Dynamic Distributed Scheduling of Soft Real-Time Tasks

André M. Barroso, Julius Leite and Orlando Loques
{melon, julius, loques}@caa.uff.br
Instituto de Computação
Universidade Federal Fluminense

Resumo

Em sistemas distribuídos de tempo real é possível transferir a execução de uma ou mais tarefas de um nó sobrecarregado para um outro nó, de forma a aumentar o número de tarefas executadas que tenham atendidas suas especificações temporais. Um método utiliza teoria Bayesiana de decisão para inferir o estado de carregamento do sistema e, assim, escolher um nó "adequado" para onde transferir a tarefa. O conceito de "adequado" é função dos objetivos da política de localização adotada. Neste método, os objetivos são representados através de uma função utilidade, U, que fornece, para cada possível decisão do escalonador, um número que representa o nível de adequação desta decisão aos objetivos estabelecidos. Neste artigo, propomos um novo método para a construção de funções utilidade através da representação de objetivos pela lógica difusa. Aqui, este método é aplicado para aumentar o percentual de tarefas aperiódicas que terão seus objetivos temporais atendidos em sistemas distribuídos de tempo real.

Abstract

In distributed real-time systems the scheduler of an overloaded node may choose to transfer the execution of one or more of its tasks to other less busy nodes in order to improve the number of tasks that meet their deadlines. One solution uses Bayesian theory to infer the loading state of the system and, based on this information, the scheduler chooses an "adequate" node to transfer a task to. The concept of "adequate" will be a function of objectives that are established in the location policy adopted. In the Bayesian decision method, objectives are represented by an utility function (U): given a system state, U returns, for each possible decision to be taken, a number that represents the level of accordance with the required objectives. However, the development of a utility function can be a tricky and somewhat subjective task. In this paper, we describe a new method which easily maps transfer objectives into useful mathematical expressions, representing location policy objectives by fuzzy sets. The proposed method was employed to add objectives to a Bayesian decision-based algorithm aiming to improve the number of aperiodic tasks executed over time in a distributed real-time system.

1. Introduction

Distributed real-time systems can be considered as a number of processes, some of them subject to time constraints, which are executed in different address spaces. Such systems have gained increasing importance in recent years, with applications in a wide range of areas, from multimedia to industrial control. Due to this interface with strategic application fields,

efforts to solve various open research issues have been demanding [8]. Depending on the time strictness imposed on the system responses under predefined environment stimuli, the real-time systems are classified as hard or soft. Hard real-time systems imply the existence of critical tasks that may not miss the deadlines assigned to them. Soft real-time systems impose only the presence of essential tasks, that can miss deadlines, although with a loss of service quality. In order to meet all its time constraints, a system has to share efficiently the necessary resources among its tasks. The assignment of portions of resources time to the tasks defines what is known as a task schedule. The task scheduling problem consists of defining a schedule that can meet all timing and logical constraints of the tasks being scheduled. This problem, in its general form, has been shown to be NP-complete [4]

In order to guarantee that a group of tasks will meet its constraints, it is necessary to perform a schedulability analysis. Such an analysis verifies whether a certain task set is able to produce a feasible schedule (= all constraints met) and assumes previous knowledge of task behavior. Predictable behavior makes it possible to assign hard deadlines to the tasks belonging to the set being scheduled: presumably, catastrophic consequences caused by an eventual violation of a timing constraint can be prevented at system design time. Tasks that present a sufficiently predictable behavior to have hard deadlines are of two types: periodic and sporadic. Besides periodic and sporadic tasks there is a third group of tasks named aperiodic. Belonging to this group are those tasks that can be triggered at any moment. Because of the impossibility of doing a worst-case analysis, aperiodic tasks cannot be assigned hard deadlines. Only on the arrival of an instance of such a task can the scheduler determine if it will be able to meet its constraints and to keep other task guarantees. If the system has only one processor and the arrival of aperiodic tasks exceeds the system capacity to meet all timing constraints, than one or more tasks will have to miss their deadlines. A measure of a system's failure to complete tasks in time is given by the system dynamic failure ratio (R_{df}), defined as the total number of tasks that miss their deadlines divided by the total number of task arrivals. In distributed systems, however, there is an alternative action which may be taken by the scheduler in overload situations: if a task cannot meet its deadline at a given node of the system, try to guarantee it elsewhere. This can be achieved by transferring the task code through the communication subsystem, or by sending activation messages to pre-loaded replicas of the task at other nodes. Independently of the mechanism employed, we refer to the action of executing a task in a remote node as task transfer or task migration.

The scheduling problem in distributed systems can be conceptually separated into two parts. As there are many nodes where a task can be executed, the first question to be answered is how to assign processors to tasks. This assignment is known as task allocation or global scheduling. Once tasks have been allocated, the problem becomes one of defining a feasible local schedule for each node. Task allocation can be done statically or dynamically. In the static case, the allocation will not suffer changes once defined. For tasks with predictable behavior (periodic and sporadic), static scheduling provides a reliable way to meet time constraints. Unfortunately, finding an optimal feasible allocation is, in almost all practical cases, NP-hard. Approaches to find a good solution to this problem in polynomial time applying heuristics can be found in [1]. Aperiodic tasks can also be statically allocated to the nodes of the system, but, as stated before, the unpredictable behavior of such tasks can lead to overload situations. Nevertheless, while some nodes are overloaded in a distributed system, others may be with low activity or even idle. Dynamic global scheduling can make use of a system's idle capacity to execute tasks that would miss deadlines if not transferred. The problem here involves two aspects. Firstly, it has to be defined when a task is to be

transferred. The methodology chosen to solve this issue defines a transfer policy. At this moment, task migration imposes another issue: where to transfer a task in order to meet its deadline. A location policy is a methodology aimed to choose a destination node for a transferred task. Making an optimal choice involves perfect knowledge of the entire system load at task arrival time at the destination node. This would not be an issue if communication costs were null. The delay in transferring state information and tasks makes it impossible for a node scheduler to obtain the global data necessary to take an optimal decision. Uncertainty will always be present in this decision-making process. Various algorithms have been proposed to deal with this problem [*e.g.*, 3, 6, 7]. The simplest approach maintains uncertainty at its highest level: random selection of a task destination. Despite its simplicity, this algorithm is able to improve the number of tasks that meet their deadlines without having to overload the communication subsystem with the exchange of state information messages. The flexible algorithm [7], Bayesian decision-based algorithm [5, 6] and others, try to reduce uncertainty through estimates based on information provided by the exchange of messages. The Bayesian decision method, as the name implies, uses Bayesian theory to infer the load on a system at the moment of task migration. Based on the inferred state of the system, the scheduler can choose an "adequate" node to transfer a locally unguaranteed task to. The concept of "adequate" will be a function of objectives that are established in the location policy adopted. An obvious objective for a real-time system transfer policy is to send a task only to those nodes able to execute it in time. Nevertheless, other objectives may be imposed on task transfers for fault-tolerance and/or efficiency reasons. Uncertainty in a task destination decision making process can be reduced by adding another aim to the location policy: a task shall preferentially be moved to a node "near" to the sender. Proximity here is measured in terms of communication delay between sender and receiver. A short delay in transferring messages and tasks decreases the probability of inconsistencies between observed and real node states. A location policy may also use a system load profile, in order to improve the chances of finding a node that would satisfy the computational needs of incoming real-time tasks. Homogeneous load sharing exploits parallelism and therefore could reduce average system response time. However, Bestavros showed [3] that, for real-time systems, a greater number of tasks could meet their deadlines if load were unequally distributed among system nodes.

In the Bayesian decision method, objectives are represented by an utility function (U): given a system state, U returns, for each possible decision to be taken, a number that represents the level of accordance with the required objectives. However, the development of an utility function can be a tricky and somewhat subjective task [6]. Simply stated, a general rule to construct such a function is given by the following reasoning: for an universe of possible decisions, high values of utility shall be conferred on decisions that conform well to the problem objectives. On the other hand, decisions that barely suit the purposes of a location policy shall receive low values of utility. In spite of its simplicity, this rule may be difficult to apply, especially if many objectives with different levels of importance must be considered. Saying that a decision is "well suited" or "barely suits" is very fuzzy. Linguistic variables and qualifiers are able to form powerful and short rules, but introduce a lot of imprecision into them. What is needed is an approach to conveniently map this imprecise statement into useful mathematical expressions. In order to perform such a task, we propose representing location policy objectives by fuzzy sets. Utility functions will then be constructed by applying a multiobjective fuzzy decision method. In this context, the method proposed will be employed to add objectives to a Bayesian decision algorithm aiming to improve the number of aperiodic tasks executed per unit of time.

2. The Bayesian Decision Model

For a given decision problem, a set of actions $A=\{a_1, a_2,...\}$ must be evaluated to define the utility of each one for our purposes. However, a precise action evaluation assumes perfect knowledge of its consequences to the system under study. This would only be possible if we knew at each time the real system state. Supposing that this information is available, a function can then be defined that assigns to each action and system state a utility value. If Θ is the universe of system states, the utility function can be represented as follows: $U(\theta,a)$: $\Theta \times A \rightarrow R$. The best action is easily identified as the one that maximizes $U(\theta,a)$. Nevertheless, as the system information available in many problems is far from perfect, another approach must be considered. If information about the real system state is not reliable, and there is a way to check later its real meaning, a likelihood distribution $P_{\theta|z}(\theta,z)$: $\Theta \times Z \rightarrow [0,1]$ can be constructed. $P_{\theta|z}$ returns the probability of the system being at state $\theta \in \Theta$, given an observation $z \in Z$ (=set of all possible observations). In this case, the expected utility for each action can be derived from the following equation:

$$\Psi(z,a) = \int_{\Theta} U(\theta,a).P_{\theta|z}(\theta,z)d\theta \qquad (1)$$

For an observation $z \in Z$, the action that maximizes (1) is considered the best one to be taken. The probability distribution $P_{\theta|z}(\theta,z)$ is calculated from the Bayes formula as follows:

$$P_{\theta|z}(\theta,z) = \frac{P_{\theta}(\theta).P_{z|\theta}(z,\theta)}{\int_{\Theta} P_{\theta}(\theta).P_{z|\theta}(z,\theta)d\theta} \qquad (2)$$

$P_{\theta}(\theta)$, referred as *a priori* likelihood distribution, gives the probability of the system being at state $\theta \in \Theta$. $P_{z|\theta}(z,\theta)$ is the conditional probability that the decider observes z when the real system state is θ.

The method described can be used in each node of a real-time distributed system to implement its dynamic global scheduling policy. As stated before, load information carried by messages exchanged between nodes can become obsolete due to communication delays. Due to this fact, if a scheduler must decide at runtime where to send a task with time constraints, all it has is imprecise information about the system load. It is possible, however, to construct at each node the probability distributions required by the method through the appropriate manipulation of message exchanges. A complete description can be found in [2].

3. Multiobjective Utility Functions

A fundamental axiom of classical logic states that either something is or is not; there is no other possibility. This dualism is normally used to classify things into groups according to type. If its possible unequivocally to define and check the properties of the elements of a group, there will be no difficulty in such a process. The groups thus formed have well-defined boundaries and are known as crisp sets. A sample of a given substance, for instance, can be precisely classified (or not) as water because of its unique molecular structure. The set of substances that are “water” is, therefore crisp. However, the classical logic dualism can be

hard to apply in some situations. Classifying samples of water as hot or not hot is an example of such a problem. One can say that water at 80°C or higher is hot. But what to say of water at 79.99°C ? If it is accepted that water at this temperature cannot yet be considered "not hot", the previously stated axiom is unable to properly express reality. A way to deal with such problem is applying fuzzy logic, instead of classical logic. The dualism is not a fundamental rule in fuzzy reasoning, and hence a set (termed fuzzy) can be defined without precise boundaries. An element can, at the same time, belong and not belong to the set. The level of conformity of a particular element with the generic properties required for elements of a fuzzy set $\underset{\sim}{A}$ is termed membership ($\mu_{\underset{\sim}{A}}$) and measured in the interval [0,1]. A membership function $\mu_{\underset{\sim}{A}}(x): U \rightarrow [0,1]$, maps the elements of universe *U* into a value in [0,1] in accordance with their membership in $\underset{\sim}{A}$. If $\mu_{\underset{\sim}{A}}(x)$ returns 1 for a given element, it clearly belongs to $\underset{\sim}{A}$. The value $\mu_{\underset{\sim}{A}}(x) = 0$ means that *x* does not belong to $\underset{\sim}{A}$. Intermediate values suggest partial conformity of the element with $\underset{\sim}{A}$.

A multiobjective decision problem involves the choice of an action, among the many that are possible, that simultaneously satisfy well more than one objective. A method to define the best action in such a problem, was proposed by Yager [9]. Suppose a set of actions A={a_1, a_2, ...} must be evaluated to define which one is the most convenient for some predefined purposes. Various objectives must be satisfied by the action chosen and these may have different levels of importance. An objective is represented by a fuzzy set $\underset{\sim}{O}$. The membership $\mu_{\underset{\sim}{O}}(\mathrm{a})$ denotes the conformity of action a $\in$ A with the objective $\underset{\sim}{O}$. An objective can also have its importance represented by a fuzzy set $\underset{\sim}{B}$. Every action will have the same membership $\mu_{\underset{\sim}{B}}(\mathrm{a}) = b$ in $\underset{\sim}{B}$. The constant *b* will be termed weight of the objective. As an action has various objectives to be satisfied, a fuzzy set $\underset{\sim}{D}$ representing them all can be defined as follows:

$$\underset{\sim}{D} = \bigcap_{i=1}^{n} (\overline{\underset{\sim}{B}}_i \cup \underset{\sim}{O}_i) \qquad (3)$$

The membership $\mu_{\underset{\sim}{D}}(\mathrm{a})$ quantifies the conformity of action a$\in$A with all the n objectives considered, and can be obtained as [see 2 for details]:

$$\mu_{\underset{\sim}{D}}(a) = \min[\max[1 - b_1, \mu_{\underset{\sim}{O}_1}(a)], ..., \max[1 - b_n, \mu_{\underset{\sim}{O}_n}(a)]] \quad (4)$$

The action with greatest membership in $\underset{\sim}{D}$ is considered the best one to be taken.

In the Bayesian decision method, objectives are represented by the utility function U. The way that objectives are described plays an important role in the construction of a utility function. If precise rules are stated, the quantification of the level of accordance with the objectives may be easy to perform. Nevertheless, such rules can be very difficult, if not impossible, to obtain in some cases. Conversely, imprecise rules are easier to state, but more difficult to map into useful numbers. In order to map imprecise rules into useful mathematical

expressions, the various objectives considered in a decision problem can be represented as fuzzy sets. Applying expression (3), a fuzzy set $\underset{\sim}{D}$ can then be formed, representing all the objectives and taking into account their relative importance for the final decision. The utility of an action is then given by the membership of $a \in A$ in $\underset{\sim}{D}$ as follows:

$$U(a) = \mu_{\underset{\sim}{D}}(a) \qquad (5)$$

Therefore, the utility function is entirely defined by the membership function in $\underset{\sim}{D}$. This method was employed to add objectives to the Bayesian decision algorithm aiming to improve the number of aperiodic tasks executed in time.

4. Evaluation of the Method through Simulation

4.1. Simulation Model and Evaluation Criteria

In the model used in this study, when an aperiodic task arrives at a node of the simulated distributed system, the scheduler tries to meet its timing constraint. If the CET (Cumulative Execution Time) of higher priority tasks is greater than the laxity of the arriving task, it must be transferred. Otherwise, it is accepted by the node and queued for execution. However, if the acceptance of the arriving task implies that other aperiodic tasks with lower priorities lose their guarantees, they will have to be moved elsewhere. After a predefined number of migrations without success, a task must be discarded. In order to evaluate the benefits of adding objectives to the Bayesian decision algorithm through the method proposed, the following schemes are applied to choose a destination node for an unguaranteed task:

- *Random:* A task is sent to a node selected randomly;
- *Pure Bayesian:* The utility of an action increases as the CET of the destination node decreases. A kind of preferred list technique [5] is also used in this scheme in order to improve the number of tasks executed in time. Each scheduler mantains a list of nodes arranged in increasing order of communication cost. When the scheduler identifies two or more actions with the same expected utility, the one chosen will be the one that transfers the task to the better ranked node in its preferred list. This mechanism improves the chances of a low transfer delay when a task is moved, while reducing the probability of more than one overloaded node sending tasks to the same underloaded node;
- *No transfer:* A task that cannot be executed in time by a node is simply discarded without any transfer attempt. This is a baseline scheme;
- *Perfect scheme with task transfer delay*: The scheduler knows exactly the current loading state of the entire system. The action chosen is always based on updated information. However, task transfer delays imply that the state of a node may change while transferring the task. This is also a baseline scheme.

The Bayesian decision-based algorithm extended with objectives through the method proposed does not employs preferred lists. All decisions in choosing nodes to transfer tasks to only depend on the utility function. The chosen objective set used for evaluation of the method, is defined as follows:

Objective I : Related to the timing constraints of the tasks. An unguaranteed task must be moved to a node able to execute it in time. This aim can be represented by a fuzzy set $\underset{\sim}{O_I}$ of actions that transfer a task to such a node. A possible membership function for this set is shown in (6):

$$\mu_{\underset{\sim}{O_I}}(a_i) = \begin{cases} 1 & \text{if } \ell - \theta_i > 0 \\ 0 & \text{otherwise,} \end{cases} \qquad (6)$$

where ℓ is the laxity of the task being transferred and θ_i is the CET of node i..

Objective II : Related to the loading profile of the system. A location policy may attempt to establish a system load profile, to improve the chances of finding a node that would satisfy the computational needs of incoming real-time tasks. Homogeneous load sharing exploits parallelism and therefore can reduce system average response time. A simple way of trying to achieve this kind of load distribution is moving unguaranteed tasks to a node with low activity. This aim can be represented by a fuzzy set $\underset{\sim}{O_{II}}$ of actions that transfer a task to such a node. A possible membership function for this set is defined as follows:

$$\mu_{\underset{\sim}{O_{II}}}(a_i) = 1 - \frac{\theta_i}{\max(\theta)}, \quad \theta \in \Theta_i \qquad (7)$$

However, as showed in [3], a greater number of tasks could be executed in time if the load were unequally distributed among system nodes through an appropriate strategy. Moving an unguaranteed task to a node with high activity is a way to achieve heterogeneous load sharing. This aim can be represented by a fuzzy set $\underset{\sim}{O_{II}}$ of actions that transfer a task to such a node. A possible membership function for this set is shown in (8):

$$\mu_{\underset{\sim}{O_{II}}}(a_i) = \frac{\theta_i}{\max(\theta)}, \quad \theta \in \Theta_i \qquad (8)$$

Objective III : Related to communication costs. A short delay in transferring messages and/or tasks decreases the probability of inconsistences between inferred and real node states. Moving tasks to a "near" node is a way of reducing delays. This aim can be represented by a fuzzy set $\underset{\sim}{O_{III}}$ of actions that transfer a task to nodes "close to the sender". A possible membership function for this set is defined as follows:

$$\mu_{\underset{\sim}{O_{III}}}(a_i) = 1 - \frac{q.com_i}{\max(com)}, \qquad (9)$$

where com_i is the cost of transferring the task to node i; max(com) is the greatest cost among the options of task migration; q is a real number in the interval [0,1]. A value of q close to 1 implies considering "distant" nodes as a very bad destination option. As the value of q gets closer to 0, the difference between "distant" and "near" nodes decreases.

Unless specified otherwise, the weight b of each objective is assumed to be 1, the value of q in expression (9) is 0.6, and objective II tries to achieve homogeneous load sharing.

The simulations were performed with a system having fourteen nodes organized as shown in Figure 1. This topology was chosen as to harden conditions as compared to the topology presented in [6]. Aperiodic tasks arrive at a node as a Poisson process with a given arrival rate (λ). The set of possible execution times required by the arriving tasks is represented as $T_{ex}=\{t_{ex1}, t_{ex2},...\}_p$, where each value occurs with the same probability p. Possible laxities are represented as $L=\{ \ell_1, \ell_2 ,...\}_{p^*}$, where p* is the probability of occurrence of each value. Every edge in the graph of Figure 1 implies a transfer delay in the communication between adjacent nodes. For state messages, the delay is computed as 1% of the average task execution time ($A(T_{ex})$). A task transfer delay is considered to be a fraction of the execution time of the task being transferred. Time measures are expressed in units of $A(T_{ex})$, which is normalized to 1.

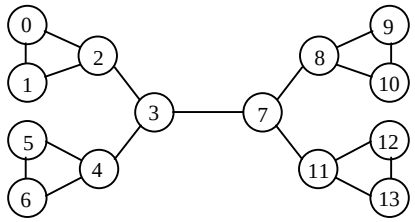

Fig. 1. Network topology

The performance evaluation of the method proposed considered the following effects:

- Varying task arrival rates (and, consequently, system load);
- Varying task transfer delays;
- Varying the weight of the objectives represented by the utility function.

For each set of input parameters, the simulation ran until it reached a confidence level of 95% in the results, for a maximum error of:

- 10% of the system dynamic failure ratio (R_{df}), defined as the total number of tasks that miss their deadlines divided by the total number of task arrivals. In Tables 4 and 5, however, the maximum error is 3% of R_{df};
- 1% of task transfer-out ratio (R_{tt}), defined as the number of tasks that must be transferred divided by the total number of task arrivals.

4.2. Results Analysis

R_{df} *versus* task arrival rates: Table 1 shows the system dynamic failure ratio as a function of task arrival rate λ for three different task sets and local scheduling FCFS.

Task set	λ	No transfer	Random	Bayes	Bayes proposed	Perfect
$T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$ $L=\{1,2,3\}_{1/3}$	0.2	1.930E-02	6.457E-04	3.321E-05	1.019E-05	5.232E-06
	0.4	5.207E-02	4.952E-03	5.441E-04	1.738E-04	8.222E-05
	0.6	9.917E-02	1.976E-02	2.947E-03	1.438E-03	7.738E-04
	0.8	1.571E-01	5.571E-02	1.478E-02	1.076E-02	7.768E-03
$T_{EX}=\{0.027, 0.27, 2.703\}_{1/3}$ $L=\{1, 2, 3\}_{1/3}$	0.2	5.766E-02	7.856E-03	1.989E-04	5.902E-05	4.606E-05
	0.4	1.211E-01	2.319E-02	4.759E-03	9.710E-04	8.361E-04
	0.6	1.866E-01	5.029E-02	1.373E-02	5.779E-03	5.457E-03
	0.8	2.498E-01	8.991E-02	2.839E-02	2.036E-02	1.200E-02
$T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$ $L=\{1\}_{1}$	0.2	4.706E-02	1.722E-03	1.275E-04	3.735E-05	1.923E-05
	0.4	1.049E-01	1.128E-02	1.779E-03	6.490E-04	3.251E-04
	0.6	1.685E-01	3.737E-02	8.569E-03	4.910E-03	2.817E-03
	0.8	2.331E-01	8.582E-02	3.236E-02	2.462E-02	1.812E-02

Table 1. R_{df} *vs.* task arrival rates for different task sets and local scheduling FCFS.

The same information presented in Table 1 is shown graphically in Figures 2 and 3, for the first and second task sets of Table 1. In all simulated cases, the proposed scheme outperforms the others presented in the literature (see [5] for previous comparisons), excepting the perfect scheme, in meeting task deadlines. Such results demonstrate the effectiveness of the method proposed in adding objectives to the Bayesian decision scheme.

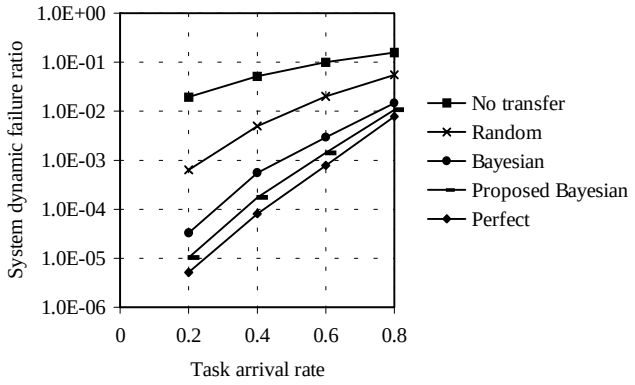

Fig. 2. R_{df} *vs.* task arrival rates for task set $T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$, $L=\{1,2,3\}_{1/3}$ and local scheduling FCFS.

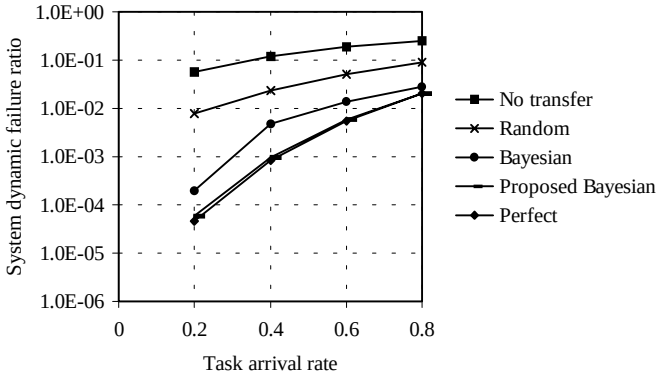

Fig. 3. R_{df} *vs.* task arrival rates for task set $T_{EX}=\{0.027, 0.27, 2.2703\}_{1/3}$, $L=\{1,2,3\}_{1/3}$ and local scheduling FCFS.

R_{df} *versus* task transfer delays: The effects of varying task transfer delays were evaluated considering them equal to 5%, 10%, 15%, 20% and 25% of the execution time of the task being transferred. As shown in Table 2 and Figure 4, all schemes have their performance degraded for higher task transfer delays. The Bayesian decision scheme is affected because of the probably greater inconsistency between the system load state, inferred at destination decision time, and the real state of the chosen node on task arrival. The random scheme is more robust to variations of task transfer delays: the destination node is always randomly chosen and the scheme does not need to infer the system state.

Local scheduling	Task transfer delay (percentage of task execution time)	Random	Bayes	Bayes proposed	Perfect
FCFS	0.05	4.557E-02	6.693E-03	5.283E-03	3.625E-03
	0.10	5.571E-02	1.478E-02	1.076E-02	7.768E-03
	0.15	6.775E-02	2.259E-02	1.675E-02	1.367E-02
	0.20	7.846E-02	3.060E-02	2.356E-02	2.055E-02
	0.25	8.815E-02	3.823E-02	3.061E-02	2.825E-02
MLFS	0.05	2.364E-02	2.722E-03	2.401E-03	2.203E-03
	0.10	3.296E-02	6.316E-03	4.712E-03	4.068E-03
	0.15	4.442E-02	1.204E-02	8.425E-03	7.630E-03
	0.20	5.480E-02	1.820E-02	1.340E-02	1.252E-02
	0.25	6.466E-02	2.469E-02	1.885E-02	1.811E-02

Table 2. R_{df} *vs.* task transfer delay for $T_{ex}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$, $L=\{1,2,3\}_{1/3}$, $\lambda=0.8$ and local scheduling FCFS/MLFS.

In order to minimize the problems caused by task transfer delays, the objective of moving tasks to "near" nodes was added to the Bayesian decision scheme. The results obtained show that the proposed scheme is more efficient in achieving such an objective than the preferred list technique.

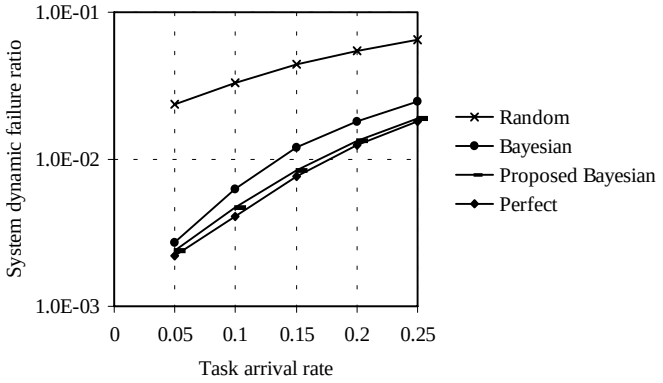

Fig. 4. R_{df} *vs.* task transfer delay for task set $T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$, $L=\{1,2,3\}_{1/3}$, $\lambda=0.8$ and local scheduling MLFS.

R_{df} *versus* weight of the objectives: Table 3 shows the system dynamic failure ratio as a function of the objectives weights for two task sets. In each column, only the weight of one objective is altered, the others being kept unchanged and equal to 1. Analysis of the results leads to the following conclusions:

- Objective II plays the role of objective I, when its aim is choosing a node with a low loading state (expression (7)). Even suppressing objective I, by making its weight equal to 0, does not increase R_{df} if the weight of objective II is kept at 1;
- Objective III has greater influence on the quality of task scheduling under the proposed scheme. Suppressing this objective causes a significant increase in R_{df}. Making b_{III}=0.3, 0.6 or 1 barely affects R_{df} because of the value chosen for the parameter q in the membership function shown in (9). In such an expression, for q=0.4, no action $a \in A$ has membership in $\underset{\sim}{O}_{III}$ lower than 0.6. It means that "distant" nodes are not considered a bad option to receive an unguaranteed task. Increasing b_{III} to values greater than a certain threshold does not change the fact that "distant" nodes are considered reasonable options for task migration. However, values of b_{III} lower than such a threshold decrease the impact of this objective in the final action decision, hence reducing the quality of the scheduling;
- Objective II has little influence on the scheduling quality when the aim is choosing a node with low loading state (expression 7) and objective III is added. Suppressing objective II makes the scheduler transfer tasks preferentially to the "nearest node" able to execute it in time. This fact implies that the task will arrive more quickly at the chosen destination, reducing the chances of state load changes during its transfer. Conversely, increasing b_{II} allows a task to be sent to a "distant" node. However, this will only occur when such node is at a very low loading state. It means that transfers to "distant" nodes are less vulnerable to state changes. Table 4 shows R_{df} *versus* b_{II} for b_I=1 and b_{III}=0. Suppressing objective III makes objective II assume a greater influence on the scheduling quality.

Task set	Weight of the objective (b)	Objective I	Objective II	Objective III
$T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$ $L=\{1,2\}_{1/2}$	0	2.414E-03	1.761E-02	2.282E-03
	0.3	2.394E-03	2.402E-03	2.280E-03
	0.6	2.366E-03	2.366E-03	2.280E-03
	1	2.366E-03	2.366E-03	2.320E-03
$T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$ $L=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$	0	3.067E-02	6.934E-02	2.708E-02
	0.3	3.072E-02	3.299E-02	2.689E-02
	0.6	3.078E-02	3.078E-02	2.772E-02
	1	3.078E-02	3.078E-02	3.078E-02

Table 3. R_{df} *vs.* weight of the objectives for different task sets and local scheduling FCFS. Just the weight of one objective is varied per time. Weights not altered are kept equal 1. Membership function for objective II given by expression (7).

Task set	Weight of the objective (b)	Objective II
$T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$ $L=\{1,2\}_{1/2}$	0	6.811E-03
	0.3	1.769E-02
	0.6	1.761E-02
	1	1.761E-02
$T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$ $L=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$	0	6.503E-02
	0.3	6.944E-02
	0.6	6.934E-02
	1	6.934E-02

Table 4. R_{df} *vs.* weight of the objective II for different task sets, b_I=1, b_{III}=0 and local scheduling FCFS.

For a discussion of the obtained results when Objective II is expressed through equation (8), see [2].

Task transfer-out ratio (R_{tt}): R_{tt} is a measure of the traffic overhead caused by task transfers and is defined as the number of tasks that must be transferred divided by the total number of task arrivals. Figure 5 and 6 show the task transfer-out ratio as a function of the task arrival rate for different task sets. The numerical results are listed in Table 5. All schemes present almost the same task transfer-out ratio for low loads (λ small) . In such a situation, most tasks can be guaranteed locally and do not need to be moved. Even if a transfer must take place, it will not be difficult to find a node able to execute the task in time. Nevertheless, when λ increases, the various schemes show their differences. For the task sets of Figures 5 and 6, the perfect scheme has a greater R_{tt} than the proposed scheme. This fact is a result of the objective imposed on the perfect mechanism: always choose the "nearest" node able to meet the deadline of the transferring task. Sometimes, the chosen node is in such a state that any change prevents it from guaranteeing the task to be received. If such a change occurs, more than one transfer will be necessary to find an adequate node for the task. The proposed scheme minimizes this problem by choosing the "nearest" node with the lowest loading state. This option makes migration success less sensitive to load changes on the chosen node. It can be shown that the R_{tt} produced by the perfect scheme is lower than that of the proposed scheme when laxities of the scheduled aperiodic tasks are more restrictive [2]. In such a situation, there will be a greater number of task transfers, and precision in defining a destination node is more critical.

For the task set of Figure 6, the Bayesian scheme with $u^{\ell}(\theta_i, a_i) = \ell - \theta_i$ generates a lower task transfer-out ratio when compared to the proposed scheme. This fact occurs due to the deficiency of the former in meeting the objective of sending tasks to "near" nodes: "proximity" is only considered as a tie-breaking criterion between equivalent alternatives (*i.e.*, with the same expected utility). Consequently, transfers to "nodes" with low load states take place despite the communication costs involved. The task set shown in Figure 6 implies a high transfer cost for the tasks with T_{ex}=2.703 (10% of its T_{ex} value per hop). This means that when moved to distant nodes, such tasks will miss their deadlines before arrival at the destination node. The tasks will then be immediatelly discarded without attempting a second transfer. Indeed, such behavior causes a decrease of the task transfer-out ratio. Nevertheless, the R_{df} in this situation is far more unfavorable to the Bayesian scheme with $u^{\ell}(\theta_i, a_i) = \ell - \theta_i$ when compared to the proposed scheme, as shown in Figure 3. If communication costs are reduced, it can be shown that the tasks moved will have a greater chance of being moved more than once, even if transferred to "distant" nodes [2]. In this case, the R_{tt} produced by the proposed scheme outperforms all the others. The results demonstrate the effectiveness of the proposed method in achieving lower traffic overhead.

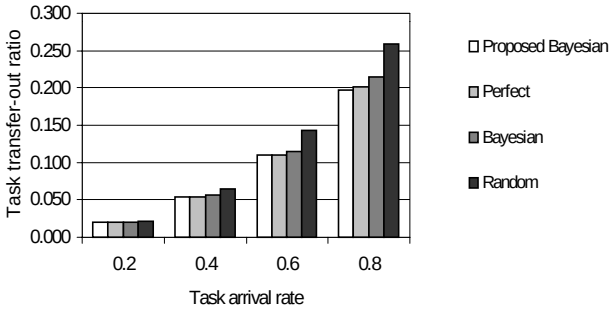

Fig. 5. R_{tt} *vs.* task arrival rates for task set $T_{EX}=\{0.4, 0.8, 1.2, 1.6\}_{1/4}$, $L=\{1,2,3\}_{1/3}$ and local scheduling FCFS.

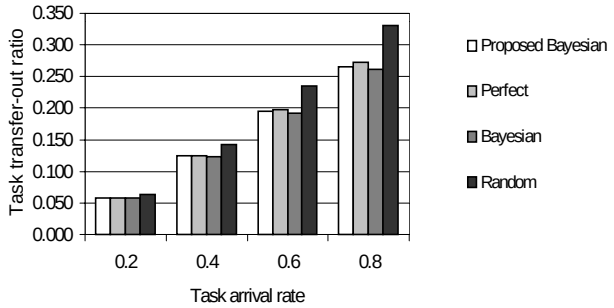

Fig. 6. R_{tt} *vs.* task arrival rates for task set $T_{EX}=\{0.027, 0.27, 2.703\}_{1/3}$, $L=\{1,2,3\}_{1/3}$ and local scheduling FCFS.

Atributos das Tarefas	**λ**	**Random**	**Bayes**	**Bayes Proposed**	**Perfect**
$T_{EX}=\{0.4 , 0.8 , 1.2 , 1.6\}_{1/4}$ $F=\{1, 2, 3\}_{1/3}$	0.2	0.021	0.020	0.020	0.020
	0.4	0.065	0.056	0.054	0.054
	0.6	0.143	0.115	0.110	0.110
	0.8	0.259	0.214	0.197	0.201
$T_{EX}=\{0.027, 0.27 , 2.703\}_{1/3}$ $F=\{1, 2, 3\}_{1/3}$	0.2	0.063	0.058	0.058	0.058
	0.4	0.143	0.123	0.124	0.125
	0.6	0.235	0.192	0.195	0.198
	0.8	0.331	0.261	0.266	0.273
$T_{EX}=\{0.027, 0.27 , 2.703\}_{1/3}$ $F=\{1, 2, 3\}_{1/3}$ % Custo com = 0,05	0.2	0.064	0.058	0.058	0.058
	0.4	0.155	0.124	0.124	0.125
	0.6	0.263	0.198	0.195	0.200
	0.8	0.379	0.292	0.281	0.294
$T_{EX}=\{0.4 , 0.8 , 1.2 , 1.6\}_{1/4}$ $F=\{1\}_{1}$	0.2	0.052	0.049	0.048	0.048
	0.4	0.130	0.116	0.113	0.113
	0.6	0.232	0.208	0.199	0.196
	0.8	0.344	0.333	0.311	0.309

Table 5. R_{tt} *vs.* task arrival rate for different task sets . Local scheduling employed: FCFS.

5. Conclusions

Using fuzzy set theory, we have proposed a method which easily maps task migration objectives into an useful mathematical expression, which may be applied to evaluate the transfer options of a distributed real-time system scheduler. In particular, the proposed method has been employed to improve the performance of a location policy algorithm based

on Bayesian theory through the definition of a meaningful utility function. In order to show the effectiveness of our approach, simulation studies were performed comparing different strategies to solve the distributed scheduling problem that were proposed in the literature. As discussed in the previous section, the results of these experiments indicate that a greater number of aperiodic tasks can attend their deadlines when the utility function is constructed by applying the multiobjective fuzzy decision method here described.

6. Acknowledgments

This work has been partially supported by the brazilian research funding agencies CNPq, Finep and Faperj.

7. References

[1] A. Barroso, J. Torreão, J. Leite, O. Loques and J. Fraga, "A new technique for task allocation in real-time distributed systems", Proceedings of the 7th Brazilian Symposium of Fault-Tolerant Computers, pp. 269-278, Campina Grande, Brazil, July 1997.

[2] A. Barroso, J. Leite, O. Loques, "On multiobjectives dynamic distributed scheduling of soft real-time tasks", CAA – Computação Aplicada e Automação, Universidade Federal Fluminense, Niterói, RJ, Brazil, Technical Report CAA RT-02/98 (www.caa.uff.br/reltec.html), July 1998.

[3] A. Bestravos, "Load profiling in distributed real-time systems", Technical Report 96-017, Department of Computer Science, Boston University, 1996.

[4] R. Garey and D. Johnson, "Complexity bounds for multiprocessor scheduling with resource constraints", SIAM J. Computing, vol. 4, no. 3, pp. 187-200, 1975.

[5] K. Shin and C. Hou, "Design and evaluation of effective load sharing in distributed real-time systems", IEEE Trans. on Parallel and Distributed Systems, vol. 5, no. 7, pp. 704-719, July 1994.

[6] J. Stankovic, "An application of Bayesian decision theory to decentralized control of job scheduling", IEEE Trans. Computer, vol c-34, no. 2, pp. 117-130, February 1985.

[7] J. Stankovic, K. Ramamrithan and S. Cheng, "Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems", IEEE Trans. Computer, vol. c-34, no. 12, pp.1130-1143, December 1985.

[8] J. Stankovic, "Strategic directions in real-time and embedded systems", ACM Computing Surveys, vol. 28, no. 4, pp. 751-763, December 1996.

[9] R. Yager, "A new methodology for ordinal multiobjective decision based on fuzzy sets", Decision Sci., vol. 12, pp. 589-600, 1981.