Effective Load Sharing in Distributed Real–Time Systems

Kang G. Shin Chao–Ju Hou

Real–Time Computing Laboratory
Department of Elec. Engr. and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109–2122.

Abstract

In a distributed real–time system, temporary uneven task arrivals among the nodes may cause some tasks to miss their deadlines even if the overall system has the capacity to meet the deadlines of all tasks. In this paper, an effective load sharing (LS) scheme is proposed as a solution to this problem.

Upon arrival of a task at a node, the node determines whether or not it can complete the task in time under the minimum–laxity–first–served policy. If the task cannot be guaranteed or if guarantees of some other tasks are to be violated due to the insertion of this task into the existing schedule, the node looks up the list of *loss–minimizing decisions*, and determines the best node among a set of nodes in its physical proximity, called its *buddy set*, to which the task(s) may be transferred. This list of decisions is periodically updated using Bayesian decision analysis and prior/posterior state distributions. These probability distributions are derived from the information collected via time–stamped state–region change broadcasts within each buddy set.

The performance of the proposed scheme is evaluated via simulation along with five other schemes. The proposed scheme is shown to outperform all but perfect LS scheme in (i) meeting task deadlines and (ii) tolerating the delays in state–information collection and task transfer.

1 Introduction

In a distributed real–time system, task arrivals may temporarily be uneven among the nodes and/or the processing power may vary from node to node, thus making some nodes temporarily overloaded while leaving others underloaded/idle. This calls for an effective method that enables underloaded nodes to share the loads of overloaded ones.

Load sharing (LS) in a distributed real–time system is different from that in a general–purpose system in that the latter tries to minimize average task response time, whereas the former is intended to minimize the probability of failure to complete each real–time task in time — this was termed the *probability of dynamic failure*, P_{dyn}, in [1]. Upon arrival of a real–time task, each node determines whether or not it can complete this task in time. If it can, the node will execute the task locally; otherwise, some other 'capable node' will be chosen to execute the task [2]. By 'capable node,' we mean a node with unused resources enough to complete transferred–in task(s) in time. As was discussed in [2], LS in a distributed system is dictated by two basic policies: the *transfer policy* for determining when to transfer a task, and the *location policy* for determining where to transfer the task. In the context of real–time applications, the transfer policy determines whether or not a task can be guaranteed locally, and the location policy determines which other node is most likely to guarantee the task to be transferred.

According to the properties of these two policies, LS schemes can be classified into three categories: deterministic, probabilistic and dynamic/adaptive. Both deterministic and probabilistic approaches do not use the state information, and thus, cannot react to dynamic situations. By contrast, an adaptive approach uses state information for their location policy. The state of a node may be the number (or queue length, QL), or the cumulative task execution time (CET), of tasks queued for execution on the node, the number and type of available resources, or a function or combination thereof. The node makes LS decisions based on the information collected via either periodic/aperiodic state broadcasting [3] or state probing or bidding [4, 5, 6].

Because an adaptive approach can adapt itself to dynamically–changing conditions, it is naturally expected to outperform non-adaptive approaches in meeting task deadlines. However, the required state probing/broadcasting could incur significant communication overheads, thus delaying the execution of tasks to be transferred. Moreover, the collected state information may be out–of–date due to the delay in collecting it. That is, a node's *observed* states of other nodes may be different from their *true* states at the time of making LS decisions. This difference often degrades the performance of adaptive LS as was analyzed in [5, 6].

To reduce the performance degradation caused by the delays in state collection and/or task transfer, we propose a new LS scheme using Bayesian decision theory as well as the concept of buddy sets, preferred lists, and state–change broadcasts in [3]. The basic ideas used here will be detailed in Section 2. Moreover, CET, instead of the commonly used QL, is used as the state of each node, since the former is more adequate for real–time applications. Using several performance metrics, we comparatively evaluate the proposed scheme along with five other schemes: no LS; LS with state probing, focused addressing and random selection; and perfect LS. As the simulation results indicate, the use of Bayesian analysis significantly reduces the performance degradation caused by the delays in collecting state information and transferring tasks — which, despite its importance, is seldom addressed in literature except for the authors of [5, 6] who analyzed the undesirable effect of communication delays without proposing any means to alleviate the problem.

The work reported here is supported in part by the Office of Naval Research under Contract N00014–85–K–0122 and the National Science Foundation under Grant No. DMC–8721492. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

0-8186-2310-1/91 $1.00 © 1991 IEEE

The rest of this paper is organized as follows. The basic ideas of the proposed scheme are described in Section 2. The Bayesian decision model used is presented in Section 3. How both the components of the Bayesian decision model and the concepts presented in [3] can be accommodated into our LS scheme is also described there. Section 4 describes how each node constructs prior/posterior probability distributions, and updates loss-minimizing decisions. In Section 5, we evaluate via simulation the performance of the proposed LS scheme along with five other schemes. The paper concludes with Section 6.

2 The Proposed Scheme

In order to reduce the overheads associated with state collection and task transfer, the LS scheme in [3] requires each node to collect and maintain the state information of only those nodes in its physical proximity, called a *buddy set*. When a node cannot guarantee a locally-arriving task, only those nodes in its buddy set are considered for transferring this task. In [3] four state regions determined by three thresholds of QL are used to characterize the workload of each node: underloaded, medium-loaded, fully-loaded, and overloaded. A node will broadcast the change of state region to the nodes in its buddy set only when it switches from underloaded to fully-loaded and vice versa. Based on the topological property of the system, each node orders the nodes of its buddy set into a *preferred list* such that a node is the k-th preferred node of one and only one other node, where k is some integer [3]. When a node is unable to guarantee a task, it will transfer the task to the first 'capable node' found in its preferred list. That is, the preferred lists are used as an effective means of selecting a receiver among several possible candidate nodes while minimizing the probability of more than one overloaded node simultaneously sending tasks to the same underloaded node.

Communication delays may still occur and thus degrade system performance unless the size of buddy set is kept very small, in which case the LS capability of the whole system may not be fully utilized. Thus, Bayesian decision theory is used to counter the communication delay problem.

Fig. 1 shows the actions of the scheduler on each node. Those tasks already queued at a node are sorted by their laxities and executed on a minimum-laxity-first-served basis. (Note that the laxity of a task is defined as the latest time a task must start execution in order to meet its deadline.) Upon arrival of a real-time task with laxity ℓ at a node, the scheduler checks if the CET (unlike in [3] where QL was used instead) on that node contributed by those tasks with laxity $\leq \ell$ is less than, or equal to, ℓ. If it is not, the new task must be transferred out, and the node's task queue remains unchanged; if it is, the new task is inserted into the task queue, and if this insertion leads to violation of existing guarantees, those tasks whose guarantees are violated need to be transferred to other capable nodes.

K state regions obtained from $K-1$ thresholds, TH_1, TH_2,...,TH_{K-1}, are used to describe the workload of each node.Each node will broadcast a time-stamped message, informing all the other nodes in its buddy set of a state-region change whenever its load crosses TH_{2k} for some k, where $1 \leq k \leq \lceil \frac{K}{2} \rceil - 1$. The reason for not broadcasting the change of state region whenever a node's load crosses any threshold is to reduce the network traffic resulting from region-change broadcasts. And the reason for not combining two adjacent state regions into one and then broadcasting the change of state region whenever a node's load crosses any threshold is to include finer information in each broadcast and thus construct more accurate distributions.

At each node:
When a task T_i with execution time E_i and laxity D_i arrives at the node:
  determine the position, j_p, in Q†such that $D_{j_p-1} \leq D_i \leq D_{j_p}$;
  if current_time + $\sum_{k=1}^{j_p-1} E_k \geq D_i$ **then**
  begin
    receiver_node := table_lookup ($\underline{x}$,D_i)‡;
    transfer task T_i to receiver_node;
  end
  else
  begin
    queue task T_i in position j_p;
    for $k = j_p + 1, length(Q)$
    begin
      if current_time + $\sum_{\ell=1}^{k-1} E_\ell \geq D_k$ **then**
      begin
        receiver_node := table_lookup ($\underline{x}$,D_k);
        dequeue and transfer T_k to receiver_node;
      end
    end
    if current_CET crosses TH_{2k} **then**
      /* $TH_1, \cdots, TH_{K-1}$ are thresholds */
      broadcast the state-region change to all nodes in its buddy set;
  end

When a broadcast message arrives from node i, $1 \leq i \leq n$:
  update observation of node i's state, x_i;
  record the (observation, true state) pair needed for constructing probability distributions;

At every clock tick:
  current_CET := current_CET − 1;
  if current_CET crosses TH_{2k} **then**
    broadcast state-region change to all nodes in its buddy set;

At every T_p clock ticks:
  update the probability distributions and the table of loss-minimizing decisions;

†The task queue Q is ordered by task laxities.
‡If a node anticipates, based on the current observation $\underline{x}$, that no other node can guarantee the task, this task is declared to be lost and discarded.

Figure 1: Operations of the task scheduler on each node.

By collecting time-stamped state samples and by keeping track of the corresponding observations at the times these samples were taken, each node can construct the prior/posterior distributions. These distributions characterize the inconsistency between the node's observed and true states of other nodes, and are used to periodically (once every T_p clock ticks) update the loss-minimizing decisions with Bayesian theory. As will become clear, the undesirable effects of the delay in broadcasting state-region changes/transferring tasks are eliminated by using these prior/posterior distributions. Whenever a node cannot guarantee a task, the node's scheduler looks up the list of loss-minimizing decisions, and choose — based on the current state information — the best candidate node for transferring this task such that the expected loss is minimized w.r.t. the posterior distribution.

3 Bayesian Decision Model

In this section, we shall describe how the proposed LS scheme can be cast into a Bayesian decision model.

3.1 Preliminaries

The elements of a Bayesian decision problem are a parameter space (a space of state of nature) Ω, a decision space D, and a real-valued loss function L which is defined on the product space $\Omega \times D$ [7]. For any point $(\omega, d) \in \Omega \times D$, the quantity $L(\omega, d)$ represents the loss when the value of the outcome W of the space Ω is ω and d is the decision chosen.

If P is any given probability distribution of the parameter W, then for any decision $d \in D$, the expected loss or risk, $\zeta(P, d)$, is given by

$$\zeta(P, d) = \int_{\Omega} L(\omega, d) dP(\omega). \tag{3.1}$$

We now want to choose a decision d which minimizes the risk $\zeta(P, d)$. The *Bayes risk* $\zeta^*(P)$ is defined to be the greatest lower bound for $\zeta(P, d)$ $\forall d \in D$. Any decision d^* whose risk is equal to the Bayes risk is called a *Bayes decision* w.r.t. P.

In many decision problems like the one we are going to discuss, before choosing a decision from D, we observe the value of a random variable X that is related to the parameter W. The essential component of problems of this kind is, in addition to the above elements, a family of sampling functions $\{f(. \mid \omega), \omega \in \Omega\}$ of observation X. Let S denote the sample space of all possible values of X. With the family of sampling functions and the (prior) probability distribution, P, of W, we can calculate the conditional distribution of W given X, $P_{W|X}$, as:

$$P_{W|X=x}(\omega) = \frac{P_W(\omega) P_{X|W}(x \mid \omega)}{\int_{\Omega} P_W(\omega) P_{X|W}(x \mid \omega) d\omega}. \tag{3.2}$$

Now, we must choose a decision function d which specifies, for every possible value $x \in S$, a decision $d \in D$ with the expected loss given the observation x as:

$$\zeta(P_{W|X=x}, d(x)) = \int_{\Omega} L(\omega, d(x)) dP_{W|X=x}(\omega). \tag{3.3}$$

Note that Eq. (3.3) is almost the same as Eq. (3.1) except that P has been replaced by $P_{W|X=x}$. Thus, any minimizing decision $d^*(x)$ is simply a Bayes decision against the conditional distribution of W when $X = x$.

3.2 Components of the Bayesian Decision Model

This subsection describes how to apply Bayesian decision theory to adaptive LS, and how to accommodate both the components of the statistical model and the concept of [3] into our scheme.

Parameter Space: the parameter space is defined as $\Omega = \Omega_1 \times \Omega_2 \times ... \times \Omega_n$, where n is the number of nodes in a buddy set, and Ω_i is the parameter space for node i. The parameter space, Ω_i, may be defined by QL, CET, resource available time (RAT) on node i, or a combination thereof, depending on task characteristics and performance requirements. Since execution time varies from task to task, we will define the state of a node to be its CET.

Probability Distribution on Parameter Space: the probability distribution on parameter space is the joint probability distribution of Ω_i's, e.g., $P_W(\underline{\omega}) = P_W(\omega_1, \omega_2, ..., \omega_n)$, where ω_i is the CET of node i. The marginal probability distribution on Ω_i, P_{W_i}, can be obtained from P_W by integration. We construct these probability distributions by collecting state samples through region-change broadcasts (to be discussed in Section 4).

Set of Available Decisions: the set of available decisions is $D = \{d_1, d_2, ..., d_n\}$, where d_i denotes the decision to move one task from the current node to node i.

Set of Loss Functions: the set of loss functions is defined as $\{L^{T_d}, T_d \in (0, T_{max}]\}$, where L describes the 'loss' resulting from each combination of state and decision, given that the laxity — which equals deadline − execution_time − current_time — of a locally unguaranteed task is T_d. T_{max} is the largest task laxity in the system. Since P_{dyn} is the main concern, the loss function may be defined as:

$$L^{T_d}(\underline{\omega}, d_i) = \delta(\omega_i - T_d) = \begin{cases} 1 & \text{if } \omega_i - T_d \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\delta(x)$ is the unit step function. In such a case, minimizing the expected loss is equivalent to minimizing the probability of dynamic failure.

Sample Space of Observation: the sample space of observation, S, is the set of all possible observations. Specifically, $S = S_1 \times S_2 \times \cdots \times S_n$, where S_i is obtained by dividing the parameter space W_i for each node i into the K regions determined by TH_1, TH_2,..., TH_{K-1}. Node i is said to be in the k-th region if $TH_k \leq \omega_i < TH_{k+1}$, where $k \geq 0$, and $TH_0 \triangleq 0$.

Note that the knowledge of a node's state region is not sufficient to determine accurately its capability of guaranteeing arbitrary tasks. For example, a node with its state in a high-numbered state region may still be able to guarantee an arriving task with a large laxity, whereas a task with a small laxity may not be guaranteed even by a first-region node if the CET on that node is greater than the task's laxity. Thus, unlike in [2, 3], these thresholds only serve as reference points, rather than indicating a node's capability of meeting task deadlines.

Each node will broadcast a time-stamped message, informing all the other nodes in its buddy set of a state-region change whenever its state crosses TH_{2k}, $1 \leq k \leq \lceil \frac{K}{2} \rceil - 1$. Upon receipt of a region-change broadcast, every node in the buddy set will update its observation of the broadcasting node, accordingly. The delay in broadcasting a region-change may cause inconsistency between the observed and true states of a node. We will characterize this inconsistency by constructing prior/posterior distributions (to be discussed in Section 4).

Family of Sampling Functions: the family of sampling functions, $\{f_{X|W}(. \mid \underline{\omega}), \underline{\omega} \in \Omega\}$, describes the conditional probability distribution of the observation X given the state $W = \underline{\omega}$. These probability distributions are derived from the samples gathered through time-stamped broadcasts. With the prior probability distribution P_W and these sampling functions $f_{X|W}$, one can derive the posterior probability distribution $P_{W|X}$ by using the Bayes rules [7] that is needed to compute the expected loss with observations.

4 Region-Change Broadcasting and Bayesian Analysis

The delay in region-change broadcasts may cause the collected information to be out-of-date. For example, consider the following scenario: after broadcasting a state-region change, say from 3 to 1, node i switches back to region 3 due to the arrival of new tasks and/or transferred-in tasks[1]. Upon receipt of the broadcast from node i, node j may decide to send a task to node i, since it is unaware that node i has switched back

[1] These tasks may have been sent by other nodes before the broadcast, but arrived at node i due to task-transfer delay.

to region 3 shortly after broadcasting the 3→1 region-change. If node j, instead of hastily believing in what it observed, can compute the probability that node i is indeed capable of guaranteeing task(s) and decide whether or not to send the task to node i, then P_{dyn} could be significantly reduced. To this end, we shall characterize the inconsistency between the observed and true states with prior/posterior distributions.

The first step is to construct both the probability distribution on the parameter space and the conditional probability distribution of an observation. These two distributions, in general, vary over both nodes and time in a dynamic environment. Thus, to monitor the dynamics of the system, each node must collect state samples on-line and construct these distributions from the samples gathered via region-change broadcasts. The methods for collecting state samples, constructing probability distributions, and deriving loss-minimizing decisions are discussed in the following subsections.

4.1 Collection of State Samples

Whenever a node's state crosses TH_{2k} $(1 \leq k \leq \lceil \frac{K}{2} \rceil - 1)$, the node will broadcast, to all the other nodes in its buddy set, a time-stamped message which contains node number i, the state ω_i^b before the change of state region, the state ω_i^a after the change, and the time t_0 when ω_i^b was sampled. When the message broadcast by node i arrives at node j, node i's state ω_i^b can be recovered by node j using the node number field and the state field, from which P_W can then be calculated. Node j can also trace back to find out its observation x_i at time t_0. This observation x_i is node j's observation of node i's state at the time when node i was actually in state ω_i^b. x_i's along with ω_i^b's are used to construct $f_{X|W}$. Any inconsistency between ω_i^b and x_i at time t_0 is characterized by this probability distribution. The only effect of the delays in task transfers and region-change broadcasts is that messages may not arrive at a node immediately after their broadcast, and thus, may become obsolete upon their arrival at other nodes. The correctness of all samples gathered is, however, not affected by these delays. Besides, ω_i^a sent by node i at time t_0 is considered as node j's new observation of node i at the time this message is received, rather than at time $t_0^+ > t_0$.

A primary advantage of region-change broadcasts over periodic state broadcasts is the elimination of the need to determine an "optimal" exchange period — a very difficult task since it depends on workload characteristics, and has to weigh the tradeoff between the resulting increase in network traffic and the negative effect of using out-of-date information.

4.2 Derivation of Probability Distributions

Each node updates, once every T_p units of time, the probability distributions using all the samples gathered so far, and re-calculates the loss-minimizing decisions. T_p should be chosen to reflect the fluctuation of system load and the number of samples required for the specified level of confidence in the results obtained.

The general rule for updating the probability distribution of W is $P_U = aP_T + (1-a)P_O$, where P_U is the updated probability distribution, P_T is calculated from the samples gathered over the last T_p units of time, and P_O is the old probability distribution. The ratio a $(0 < a \leq 1)$ represents the tradeoff between obtaining better averages and reflecting load changes. One may increase (decrease) a if system load varies rapidly (slowly). The same rule may be applied to update the sampling functions, $f_{X|W}$.

Non-informative probability distributions (e.g., uniform distributions) or some default probability distributions (obtained from previous experiences) may be used as the initial distribution of W and the sampling functions. According to our simulation results, the performance of the proposed scheme is found to be rather insensitive to the choice of an initial probability distribution. Each node may initially rely on the preferred list for LS decisions. This is because the prior/posterior distributions will be iteratively updated as time goes on, and usually represents the true system characteristics after two or three updates.

4.3 Calculation of Loss-Minimizing Decisions

With the prior distribution of W and the sampling function, $f_{X|W}$, one can calculate the posterior distribution $P_{W|X}$ using the Bayes rule (Eq. (3.2)). For each possible observation $\underline{x} \in S$ and for each possible laxity $T_d \in (0, T_{max}]$, a node then computes the expected loss associated with the decision d_i given the observation $\underline{x}$ and the laxity T_d as:

$$\zeta^{T_d}(P_{W|X=\underline{x}}, d_i) = \int_{\Omega} L^{T_d}(\underline{\omega}, d_i) dP_{W|X}(\underline{\omega}) \qquad (4.1)$$

for $i = 1, ..., n$. The decision $d_i = d^{T_d}(\underline{x})$ that yields the minimum expected loss is chosen as the optimal decision given the observation $\underline{x}$. A tie will be broken by choosing, from the preferred list, the first d_i with the minimum expected loss.[2] Because of the way $L^{T_d}(\underline{\omega}, d_i)$ was defined and the assumption that W_i is stochastically independent of the state of node j for $j \neq i$ [2], the computation of the expected risk, $\zeta^{T_d}(P_{W|X=\underline{x}}, d_i)$, depends only on the marginal probability distribution, $P_{W_i|X_i}(\omega_i)$. That is, if $L^{T_d}(\underline{\omega}, d_i) = \delta(\omega_i - T_d)$, then

$$\begin{aligned}
\zeta^{T_d}(P_{W|X=\underline{x}}, d_i) &= \int_{\Omega} \delta(\omega_i - T_d)\, p_{W|X=\underline{x}}(\underline{\omega}) d\underline{\omega} \\
&= \int_{\Omega_i} \delta(\omega_i - T_d)\, p_{W_i|X=\underline{x}}(\omega_i) d\omega_i \\
&= \int_{\Omega_i} \delta(\omega_i - T_d)\, p_{W_i|X_i=x_i}(\omega_i) d\omega_i \\
&= P_{W_i|X_i}(W_i > T_d \mid X_i = x_i). \qquad (4.2)
\end{aligned}$$

In other words, the expected loss of adopting decision d_i given the observation $\underline{x}$ and the task laxity T_d is the probability that node i's CET is greater than T_d. The second equality in Eq. (4.2) follows from the property of total probabilities, and the third equality results from the assumption that W_i is stochastically independent of the observation X_j of node j, $j \neq i$, i.e., $p_{W_i|(X_1,X_2,...,X_n)}(\omega_i) = p_{W_i|X_i}(\omega_i)$. The set of loss-minimizing decisions, $\{d^{T_d}(\underline{x}) : \underline{x} \in S,\ T_d \in (0, T_{max}]\}$, is a list of decisions to choose for each possible observation and each possible task laxity. Once these calculations are completed, a task scheduler only needs to look up a table when determining a LS decision for a given observation $\underline{x}$.

5 Numerical Examples

To demonstrate the effectiveness of the proposed LS scheme, we carried out simulations for task sets with both exponential and hyperexponential interarrival times of external tasks. The proposed scheme and three other LS schemes are comparatively

[2] The nice property (of the preferred lists) in distributing unguaranteed tasks among capable nodes is thus maintained in the proposed scheme.

evaluated. The schemes under consideration differ in the way a node treats locally unguaranteed tasks as follows:

- The state probing scheme: a node with an unguaranteed task randomly probes up to some predetermined number of nodes and transfers the task to the first capable node found during the probing.
- The random selection scheme: each locally unguaranteed task is sent to a randomly selected node.
- The focused addressing scheme: A node sends its unguaranteed task to a node (called the *focused* node) which is randomly selected among those nodes 'seen' to be capable of guaranteeing the task. node itself.) Meanwhile, the node also sends request-for-bid (RFB) messages to all the other nodes in the system, indicating that bids should be returned to the designated focused node. If the focused node cannot guarantee the task, it chooses, based on the bids received, a capable node for transferring the task; otherwise, the task is queued on the focused node, and the received bids are used to locate the receiver nodes for those tasks, if any, whose guarantees become invalid as a result of accepting the transferred task. The bids received at the focused node are also used to update the observation of other nodes' states. When a task arrives at a node after its bid has been accepted, the node will check again whether or not the task can be guaranteed. This is a simplified version of the scheme proposed in [4]. It also differs slightly from that of [4] in the way a node chooses the focused node. The authors of [4] used the percentage of free time during the next window (which is a design parameter) and many other estimated parameters to determine the focused node or the node to which the task must be transferred again. However, we use the observed CET of other nodes to determine the node(s) for transferring tasks.
- The proposed scheme: a node sends each unguaranteed task to another node in its buddy set based on a technique that combines preferred lists, state-region change broadcasts, and Bayesian analysis.

These schemes are compared with one another as well as with two other baseline schemes. The first baseline scheme assumes no load sharing, while the second is an idealistic proposed scheme where each node has complete information on the workload of other nodes without any overheads in collecting it.

A 16-node regular system[3] is used as an example for the simulations. For convenience, all time-related parameters are expressed in units of average task execution time. The size of buddy set is chosen to be 10, since the performance improvement by increasing it beyond 10 was shown in [3] to be insignificant. The maximum number of nodes to be probed randomly for each locally unguaranteed task is restricted to 5 based on the finding in [2]. The computational overhead for each bidding, state probing, region-change broadcast, and probability distribution update is assumed to be 1, 1, 1, and 2 % of $E(R)$, respectively.

Each communication medium/link is equipped with buffers, and transferred tasks or broadcast messages are queued and/or transmitted in order of their arrival. That is, no priority mechanism regulates the transmission over the medium. Unless specified otherwise, the delay associated with each task transfer is assumed to be 10 % of the execution time of the task being transferred. The queueing delay due to task transfers, region-change broadcasts, requests and responses for bids, and state probes dynamically changes with system load and traffic, and is modeled as a linear function of the number of tasks/messages queued in the particular medium/link.

Let q_i $(1 \leq i \leq m)$ and $\hat{q}_j$ $(0 \leq j \leq T_{max})$ represent the probability that an external task requires i units of time to execute and that a task has laxity of j units of time, respectively. For notational convenience, $\{e_1, e_2, ..., e_k\}_{\{q_{e_1}, q_{e_2}, ..., q_{e_k}\}}$ is used to denote the task set in which a task requires execution time e_i with probability q_{e_i}, $\forall i$. If $q_{e_i} = q \; \forall e_i$, then $\{q_{e_1}, q_{e_2}, ..., q_{e_k}\}$ is condensed to q. Similarly, $\{\ell_1, \ell_2, ..., \ell_n\}_{\{q_{\ell_1}, q_{\ell_2}, ..., q_{\ell_n}\}}$ is used to describe the distribution of task laxity. The simulation was carried out for a task set with the external task arrival rate on each node varying from 0.2 to 0.9, the ratio of $\frac{e_{j+1}}{e_j}$ $(1 \leq j \leq k-1)$ varying from 2 to 10, and the ratio of of $\frac{\ell_{j+1}}{\ell_j}$ $(1 \leq j \leq n-1)$ varying from 2 to 6. Due to space limitation, we present only representative results. However, the results are found to be quite robust in the sense that the conclusion drawn from the performance curves for a task set with the given task execution and laxity distributions is valid over a wide range of combinations of task execution time and laxity distributions.

For each combination the simulation ran until it reached a confidence level 95% in the results for a maximum error (e.g., one half of the confidence interval) of (1) 2% of the specified probability if P_{dyn} is the measure of interest, and (2) 5% of the ratio or frequency value if task transfer-out ratio or frequency of broadcasts/state probes is the measure.We first determine the tunable parameters used in the proposed scheme. Second, we evaluate and compare different schemes with respect to several important performance metrics obtained from the simulations. We also analyze (1) the impact of varying communication overheads on the performance of these schemes; and (2) the impact of statistical fluctuation in task arrivals on the performance of the proposed scheme.

5.1 Determination of Tunable Parameters

The accuracy of prior/posterior distributions depends on the values of such tunable parameters as the probability update interval T_p, the probability update ratio a, and the number (K) and values of state-region thresholds. It is, however, difficult to objectively determine an optimal combination of these parameters which will give accurate prior/posterior distributions while incurring the least overhead. Thus, we shall determine the tunable parameters for each task set with the following two steps:

S1. We first fix all but one parameter of interest at a time, and obtain the performance curve as a function of this parameter from which its optimal value can be determined. Next, we vary another parameter of interest while keeping the first parameter fixed at its optimal value and the rest of the parameters fixed at their originally chosen values. This process will be repeated until all the parameters have been varied.

S2. Since a different order of examining parameters in **S1** may lead to different results, there may be more than one parameter set from which we choose the one with the smallest P_{dyn} and at the same time, reasonably small processing/communication overheads as the 'optimal' parameter set.

[3] A system is said to be *regular* if all node degrees are identical.

The sets of parameters obtained through the above two steps may not be globally optimal, but our simulation results have shown them to yield good results as compared to other schemes. Moreover, our simulation results indicate that the proposed scheme is robust to the variation of the tunable parameters, as compared to the other schemes reported in [3, 6]. The change in P_{dyn} is shown to be less than 10^{-3} for any given change in either the threshold interval, or the number of state regions, or the values of thresholds. This robustness is an important advantage coming from the use of prior/posterior distributions and Bayesian analysis. The proposed LS scheme can thus provide good performance even with not well-tuned parameters as long as the general rules discussed above are followed.

5.2 Performance Evaluation

5.2.1 Probability of Dynamic Failure

A dynamic failure occurs if the sum of its queueing-for-execution time and the delay in transferring the task exceeds its given laxity. Let $P_{dyn|d}$ denote the probability of missing deadlines for a task with laxity d. Then, $P_{dyn} = \sum_{j=0}^{T_{max}} P_{dyn|j}\ \hat{q}_j$. Figs. 2 and 3 are the plots of P_{dyn} vs. external task arrival rate (λ) and $P_{dyn|d}$ vs. task laxity, respectively. Table 1 shows some numerical results of $P_{dyn|d}$ under different schemes. As was expected, P_{dyn} increases as the system load gets heavy and/or the task laxity gets tight.

The random selection scheme outperforms the state probing scheme when the system load gets heavy or the task laxity gets tight, e.g., $L = \{1, 2, 3\}$ as compared to $L = \{1\}$ in Table 1(b). This is because (1) under heavy loads, most nodes are likely to become unable of guaranteeing tasks, which will in turn make state probing unsuccessful most of the time, and (2) probing other nodes before sending an unguaranteed task requires two communication messages (one for request and the other for response), whereas the random selection does not require such messages. This negative effect becomes more pronounced as timing constraints get tighter.

The focused addressing scheme outperformed the state probing and random selection schemes, but was inferior to the proposed scheme, especially when the task laxity is tight. This is because:

- Not many RFB messages are issued under light loads, thus making a node unable to keep its observation of other nodes up-to-date and increasing the chance of transferring a task to an incapable focused node. This becomes intolerable for tasks with tight laxities.
- Requests and replies for bids become excessive under heavy loads, thus increasing communication delays. The state information collected via periodic state exchange or the bids sent from other nodes may become out-of-date.

In all cases simulated, the proposed LS scheme is shown to to outperform all but perfect information scheme in meeting task deadlines, demonstrating its effectiveness achieved by using the judicious collection and use of state information. It does not perform as well as the perfect information scheme due to the additional processing overhead introduced by the probability update process and the communication delays incurred in task transfers and region-change broadcasts.

5.2.2 Maximum System Utilization

The system utilization is defined as the ratio of the external (exponential) task arrival rate (λ) to the system service rate ($1/E(R)$). The service rate is normalized to 1 in our analysis, and thus, the system utilization simply becomes λ. Since P_{dyn} increases with system load (Fig. 2), there exists an upper bound for λ, termed as *maximum system utilization* λ_{max}, below which $P_{dyn} \leq \epsilon$ can be guaranteed for some prespecified $\epsilon > 0$. Fig. 4 shows plots of the maximum system utilization versus ϵ for a 16-node regular system. One important result is that we do not have to sacrifice system utilization to lower P_{dyn}, which is in contrast to the common notion of trading system utilization for real-time performance.

5.2.3 Frequency of Information Exchange

In the proposed LS scheme, each node has to broadcast the change of state regions to all the other nodes in its buddy set. Thus, the frequency of region-change broadcasts, f_b, determines the traffic overhead in collecting state information. In both the state probing and focused addressing schemes, on the other hand, the traffic overhead is determined by the frequency of state probing, f_p, and the frequency of RFB, f_r, respectively. Table 2 summarizes the simulation results on f_b, f_p, and f_r in terms of number of messages per $E(R)$.

Under light to medium loads (e.g., $\lambda = 0.2$—0.6 in Table 2(a)), the proposed scheme introduces more traffic overhead (in the worst case, about 0.5 broadcast per $E(R)$) than the other two schemes, but only about 2%—13% of the arriving tasks are transferred to other nodes. Under light to medium loads, the additional traffic introduced by broadcasts does not impede the transmission of tasks and/or other messages. Besides, the effect of the increased communication delay (as a result of broadcasts) on the inconsistency between a node's observed and true states of other nodes is taken care of by Bayesian analysis.When the system load is heavy, the state probing scheme and the focused addressing scheme perform worse than the proposed scheme (Table 2(b)). This phenomenon becomes more pronounced when the variance of task execution time is large or when the task laxity is tight.

5.2.4 Sensitivity to Communication Delays

There are two types of communication delay to consider: one is medium-queueing delays (queueing-related costs), and the other is the transmission delay (transmission costs) associated with task transfers. To study the effect of communication delays, P_{dyn} was computed with (1) the transmission cost associated with task transfers of 5, 10, 15, and 20 % of the task execution time, and (2) the queueing-related costs being halved, doubled, and tripled.

As shown in Fig. 5, the state probing scheme, the random selection scheme, and the focused addressing scheme are all more sensitive to the variation of the transmission cost than the proposed scheme. The performance degradation by the state probing scheme occurs because, as the task transmission delay increases, other tasks may arrive at a probed node during the period between the time it was probed and the time an unguaranteed task (of the probing node) arrives at that node. Thus, there is not much correlation between the state when a node was probed and the state when an unguaranteed task arrived at the node. (Similarly, one can reason about the performance degradation of the focused addressing scheme.) The performance of the random selection scheme degrades as the transmission delay increases, due to the combined effect of higher task transfer-out ratios and large transmission costs. Varying queueing-related costs has a similar effect as varying transmission costs on the performance of LS.

In contrast, our proposed scheme is less sensitive to the communication delays (both queueing delays and transmission delays) because of the use of prior/posterior distributions to characterize the correlation between the observation made by a node and the corresponding true state.

5.2.5 Effect of Statistical Fluctuation on LS

One issue in using a Bayesian decision model is to what extent the proposed scheme remains effective when the system state randomly fluctuates. This effect is evaluated by simulating different task sets with hyperexponential external task interarrival times. This represents a system potentially with bursty task arrivals, and the degree of state fluctuation over short periods is modeled well by varying the coefficient of variation (CV) of the hyperexponential external task interarrival times. Fig. 6 shows the simulation results under heavy system load ($\lambda = 0.8$) where the LS performance is sensitive to the variation of CV. From Fig. 6, we draw the following conclusions: (1) the two curves labeled as the proposed scheme and the proposed scheme without the use of Bayesian analysis give another evidence that LS does benefit from the use of Bayesian decision theory; the undesirable effect of using outdated state information is alleviated; (2) the performance of the proposed scheme degrades as CV increases. However, the proposed scheme remains effective up to CV=5.48 (or $CV^2 = 30$) beyond which it reduces essentially to the scheme without the use of Bayesian decision analysis.

6 Conclusion

Using prior/posterior distributions and Bayesian analysis, we proposed a new LS scheme which can estimate, even with outdated state information, the workload of other nodes, and select the best candidate receiver of each unguaranteed task. The P_{dyn} as a result of using outdated information is thus reduced significantly. Moreover, the ability of making Bayesian decisions based on imperfect state information makes this scheme insensitive to communication delays. The proposed scheme is also shown to be robust to the variation of tunable parameters used in adaptive LS and effective within a wide range of statistical fluctuation in external task interarrival times.

References

[1] K. G. Shin, C. M. Krishna, and Y.-H. Lee, "A unified method for evaluating real-time computer controllers its application," *IEEE Trans. on Automatic Control*, vol. AC-30, pp. 357–366, April 1985.

[2] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems," *IEEE Trans. on Software Engineering*, vol. SE-12, no. 5, pp. 662–675, 1986.

[3] K. G. Shin and Y.-C. Chang, "Load sharing in distributed real-time systems with state change broadcasts," *IEEE Trans. on Computers*, vol. C-38, no. 8, pp. 1124–1142, August 1989.

[4] K. Ramamritham, J. A. Stankovic, and W. Zhao, "Distributed scheduling of tasks with deadlines and resource requirements," *IEEE Trans. on Computers*, vol. C-38, no. 8, pp. 1110–1141, August 1989.

[5] R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Analysis of the effect of delays on load sharing," *IEEE Trans. on Computers*, vol. C-38, no. 11, pp. 1513–1525, November 1989.

[6] R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Adaptive load sharing in heterogeneous systems," *IEEE Proc. 9th Int'l Conf. on Distributed Computing Systems*, pp. 298–306, 1989.

[7] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*, Springer-Verlag, 1986.

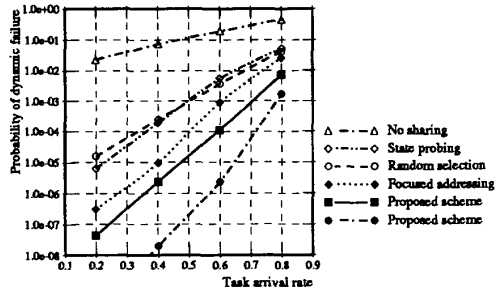

Figure 2: P_{dyn} vs. task arrival rate for a 16–node system with a task set: $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{1/3}$

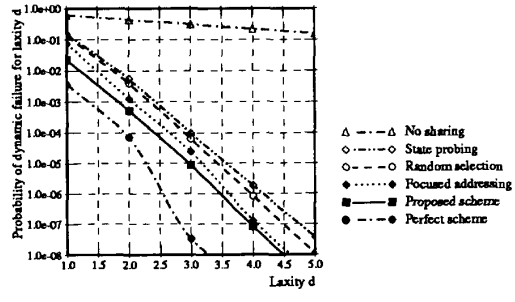

Figure 3: $P_{dyn|d}$ vs. task laxity d for a 16–node system with a task set: $\lambda = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3, 4, 5\}_{0.2}$)

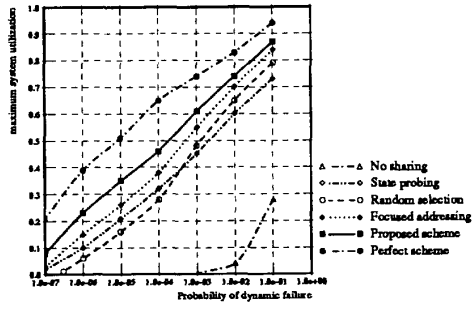

Figure 4: λ_{max} vs. P_{dyn} for a 16–node system.

($\lambda = 0.8$) Task attributes	Lax. d	No sharing	State probing	Random selection	Focused addressing	Proposed scheme	Perfect scheme
$ET = \{0.4, 0.8,$	1	0.6107	0.1515	0.1214	8.649×10^{-2}	2.123×10^{-2}	5.093×10^{-3}
$1.2, 1.6\}_{0.25},$	2	0.4317	4.779×10^{-3}	2.162×10^{-3}	9.746×10^{-4}	3.523×10^{-4}	6.492×10^{-5}
$L = \{1, 2, 3\}_{1/3}$	3	0.3058	3.514×10^{-5}	1.231×10^{-5}	1.026×10^{-5}	7.828×10^{-6}	5.721×10^{-8}
$ET = \{0.4, 0.8,$ $1.2, 1.6\}_{0.25},$ $L = \{1\}$	1	0.6075	0.1293	8.016×10^{-2}	7.153×10^{-2}	2.583×10^{-2}	6.012×10^{-3}

(a) $\lambda = 0.8$

($\lambda = 0.4$) Task attributes	Lax. d	No sharing	State probing	Random selection	Focused addressing	Proposed scheme	Perfect scheme
$ET = \{0.4, 0.8,$	1	0.1612	3.594×10^{-4}	7.293×10^{-4}	8.264×10^{-5}	1.391×10^{-5}	5.892×10^{-8}
$1.2, 1.6\}_{0.25},$	2	0.0421	5.402×10^{-6}	1.262×10^{-5}	9.536×10^{-7}	2.930×10^{-7}	1.583×10^{-10}
$L = \{1, 2, 3\}_{1/3}$	3	0.0117	1.782×10^{-7}	4.296×10^{-7}	4.846×10^{-8}	7.497×10^{-9}	0
$ET = \{0.4, 0.8,$ $1.2, 1.6\}_{0.25},$ $L = \{1\}$	1	0.1660	8.163×10^{-4}	6.250×10^{-4}	3.818×10^{-4}	7.018×10^{-5}	9.476×10^{-8}

(b) $\lambda = 0.4$

Table 1: $P_{dyn|d}$ vs. task laxity d for different task sets under different schemes ($N = 16$).

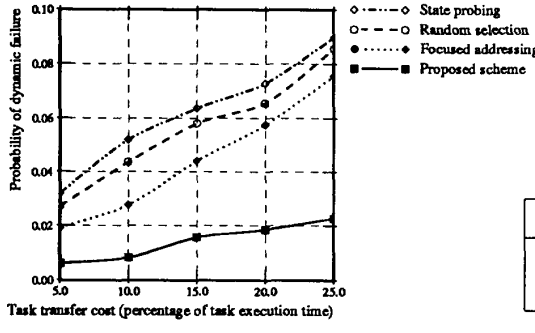

Figure 5: P_{dyn} vs. task transfer costs for a 16-node system with a task set: $\lambda = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{1/3}$.

Task arrival Rate (λ)	State probing Freq. of probes	Foc. addressing Freq. of requests	Prop. scheme Freq. of b-casts
0.2	0.0041	0.0178	0.2948
0.4	0.0287	0.2458	0.4517
0.6	0.1144	0.6821	0.4810
0.8	0.4836	1.2346	0.4943

(a) Frequency of state-collection vs. different λ for a task set with $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$ and $L = \{1, 2, 3\}_{1/3}$

($\lambda = 0.8$) Task attributes	State probing Freq. of probes	Foc. addressing Freq. of requests	Prop. scheme Freq. of b-casts
$ET = \{0.4, 0.8,$ $1.2, 1.6\}_{0.25},$ $L = \{1, 2, 3\}_{1/3}$	0.4836	1.2346	0.4943
$ET = \{0.4, 0.8,$ $1.2, 1.6\}_{0.25},$ $L = \{1\}$	0.9061	1.5023	0.6872

(b) Frequency of state information collection versus different task sets when $\lambda = 0.8$

Table 2: Comparison of the traffic overhead associated with collecting state information between the state probing and the proposed schemes.

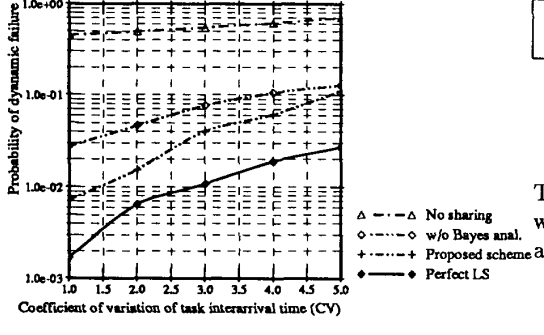

Figure 6: P_{dyn} vs. coefficient of variation of task interarrival times for a 16-node system with a task set: $\lambda = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{1/3}$.