min}} - z_{vi_{max}}} \tag{25}$$

$$M_{vi_2} = \frac{z_{v_i} - z_{vi_{max}}}{z_{vi_{min}} - z_{vi_{max}}} \tag{26}$$

These are related to the statements:

- M_{vi_1} insufficient resources for application i.
- M_{vi_2} sufficient resources for application i.

and with the maximum vi_{max} and minimum vi_{min} values of the virtual platform i.

4.2 Service level

The service level adjustment is also carried out within each application since, as mentioned above, it depends on the nature of the application. For this purpose, maximum si_{max} and minimum si_{min} service levels must be defined, so that the service level is always between them. The level of service is calculated from the following equation

$$\dot{s}_i = s_i + \epsilon f_i = s_i + \left[\epsilon\left(\frac{\beta_i \delta \bar{v}_i}{s_i} - 1\right)\right]_{-} \tag{27}$$

where it can be noted that calculations for application i only require local information. A premise variable for each application is defined as:

$$z_{s_i} = \epsilon\left(\frac{\beta_i \delta}{s_i} - 1\right) \tag{28}$$

From this variable premise, two membership functions are defined, which represent the extremes of the situations in which the service level can be found and that are described by the following equations

$$M_{si_1} = 1 - \frac{z_{s_i} - z_{si_{max}}}{z_{si_{min}} - z_{si_{max}}} \tag{29}$$

$$M_{si_2} = \frac{z_{s_i} - z_{si_{max}}}{z_{si_{min}} - z_{si_{max}}} \tag{30}$$

Each of these membership functions is related to the statements:

- M_{si_1} maximum service level in application i.
- M_{si_2} minimum service level in application i.

Table 2: Relationships between the membership functions $M_{vi_1}(z_{v_i})$, $M_{vi_2}(z_{v_i})$, $M_{si_1}(z_{s_i})$, $M_{si_2}(z_{s_i})$; $i = 1, 2, 3$, y w_{vj}, $j = 1, 2, 3...12$ is the quantization of each combination. The form of the linear systems $A_{vj}x_v + B_v u_v$ is given in Equation (31)

$w_{vj}(z)$	IF	THEN $A_{vj}x_v + B_v u_v$
$w_{v1}(z)$	$M_{v1_1}(z_{v_1})$	$A_{v1}x_v + B_v u_v$
$w_{v2}(z)$	$M_{v1_2}(z_{v_1})$	$A_{v2}x_v + B_v u_v$
$w_{v3}(z)$	$M_{v2_1}(z_{v_2})$	$A_{v3}x_v + B_v u_v$
$\vdots$	$\vdots$	$\vdots$
$w_{v11}(z)$	$M_{s3_1}(z_{s_3})$	$A_{v11}x_v + B_v u_v$
$w_{v12}(z)$	$M_{s3_2}(z_{s_3})$	$A_{v12}x_v + B_v u_v$

The relationship between the fuzzy rules and the linear systems associated with each one is illustrated in Table 2, where the systems $A_{vj}x_v + B_v u_v$ have the structure of the equation (31), with i the number of the application and n the total number of these. Since the computation of the virtual platform $\bar{v}_i$ and the service level is performed in a distributed manner, the number of rules and associated linear systems is $4n$. In the example of Table 2, $n = 3$, combining the virtual platform level and service level rules results in 12 rules associated with 12 linear systems. The structure of each of these 12 linear systems is shown below, where the differential equations are obtained from the equations in differences for the virtual platforms and the service levels (see details in [2]). In addition, n equations for the priority dynamics of each application are also incorporated. Note that the inputs u_i are associated with the fairness functions $F_{i,k}$. The structure of the complete system is [1]:

[1]Note that in these equations the processor number is omitted, since as will be explained later, each application will be assigned to only one processor

$$\begin{bmatrix}\dot{\bar{v}}_1\\ \vdots\\ \dot{\bar{v}}_n\\ \dot{s}_1\\ \vdots\\ \dot{s}_n\\ \dot{\lambda}_1\\ \vdots\\ \dot{\lambda}_n\end{bmatrix}=\begin{bmatrix}z_{v_1} & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & 0\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & z_{v_n} & 0 & \cdots & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & z_{s_1} & \cdots & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \cdots & z_{s_n} & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\bar{v}_1\\ \vdots\\ \bar{v}_n\\ s_i\\ \vdots\\ s_n\\ \lambda_i\\ \vdots\\ \lambda_n\end{bmatrix}+\begin{bmatrix}0 & \cdots & 0\\ 0 & \cdots & 0\\ 0 & \cdots & 0\\ & \vdots & \\ 1 & \cdots & 0\\ 0 & \ddots & 0\\ 0 & \cdots & 1\end{bmatrix}\begin{bmatrix}u_{v1}\\ \vdots\\ u_{vn}\end{bmatrix} \tag{31}$$

To guarantee closed-loop stability, the F_i matrices must be proposed and a matrix $P > 0$ must be found that satisfies the following inequality for all A_i matrices [17].

$$G_{ii} = A_i - B_i F_i \tag{32}$$
$$0 > G_{ii}^T P G_{ii} - P \tag{33}$$

4.3 CBS level

The CBS updates two states, the deadline d_{si} and the budget c_{si}, according to the following equations.

$$\dot{c}_{si} = Q_{si}\rfloor c_{si} - (d_{si} - r_i)\delta\bar{v}_i\lceil + Q_{si}\rfloor -c_{si}\lceil \tag{34}$$
$$\dot{d}_{si} = T_{si}\rfloor (d_{si} - r_i)\delta\bar{v}_i c_{si}\lceil + T_{si}\rfloor -c_{si}\lceil \tag{35}$$

Having two equations, two premise variables are defined per application, making the total number of combinations 2^{n*2}, where n is the number of applications.

$$z_{cs_i} = \delta T_{si}\rfloor c_{si} - (d_{si} - r_i)\delta\bar{v}_i\lceil \tag{36}$$
$$z_{ds_i} = \frac{1}{\delta\bar{v}_i}\rfloor (d_{si} - r_i)\delta\bar{v}_i c_{si}\lceil \tag{37}$$

The membership functions are defined by the equations (38-41)

$$M_{csi_1} = 1 - \frac{z_{cs_i} - z_{csi_{max}}}{z_{csi_{min}} - z_{csi_{max}}} \tag{38}$$
$$M_{csi_2} = \frac{z_{cs_i} - z_{csi_{max}}}{z_{csi_{min}} - z_{csi_{max}}} \tag{39}$$
$$M_{dsi_1} = 1 - \frac{z_{ds_i} - z_{dsi_{max}}}{z_{dsi_{min}} - z_{dsi_{max}}} \tag{40}$$
$$M_{dsi_2} = \frac{z_{ds_i} - z_{dsi_{max}}}{z_{dsi_{min}} - z_{dsi_{max}}} \tag{41}$$

where csi_{max}, csi_{min}, dsi_{max} y dsi_{min} are the maximum and minimum limits of cs_i y ds_i, respectively, each linked to the following statements:
- M_{csi_1} insufficient budget for application i.
- M_{csi_2} fill in budget for application i.
- M_{dsi_1} not required to update application deadline i.
- M_{csi_2} update application deadline i.

The linear systems $A_{ci}x_c + B_c u_c$ have the form of the equation (42):

$$\begin{bmatrix} \dot{c}_{s_i} \\ \vdots \\ \dot{c}_{s_n} \\ \dot{d}_{s_i} \\ \vdots \\ \dot{d}_{s_n} \end{bmatrix} = \begin{bmatrix} 1 & \dots & 0 & z_{dc_i} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 1 & 0 & \cdots & z_{dc_n} \\ 0 & 0 & 0 & z_{d_i} & \cdots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & z_{d_n} \end{bmatrix} \begin{bmatrix} c_{s_i} \\ \vdots \\ c_{s_n} \\ d_{s_i} \\ \vdots \\ d_{s_n} \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 1 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_i \\ \vdots \\ u_n \end{bmatrix} \tag{42}$$

As in the two previous subsystems, the stability condition stated in [17] must be met to guarantee closed-loop stability.

5 Results

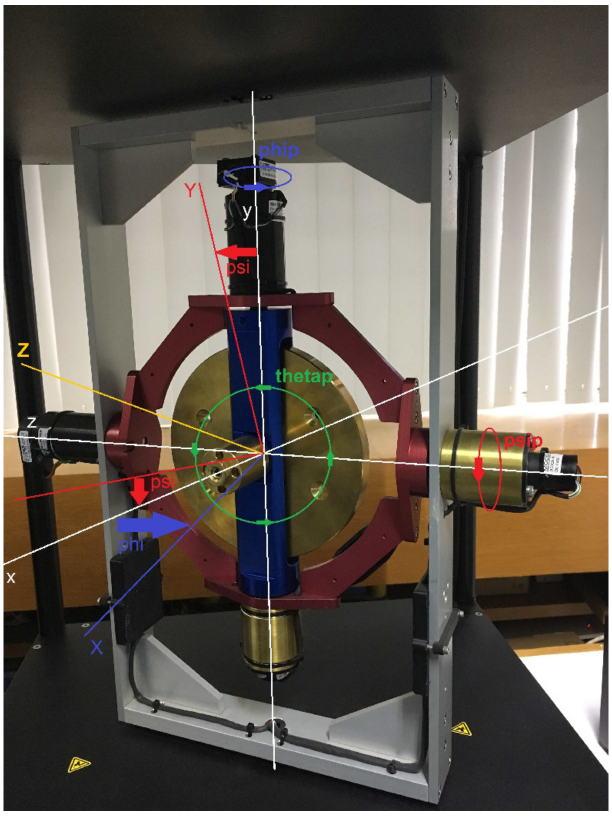


Figure 6: Gyroscope, reference frame: $theta = \theta$, $psi = \psi$, $phi = \phi$, $thetap = \dot{\theta}$, $psip = \dot{\psi}$, $phip = \dot{\phi}$.

To test the performance of the RM and CBS, a directly configured networked control system (NCS) of a three-degree-of-freedom gyroscope model is simulated (see Figure 6).

The goal of the NCS is to control two of the three degrees of freedom of the gyroscope, which correspond to the angles ϕ and ψ in the reference frame shown in Figure 6 and whose dynamic model is given by [14].

$$M_b = J_y\ddot{\phi} - J_x^d\dot{\theta}\dot{\psi}\cos(\phi) + (J_z - J_x)\dot{\psi}^2\sin(\phi)\cos(\phi) \tag{43}$$

$$\begin{aligned} M_r &= (J_z^r + J_z\cos^2(\phi) + J_x\sin^2(\phi))\ddot{\phi} \\ &\quad + J_x^d\dot{\theta}\dot{\phi}\cos(\phi) + 2(J_x - J_z)\dot{\psi}\dot{\phi}\sin(\phi)\cos\phi \end{aligned} \tag{44}$$

The terms composing the equations of motion (43-44) and the value of their parameters given by the manufacturer [13] are:

θ	: angular velocity of the disk around its own axis of rotation x. $\theta = 150rad/s$.
ϕ	: angular position of the blue gimbal suspension about y.
ψ	: angular position of the red gimbal suspension over Z.
J_z^r	: moment of inertia of the red gimbal suspension about Z axis. $J_z^r = 0.0342kgm^2$.
J_z^d	: moment of inertia of the disk about x axis. $J_z^d = 0.0056kgm^2$.
J_x	: moment of inertia of the disk and the blue gimbal suspension about x axis. $J_x = 0.0074kgm^2$.
J_y	: moment of inertia of the disk and the blue gimbal suspension around the y axis. $J_y = 0.0026kgm^2$.
J_z	: moment of inertia of the disk and the blue gimbal suspension around the z axis. $J_z = 0.0056kgm^2$.
M_b	: total external torque around the rotation axis of the blue gimbal suspension.
M_r	: total external torque around the rotation axis of the red cardan suspension.

The closed-loop control of the gyroscope is also realized through a fuzzy system, details of its design can be found in [1].

Three non-priority applications are set up in the NCS, each of which is assigned different tasks, as described in Figure 7. The $APP1$ is responsible for sensors readout tasks, $APP2$ for control signal computation, and $APP3$ for sending the control signal to the actuators tasks. The RM is implemented in each of the applications, while the CBS was implemented by a prioritized centralized task. In addition to performing their assigned tasks, the applications must send over the network the instantaneous value of their pairing functions $f_i; i = 1, 2, 3$.

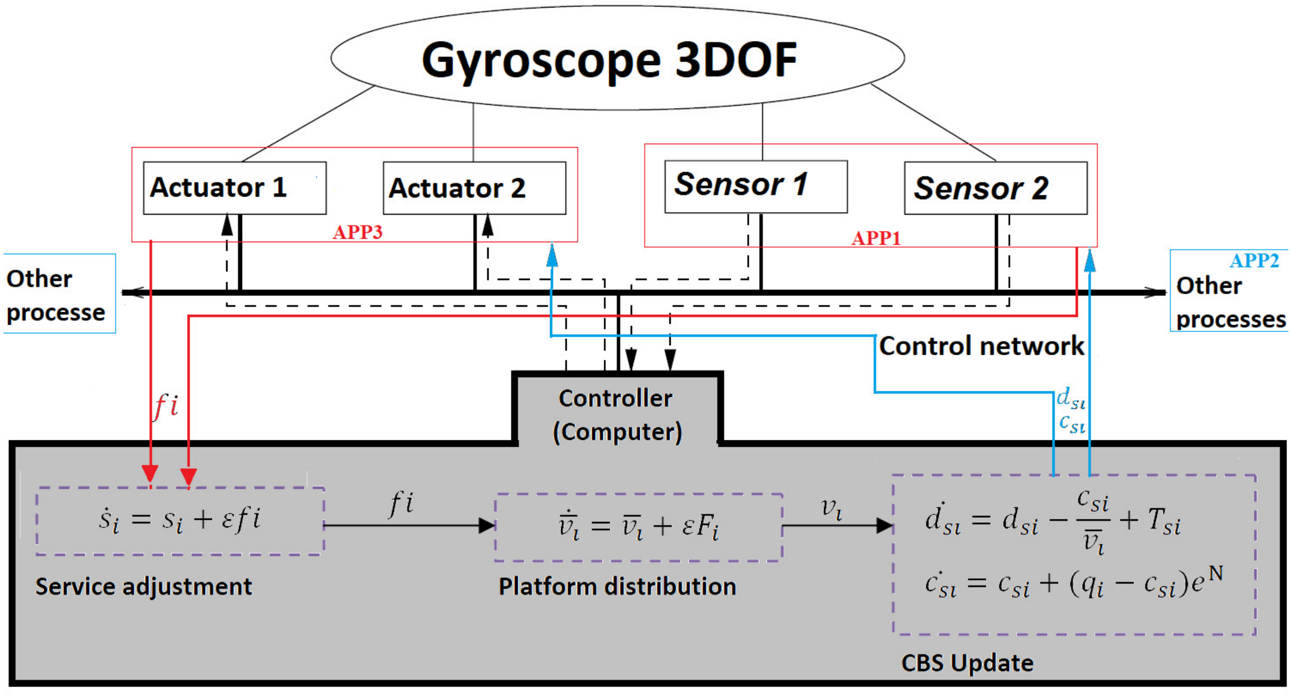
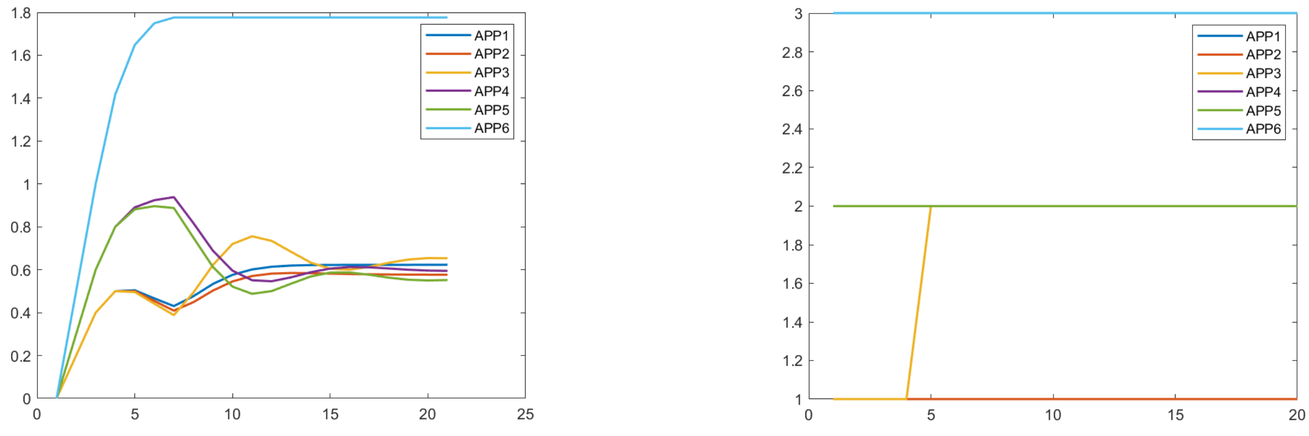


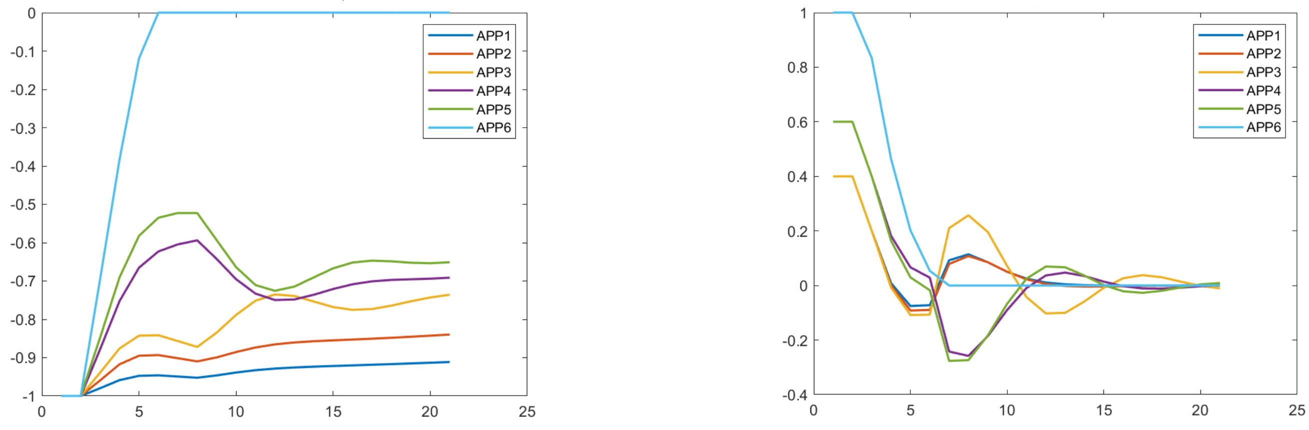
Figure 7: NCS system direct configuration, connected to the resource manager, it is observed that the only information received from the applications is the value of the pairing function f_i.

For the first experiment we use the following initial conditions: All virtual platforms are initially zero, we have three processors with different capacities $P_1 = 2$, $P_2 = 3$, $P_3 = 4$, serving six tasks, with different requirements. For simplicity, the requirements are ordered with the constant β_i. The chosen values are: $\beta_1 = 1 < \beta_2 = 2 < \beta_3 = 3 < \beta_4 = 4 < \beta_5 = 5 < \beta_6 = 6$. The results are described below.

In Figure 8a, it can be seen that the platform for APP6 is more demanding than the other platforms, which is why it takes a much larger amount of resources. Some oscillations are noticeable due to the resource balancing between the applications, but these tend to decrease as the algorithm finds a fair resource allocation. In Figure 8b, it is shown that APP3 migrates from processor 1 to 2 in iteration 4. In processors 2 and 3 there are no migrations because their capacity is greater than that of processor 1. In Figure 8c, the matching function presents oscillations due to the changes in the virtual platforms, but also here, when the algorithm approaches a fair distribution, the oscillations decrease. It should be noted that only APP6 converges to zero in all twenty iterations, so locally APP6 does not ask for more resources. Finally in Figure 8d it can be seen that despite what we have seen in the matching function, the fairness function converges to zero for all applications, so it is concluded that the distribution is fair.



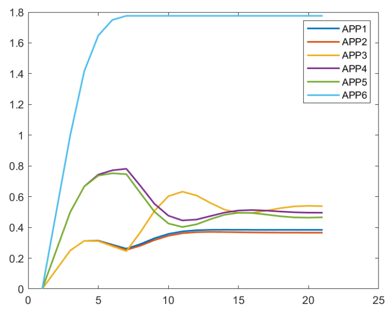
(a) Virtual platform $v_{i,k}$, case 1. (b) Migration between processors, case 1.



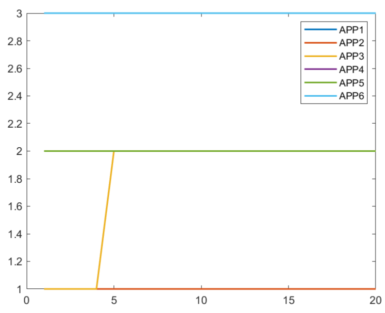
(c) Matching function $f_{i,k}$, case 1. (d) Nominal fairness function $F_{i,k}$, case 1.

Figure 8: Behavior of case 1

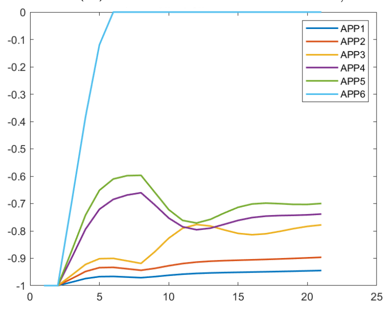
For the second experiment, we have the same parameters as in the previous simulation, only the processor capacity is decreased: All virtual platforms start at zero, we have three processors with different capacities $P_1 = 1$, $P_2 = 2$, $P_3 = 3$, serving the same six tasks. The results are shown in Figures 9a-9d. In Figure 9a, it is observed that despite having less capacity, the distribution for APP6 is equal to case 1, presenting the same oscillations due to the exchange of resources between applications, however the final distribution of resources of the other applications is more disperse. In Figure 9b, once again the migration of APP3 from processor 1 to 2 is observed. In Figure 9c, similarly to case 1, APP6 reaches a satisfaction of its resources since it is only executed by one processor. Finally, in Figure 9d the matching functions converge again presenting oscillations that tend to decrease as the algorithm evolves and makes a better balance between the resources assigned to the different applications, moreover, it is concluded that the distribution is fair.



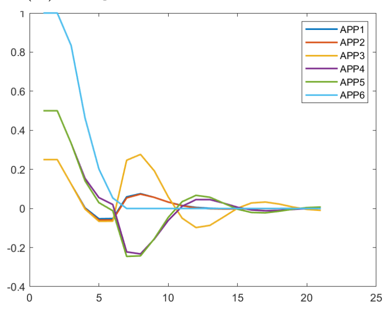
(a) Virtual platform $v_{i,k}$, case 2.



(b) Migration between processors, case 2.



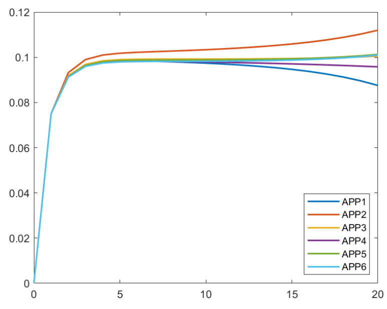
(c) Matching function $f_{i,k}$, case 2.



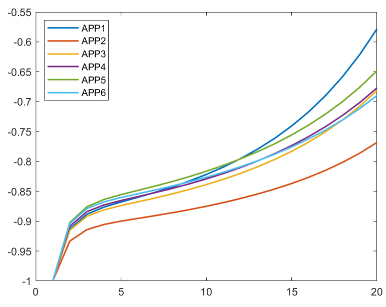
(d) Nominal fairness function F_i, case 2.

Figure 9: Behavior of case 2

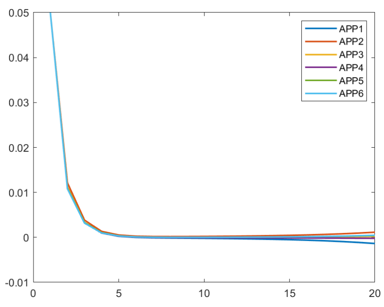
Finally, an experiment with similar characteristics to the second case is performed, but using the homogeneous algorithm and where the three processors have a capacity $P = 2$. In addition, the migration of tasks between processors was suppressed. It can be seen in Figure 10a that the algorithm starts to diverge at the end of the simulation. Given the excess consumption required by the applications, convergence is not possible. As in the heterogeneous case, there are oscillations due to the exchange of resources between the applications. The matching function shown in Figure 10b confirms the divergence trend. As the resource demand of the applications is very large, in 20 iterations the algorithm does not reach a fair distribution. Finally, the fairness function in Figure 10c, shows that it is not possible to reach the convergence to zero that would indicate a fair distribution and that eventually some tasks do not receive enough resources.



(a) Virtual Platform $v_{i,k}$, case 3.



(b) Matching function $f_{i,k}$, case 3.



(c) Nominal fairness function F_i, case 3.

Figure 10: Behavior of case 3

Analyzing the three experiments, it can be seen that in the first case the availability of resources is higher than the demand by the applications, therefore, the RM can allocate all the requested resources to all of them. In the second case, the requested resources are closer to the available capacity, however, the heterogeneity of the processors helps to better cope with the demand for resources and all applications also receive the necessary resources. In these first two cases, the fairness function converges to zero, indicating a fair distribution. Finally, in the third experiment, the sum of available resources on the three processors matches the second experiment, however, the RM can no longer allocate the necessary resources to all applications, which is indicated by the onset of divergence in the virtual platforms and the lack of convergence in the fairness function.

6 Conclusions

A model of a RM for heterogeneous processors described by difference equations was presented, which takes into account the distribution of global and local resources, as well as the behavior of the CBS. This RM takes into account the performance of the tasks in each processor partition, in case of not meeting an expected performance, the task can be migrated to another processor or discarded.
A representation by means of RM difference equations was designed in order to obtain a mathematical description of each component of the system. The equations describe the resource distribution, and the budget padding of the CBS, as well as the deferral of the deadline.
Simulation results were presented, confirming the feasibility of the proposed scheme and emphasizing the need to have the minimum resources for its correct execution.
The main disadvantage of this scheme is that the convergence speed is linked to the number of tasks; the more tasks the convergence tends to be slower, i.e., achieving a fair distribution takes more time. This could be impractical especially in applications that require real time.

Acknowledge

The authors are grateful for the support of UNAM and its institutes, especially Rita Rodriguez Martinez and Ricardo Villareal Martinez of the Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas - (IIMAS).

Author contributions

The authors contributed equally to this work.

Conflict of interest

The authors declare no conflict of interest.

References

[1] Aparicio-Santos, J.-A., 2017. Diseño de un controlador difuso para compensar cargas de comunicación en tiempo real. Master's thesis, Universidad Nacional Autónoma de México, México.

[2] Aparicio-Santos, J.-A., Hermosillo-Gómez, J.-A., Benítez-Pérez, H., Icaza-Longoria, L. A., 2021. Controlador difuso para compensar cargas de comunicación en sistemas en tiempo real. Revista Iberoamericana de Automática e Informática industrial 18 (3). DOI: 10.4995/riai.2021.14544

[3] Baruah, S., Bertogna, M., Buttazzo, G., 2015. Multiprocessor Scheduling for Real-Time Systems. Springer Publishing Company, Incorporated.

[4] Bini, E., Buttazzo, G., Eker, J., Schorr, S., Guerra, R., Fohler, G., Arzen, K. E., Romero, V., Scordino, C., May 2011. Resource management on multicore systems: The actors approach. IEEE Micro 31 (3), 72–81. DOI: 10.1109/MM.2011.1

[5] Boutalis, Y., Theodoridis, D., Kottas, T., Christodoulou, M. A., 2014. System Identification and Adaptive Control: Theory and Applications of the Neurofuzzy and Fuzzy Cognitive Network Models. Springer.

[6] Byeong Gi, L., Daeyoung, P., Hanbyul, S., 2009. Wireless Communications Resource Managemen. John Wiley and Sons.

[7] Chasparis, G. C., Maggio, M., Bini, E., Arzen, K.-E., 2016. Desing and implementation of distributed resource management for time-sensitive applications. Automatica 64, 44 – 53. DOI: https://doi.org/10.1016/j.automatica.2015.09.015

[8] Ganz, A., Ganz, Z., Wongthavarawat, K., 2003. Multimedia Wireless Networks: Technologies, Standards and QoS. Pearson Education.

[9] Hermosillo, J.-A., 2020. Algoritmos de planificación de tiempo real en sistemas distribuidos hetérogeneos considerando la migración de tareas. Ph.D. thesis.

[10] Mahmoud, M., of Engineering, I., Technology, 2013. Distributed Control and Filtering for Industrial Systems. Control, Robotics and Sensors. Institution of Engineering and Technology.

[11] Mok, A. K., Feng, X., May 2001. Resource partition for real-time systems. In: Proceedings Seventh IEEE Real-Time Technology and Applications Symposium. pp. 75–84. DOI: 10.1109/RTTAS.2001.929867

[12] Nesbit, K. J., Moreto, M., Cazorla, F. J., Ramirez, A., Valero, M., Smith, J. E., May 2008. Multicore resource management. IEEE Micro 28 (3), 6–16. DOI: 10.1109/MM.2008.43

[13] Quanser, 2012. USER MANUAL 3 DOF Gyroscope Experiment Set Up and Configuration. Quanser inc.

[14] Robert H. Cannon, J., 2003. Dynamics Of Physical Systems. Dover Publications, INC.

[15] Subrata, R., Zomaya, A. Y., Landfeldt, B., Oct 2008. A cooperative game framework for qos guided job allocation schemes in grids. IEEE Transactions on Computers 57 (10), 1413–1422. DOI: 10.1109/TC.2008.79

[16] Tanaka, K., Ikeda, T., Wang, H. O., May 1998. Fuzzy regulators and fuzzy observers: relaxed stability conditions and lmi-based designs. IEEE Transactions on Fuzzy Systems 6 (2), 250–265. DOI: 0.1109/91.669023

[17] Tanaka, K., Wang, H. O., 2001. Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach. Wiley-Interscience.

Cite this paper as:

Aparicio-Santos, J.A.; Benítez-Pérez, H.; Alvarez-Icaza L.; Mendoza-Rodríguez, L. (2024). Resource Manager for heterogeneous processors, *International Journal of Computers Communications & Control*, 19(4), 6606, 2024.
https://doi.org/10.15837/ijccc.2024.4.6606