Incorporation of Optimal Timeouts into Distributed Real–Time Load Sharing

Chao-Ju Hou Kang G. Shin

Real–Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122

Abstract

This paper addresses the problem of designing and incorporating a timeout mechanism into load sharing (LS) with state-region change broadcasts in the presence of node failures in a distributed real–time system. Failure of a node is diagnosed by the other nodes through communication timeouts, and the timeout period used to diagnose whether a node is faulty or not depends on the dynamic changes in system load, the task attributes at the node, and the state the node was initially in. We formulate the problem of determining the 'best' timeout period $T_{out}^{<i>}$ for node i as a hypothesis testing problem, and maximize the probability of detecting node failures subject to a prespecified probability of falsely diagnosing a healthy node as faulty.

The parameters needed for the calculation of $T_{out}^{<i>}$ are estimated on–line by node i using the Bayesian technique and are piggybacked in its region–change broadcasts. The broadcast information is then used to determine $T_{out}^{<i>}$. If node n has not heard from node i for $T_{out}^{<i>}$ since its receipt of node i's broadcast, it will consider node i failed. On the other hand, each node n also determines its own timeout period $T_{out}^{<n>}$, and broadcasts its state not only at the time of state–region changes but also when it has remained within a broadcast-threshold interval throughout $T_{out}^{<n>}$.

Our simulation results show that the LS algorithm which combines the on–line parameter estimation, the timeout mechanism, and a few extra, timely broadcasts can significantly reduce the probability of missing task deadlines.

1 Introduction

In a distributed real–time system, tasks may arrive unevenly and randomly at the nodes and/or computation power may vary from node to node, thus getting some nodes temporarily overloaded while leaving others idle or under-loaded. Consequently, some tasks may miss their deadlines even if the overall system has the capacity to meet the deadlines of all tasks. Many load sharing (LS) algorithms have been proposed to counter this problem, especially aiming at minimizing the probability of tasks missing their deadlines, or, the *probability of dynamic failure*, P_{dyn}.

The work described in this paper was supported in part by the NSF under Grant No. MIP–9012549 and the ONR under Grants N00014-85-K-0122 and N00014-92-J-1080.

Upon arrival at a node of a real–time task with laxity ℓ,[1] real–time LS algorithms determine whether or not the node can guarantee (complete in time) the task under some local scheduling discipline. The minimum–laxity–first–served (MLFS) discipline is shown in [1] to, on average, outperform others in reducing P_{dyn}, and is hence commonly used as a local scheduling discipline. That is, the cumulative execution time (CET) contributed by those tasks with laxity $\leq \ell$ on node i determines the node's capability to meet the deadlines of these tasks. If a node cannot complete a newly-arrived task in time or some of its existing guarantees are to be violated as a result of inserting the task into its schedule, the node has to determine — based on some state information — candidate receiver(s) for task transfer(s).

The state information required for all dynamic LS algorithms can be collected through periodic exchange of state information, bidding/state probing [2, 3], or aperiodic state–region change broadcasts [4, 5]. The algorithms based on the periodic exchange of state information require a good or optimal means of determining the period of information exchange, since the accuracy of state information when a LS decision has to be made depends heavily on this period. On the other hand, the algorithms based on bidding/state probing generates at least two additional messages per bidding/probing. Consequently, the performance of these algorithms is very sensitive to the variation of communication delay.

For LS algorithms using state–region change broadcasts, each node i broadcasts a message, informing the other nodes in its buddy set of a stage–region change *whenever* its CET crosses a certain broadcast threshold [5]. Such algorithms have the advantage of maintaining more up–to–date state information and collecting it inexpensively before it is needed for a LS decision. However, there still remain several potential problems for this kind of algorithms as follows.

- The communication overhead may become excessive as the system load gets heavy or as the number of communicating nodes in the system gets large.
- The state information gathered may still be out–of–date if the queueing/task–transfer delay is large.
- The performance is susceptible to node failures. If node

[1] The laxity of a task is defined as the latest time a task must start execution in order to meet its deadline.

0-8186-1060-3425/93 $03.00 © 1993 IEEE

i has been silent (i.e., does not broadcast its state–region changes) for a long time, other nodes have no way of knowing whether this is an indication of node i's failure or a coincidence of task arrival and completion/transfer activities alternating on node i.

Shin and Chang [4] proposed the concept of the buddy sets and the preferred lists to reduce the undesirable effects of the first problem. In another paper [5], we proposed a decentralized, dynamic LS algorithm which significantly alleviates the second problem in the presence of non-negligible communication delays. In this paper, we will design a timeout mechanism that can be incorporated into LS with aperiodic state–region change broadcasts to counter the third problem.

Specifically, each node n makes the transfer and location decisions as specified by the LS algorithm. In addition, node n considers node i failed if it has not heard from node i for the timeout period, $T_{out}^{<i>}$, since its receipt of node i's latest broadcast, and will henceforth not send its overflow task(s) (i.e., the tasks that cannot be completed in time) to node i even if node i is observed to be capable of completing the task(s) in time. Obviously, the determination of $T_{out}^{<i>}$ is crucial to the performance of the timeout mechanism, and is the main subject of this paper.

There are two possible scenarios of node n not receiving any region–change broadcast from node i for $T_{out}^{<i>}$:

S1. Node i failed sometime after issuing its last broadcast message;

S2. Task arrival and completion/transfer activities alternate in such a way that the state or CET of node i remains in a broadcast-threshold interval.

The determination of $T_{out}^{<i>}$ thus involves a tradeoff between the performance improvement gained by reducing $T_{out}^{<i>}$ (thus enabling early detection of a node failure) and the performance degradation resulting from hasty, incorrect diagnoses. We will formulate this problem as a hypothesis testing (HT) problem, and determine $T_{out}^{<i>}$ by maximizing the probability of detecting node failures subject to a prespecified probability of incorrect diagnosis.

To further reduce the probability of incorrect diagnosis, each node n calculates the "best" timeout period for itself as well as for other nodes, and broadcasts its state not only at the time of state–region changes but also when it has remained within a broadcast-threshold interval for $T_{out}^{<n>}$.

One factor that complicates the design of a timeout mechanism is that the task arrival and completion/transfer activities on a node (and thus the optimal value of $T_{out}^{<i>}$) dynamically vary with the system load, the task attributes, and the initial state of the node. Thus, the calculation of $T_{out}^{<i>}$ calls for on–line estimation of the parameters related to task attributes on node i. So, the proposed timeout mechanism requires each node i to collect statistics, estimate on–line its "composite"[2] task arrival rate and distributions of task execution time and laxity, and convey the estimated parameters to other nodes in its buddy set by piggybacking them in state–region change broadcasts. This information will then be used by the other nodes to calculate $T_{out}^{<i>}$.

The LS algorithm in [5] will be used here as an example to demonstrate how to incorporate the proposed timeout mechanism into a LS algorithm with aperiodic state–change broadcasts. The rest of the paper is organized as follows. Section 2 outlines the LS algorithm and the proposed timeout mechanism. Section 3 and 4 establishes a theoretical basis for the calculation of optimal $T_{out}^{<i>}$. The HT formulation is treated in Section 3, while the probability distribution needed in the HT formulation is derived in Section 4. Section 5 presents and discusses representative numerical examples, and the paper concludes with Section 6.

2 The Proposed Algorithm

We proposed in [5, 6] a decentralized, dynamic LS algorithm for distributed real–time systems without considering node failures. We first state the assumptions made about the system under consideration, and summarize this algorithm for completeness. We then incorporate the proposed timeout mechanism in it to tolerate node failures.

We assume that the node clocks in the system are synchronized to establish a global time–base. A scheme for achieving this synchronization is presented in [7]. We also assume that the underlying communication subsystem supports reliable broadcasts [8] so that a non–faulty node can correctly broadcast its state change to all other non–faulty nodes in the system. Moreover, each node is assumed to have a constant exponential failure rate λ_F.

To facilitate algorithm description and analysis, we introduce the following notation and assumptions:

- λ_i: the composite (external and transferred–in) task arrival rate at node i. We *approximate* the composite task arrival process to be Poisson, and the validity of this approximation was discussed in [6].
- $\{p_i(j), 1 \le j \le E_{max}\}$: the distribution of execution times of both external and transferred–in tasks at node i, where E_{max} is the maximum task execution time measured in number of clock ticks.
- $\{\hat{p}_i(j), 1 \le j \le L_{max}\}$: the distribution of laxities of both external and transferred–in tasks at node i measured in clock ticks, where L_{max} is the maximum laxity. Both $p_i(j)$ and $\hat{p}_i(j)$ will be estimated on–line by each node i.
- CET_i: the CET on node i.
- $\underline{T_Q} = (T_1; T_2; ...; T_{L_{max}})$: the description of the sorted task queue on a node, where $T_j \triangleq e_1^j e_2^j ... e_{j+1}^j$ is a record of tasks with laxity $j \in \{1, \ldots, L_{max}\}$ currently queued on a node, and $e_k^j \in \{0, \ldots, E_{max}\}$, $1 \le k \le j+1$, is the execution time required by the k–th task with laxity j in the queue.
- O_i: the observation of CET_i made by some node $j \neq i$.
- $p_C(\cdot \mid O_i)$: the posterior distribution of CET_i given the observation O_i.
- TH_k, $1 \le k \le K_t - 1$: the state (CET) thresholds for broadcasting region–change messages, where K_t is the total number of state regions.

[2] both external and transferred-in

$T_{out}^{<i>}$: the timeout period; node i will be diagnosed as failed if no broadcast message from node i has been received for this period since the receipt of its last broadcast.

2.1 LS with Region–Change Broadcasts

Upon arrival of a task with laxity ℓ at node n, the node checks whether or not it can complete the task in time under the MLFS scheduling discipline. If it can, the task is queued at node n. If the task cannot be completed in time locally by node n or some of existing guarantees are to be violated as a result of inserting the newly-arrived task into the node's schedule, node n looks up the list of best LS decisions, and chooses — based on the current observation about other nodes' states, $\underline{O}$, and the laxity of the task(s) to be transferred — the best candidate receiver(s) in a small set of nodes in its physical proximity called a *buddy set*. (If multiple tasks have to be transferred, the observation about other nodes will be updated each time a LS decision is made.) The observation, $\underline{O}$, about other nodes is made via region-change broadcasts with time-stamped messages. The list of best LS decisions is updated periodically based on the state samples gathered via region–change broadcasts and Bayesian decision analysis, each of which is sketched below.

Buddy Sets: Each node communicates with, maintains the state information of, and transfers overflow tasks to, the nodes in its buddy set *only*. The communication overheads resulting from broadcasts/task transfers are thus reduced. On the other hand, the buddy sets in the system overlap with one another, and thus, the overflow tasks within a buddy set are shared by all capable nodes in the entire system, instead of overloading a few nodes within one buddy set [4].

Region–Change Broadcasts: The K_t state regions defined by $K_t - 1$ thresholds, TH_1, TH_2,..., TH_{K_t-1}, are used to characterize the workload of each node. Each node i broadcasts a time–stamped message, informing all the other nodes in its buddy set of its state–region change and all its on–line estimated parameters, whenever its CET crosses TH_{2k} for some k, where $1 \leq k \leq \lceil \frac{K_t}{2} \rceil - 1$. The state information kept at each node is thus up–to–date as long as the broadcast delay is not significant.

Bayesian Analysis: A node's observation O_i may be different from CET_i at the time of making a LS decision due to the delay in collecting it. In [5] we countered this problem by using Bayesian decision analysis. Each broadcast message from node i is time–stamped and contains the information of (1) the node number i, (2) CET_i, and (3) the time t_0 when this message is sent. When the message broadcast by node i arrives at node n, node i's CET_i at t_0 can be recovered by node n. Node n can also trace back to find its observation O_i about node i at time t_0. This observation O_i is what node n thought (observed) about node i when node i actually has CET_i. O_i's along with CET_i's are used by node n to compute/update periodically the posterior distribution, $p_C(\cdot \mid O_i)$, of CET_i given the observation O_i. (See [5] for a detailed account of this operation.) Any inconsistency between CET_i and O_i is characterized by this probability distribution. Besides, CET_i sent by node i at time t_0 is transformed into node n's new observation, O_i, about node i at the time node n receives this message, according to the rule that $O_i = k$ if $TH_k \leq CET_i < TH_{k+1}$, $0 \leq k < K_t - 1$, where $TH_0 \triangleq 0$ and $TH_{K_t} \triangleq \infty$.

To make a LS decision, node n — instead of hastily believing in its observation O_i about node i — estimates CET_i based on its (perhaps outdated) observation and determines node i's LS capability using $p_C(. \mid O_i)$, i.e., node n chooses the node i with the largest value of

$$P(CET_i \leq \ell) = \sum_{k=0}^{\ell} p_C(k \mid O_i).$$

2.2 Incorporation of the Timeout Mechanism into LS

As mentioned in Section 1, there are two possible scenarios, **S1** and **S2**, that node i may not broadcast any state–region change for a long time. The occurrence of **S1** is determined by the failure rate of node i, while **S2** is determined by the task arrival, completion, or transfer activities on node i, all of which dynamically change with the composite task arrival rate, the attributes of tasks arriving at node i, and node i's initial state. The timeout mechanism thus needs to dynamically adjusts $T_{out}^{<i>}$ based on the attributes of the tasks arriving at node i and the state of node i at the time of its last broadcast.

The timeout mechanism to be incorporated into LS is composed of the following sub-mechanisms.

On–line Parameter Estimation: Node i records on–line the interarrival time, the required execution time, and the laxity of each task upon its arrival, and applies the Bayesian technique to estimate the task parameters: λ_i, $\{p_i(j), 1 \leq j \leq E_{max}\}$, and $\{\hat{p}_i(j), 1 \leq j \leq L_{max}\}$. Application of the Bayesian technique to estimate these parameters can be found in [6].

Determination of Timeout Periods and Detection of Node Failures: Upon receiving a message broadcast by node i, node n uses the task parameters and $\underline{T_Q}$ contained in the message to calculate $T_{out}^{<i>}$. A theoretical basis for determining $T_{out}^{<i>}$ will be established in Sections 3 and 4. Conceptually, the problem of determining $T_{out}^{<i>}$ is first formulated as a HT problem by making a tradeoff between **S1** and **S2**. Then, the key expression needed in the HT formulation, i.e., the probability distribution that no message has been received from node i within time t given that node i is operational is derived by the randomization technique.

Node n considers node i failed if it has not heard from node i (via region–change broadcasts) for $T_{out}^{<i>}$ since node i's latest broadcast, and will not transfer any overflow tasks to node i until it receives a broadcast message from node i again. Whenever a failed node i is recovered, it broadcasts its recovery to all the other nodes in its buddy set. Upon receiving such a broadcast message, node n will consider node i capable of receiving tasks if the subsequent region–change broadcasts indicate so. On the other hand, node n also calculates its own timeout period $T_{out}^{<n>}$ at the time of broadcasting a state-region change. If node n has remained within

a broadcast-threshold interval for $T_{out}^{<n>}$, it broadcasts an extra message to inform other nodes of its fault-free (or 'I am alive') status.

3 Determination of the Optimal Timeout Period

In this section and the next section, we will establish a theoretical basis for the determination of $T_{out}^{<i>}$. To do this, we need:

1. On-line estimation of λ_i, $\{\hat{p}_i(j), 1 \le j \le L_{max}\}$, and $\{p_i(j), 1 \le j \le E_{max}\}$.
2. Node i's sorted task queue, $\underline{T_Q}$, which is contained in the most-recently-received broadcast message.

In Section 2, we discussed how the on-line estimated parameters are broadcast. λ_i, $\hat{p}_i(j)$'s, and $p_i(j)$'s are estimated by applying the Bayesian estimation technique [6].

The determination of $T_{out}^{<i>}$ requires to make a tradeoff between **S1** and **S2**, and can thus be formulated as a HT problem with two hypotheses. Specifically, let $Ob(t) \in \{0,1\}$ indicate whether or not a broadcast message from node i is received within time t, and let T_{nb} be the random variable representing the time to node i's next broadcast. We have two hypotheses:

H_0: **node i is operational** $\quad Ob(t) \sim p_0$,
H_1: **node i is faulty** $\quad Ob(t) \sim p_1$,

where p_0 and p_1 are the p.d.f. of $Ob(t)$ under the hypothesis of H_0 and H_1, respectively, and can be expressed as

$$\begin{aligned} p_0(Ob(t)=0) &= P(\text{no message received from node } i \text{ within } t \mid \text{node } i \text{ is non-faulty}) \\ &= P(T_{nb} \ge t \mid \text{node } i \text{ is non-faulty}), \\ p_0(Ob(t)=1) &= 1 - p_0(Ob(t)=0), \\ p_1(Ob(t)=0) &= 1, \text{ and } p_1(Ob(t)=1) = 0. \end{aligned}$$

Also, the probability that H_0 or H_1 is true without conditioning on any observation can be expressed as $\pi_0 = e^{-\lambda_F t}$ or $\pi_1 = 1 - e^{-\lambda_F t}$, respectively.

Now, a decision $\delta(Ob(t)) \in \{0,1\}$ must be made on which hypothesis must be accepted based on the observation $Ob(t)$. Two types of error may be encountered: (1) *false-alarm*, or H_0 is falsely rejected, the probability of which is denoted by $P_F(\delta)$; (2) *miss*, or H_1 is falsely denied, the probability of which is denoted by $P_M(\delta)$. The corresponding *detection* probability is $P_D(\delta) = 1 - P_M(\delta)$. A criterion for designing a test for H_0 versus H_1, called the Neyman-Pearson criterion, is to place a bound on the false-alarm probability and then to minimize the miss probability subject to this constraint; that is, the Neyman-Pearson design criterion is

$$\max_\delta P_D(\delta) \quad \text{subject to } P_F(\delta) \le \alpha_{ht}, \tag{3.1}$$

where α_{ht} is the significance level of the hypothesis test. Specifically, let the decision $\delta(\cdot)$ be

$$\delta(Ob(t)) = \begin{cases} 1, & \text{if } \pi_1 \cdot p_1(Ob(t)) \ge \pi_0 \cdot p_0(Ob(t)), \\ 0, & \text{otherwise,} \end{cases} \tag{3.2}$$

where the *maximum a posteriori* (MAP) probability is used to determine whether to accept H_1 or not. Then, $P_F(\delta)$ can be expressed as

$$\begin{aligned} P_F(\delta) &= P(\text{accept } H_1 \mid H_0 \text{ is true}) = E_0(\,\delta(Ob(t))\,) \\ &= P_0(\,\pi_1 \cdot p_1(Ob(t)) \ge \pi_0 \cdot p_0(Ob(t))\,) \\ &= \sum_{Ob(t)\in\{0,1\}} P(\,\pi_0 p_0(Ob(t)) \le \pi_1 p_1(Ob(t))\,) \cdot p_0(Ob(t)) \\ &= P(p_0(Ob(t)=0) \le \frac{\pi_1}{\pi_0}) \cdot p_0(Ob(t)=0), \end{aligned} \tag{3.3}$$

where $E_0(\cdot)$ and $P_0(\cdot)$ denote the expectation and the probability under H_0, and the last equality comes from $P(\pi_0 \cdot p_0(1) \le \pi_1 \cdot p_1(1)) = 0$. Similarly, $P_D(\delta)$ can be expressed as

$$\begin{aligned} P_D(\delta) &= E_1(\,\delta(Ob(t))\,) \\ &= P(p_0(Ob(t)=0) \le \frac{\pi_1}{\pi_0}). \end{aligned} \tag{3.4}$$

If the expression of $p_0(Ob(t) = 0) = P(T_{nb} \ge t \mid$ node i is operational) can be derived as a function of t, then the best $T_{out}^{<i>}$ under the Neyman-Pearson criterion is the minimum t such that both

$$p_0(Ob(t)=0) \le \alpha_{ht} \quad \text{and} \quad p_0(Ob(t)=0) \le \frac{\pi_1}{\pi_0} = e^{\lambda_F t} - 1 \tag{3.5}$$

are satisfied, in which case $P_D(\delta) = 1$ and $P_F(\delta) \le \min\{\alpha_{ht}, e^{\lambda_F t} - 1\}$.

4 Derivation of $P(T_{nb} \ge t \mid$ node i is operational)

We now use the randomization technique [9] to calculate $P(T_{nb} \ge t \mid$ node i is operational). Since this technique can be applied only to a finite state-space continuous-time Markov chain, we model the state evolution of a node as such. We first describe how the system model is constructed. Then, we derive $P(T_{nb} \ge t \mid$ node i is operational).

4.1 System Model

The state/CET evolution of a node is modeled as a continuous-time Markov chain $\{X(t), t \ge 0\}$ on a finite state space S. Transitions in the Markov chain are characterized by the generator matrix $Q = (q_{ij})$, where q_{ij}, $0 \le i, j \le N$, is the transition rate from state i to state j. The parameters needed in the model, λ_i, $p_i(j)$, and $\hat{p}_i(k)$, are estimated on-line by each node i and piggybacked in region-change broadcasts to the other nodes.

We characterize the CET evolution caused by task acceptance/completion under the non-preemptive MLFS discipline. With a minor modification, our model can also be applied to the case when the loading state is queue length. To construct a continuous-time Markov chain on a finite state space, we approximate the deterministic consumption of CET on node i (at a pace of 1 per unit time) as an Erlang distribution with rate K and shape parameter K. The Erlang distribution becomes exact (i.e., deterministic with rate 1) as $K \to \infty$. We choose K such that $P(T_{nb} \ge t \mid$ node i is operational) obtained from the corresponding $M^{[n]}/E_K/1$ model is very close to that obtained from $M^{[n]}/E_{K+1}/1$ model. In Section 5.1, $K \ge 5$ is shown to satisfy the above criterion for all combinations of task

attributes studied. Each accepted/queued task contributes Km service stages with probability $p_i(m)$, $1 \le m \le E_{max}$, and each service stage is consumed at (an exponential) rate K.

Definition of state: The state of node i is defined as $\underline{H} = (H_0; H_1; H_2; ...; H_{L_{max}})$, where $H_j \triangleq h_1^j h_2^j ... h_{j+1}^j$ is a sequence of $j+1$ numbers with $h_k^j \in \{0, \ldots, KE_{max}\}$ representing the number of service stages contributed by the k-th laxity-j task in the node's queue. Since all laxity-j tasks queued on node i must start execution by their laxity, there are at most $j+1$ laxity-j tasks that can be queued on node i (in which case all but, perhaps, the last task require 1 unit of execution time). Moreover, let $c_j \triangleq \sum_{k=1}^{j+1} h_k^j$ denote the total number of service stages contributed by all laxity-j tasks, $last(H_j)$ denote the index of the last nonzero entry in H_j, and

$$L\text{now}(\underline{H}) \triangleq \begin{cases} -1, \text{ if } \underline{H} = \underline{0}; \\ \text{minimum } \ell \text{ s.t. } \prod_{j=0}^{\ell-1}(1 - \Delta^{>}(c_j)) \times \Delta^{>}(c_j) = 1, \\ \quad \text{if } \underline{H} \neq \underline{0}, \text{ and } h_1^j \in \{0\} \cup \{Km : 1 \le m \le E_{max}\}\ \forall j; \\ \text{the only index } \ell \text{ s.t. } h_1^\ell \notin \{0\} \cup \{Km : 1 \le m \le E_{max}\}, \\ \quad \text{if } \underline{H} \neq \underline{0}, \text{ and } \exists j \text{ s.t. } h_1^j \notin \{0\} \cup \{Km : 1 \le m \le E_{max}\} \end{cases}$$

denote the laxity of the task currently under service, where

$$\Delta^{>(\ge)}(x) \triangleq \begin{cases} 1, & \text{if } x > (\ge) 0, \\ 0, & \text{otherwise.} \end{cases}$$

For example, consider a system model with $L_{max} = 3$, $E_{max} = 2$, and $K = 4$. $L\text{now}(\ (0;40;000;1000)\ ) = 3$ indicates that the task currently being served has 3 time units of laxity and 1 remaining service stage. $L\text{now}(\ (0;00;440;8000)\ ) = 2$ indicates that the task to be served next is the one with 2 time units of laxity and 4 service stages if there are no new laxity-1 task arrivals before the next state transition.

Under the non-preemptive MLFS discipline, the state $\underline{H}$ has the following properties:

P1: $h_k^j \in \mathcal{N}$ is an integer multiple of K except for perhaps h_1^j, the number of service stages contributed by the laxity-j task currently under service.

P2: The size of the state space is bounded by $\prod_{i=0}^{L_{max}} (KE_{max}+1)(E_{max}+1)^i$ and thus is finite.

P3: Since a task with laxity j is accepted/queued only if the CET contributed by both the tasks with laxity $\le j-1$ and the task currently under service is no greater than j units of time, we have $c_j > 0$ only if

$$Kj \ge \sum_{n=0}^{j-1} c_n, \ \forall j \in [L\text{now}(\underline{H}) + 1, L_{max}],$$

or, $Kj \ge \sum_{n=0}^{j-1} c_n + h_1^{L\text{now}(\underline{H})}, \ \forall j \in [0, L\text{now}(\underline{H}) - 1]$. Note that $c_{L\text{now}(\underline{H})} > 0$ by the definition of $L\text{now}(\underline{H})$ (except for the case of $\underline{H} = \underline{0}$).

P4: Since every laxity-j task queued on node i must be able to start execution by its laxity, the number of service stages queued 'in front of' it must be $\le Kj$, i.e.,

$$\sum_{n=0}^{j-1} c_n + \sum_{n=1}^{last(H_j)-1} h_n^j \le Kj, \ \forall j \in [L\text{now}(\underline{H}) + 1, L_{max}],$$

or,

$$\sum_{n=0}^{j-1} c_n + h_1^{L\text{now}(\underline{H})} + \sum_{n=1}^{last(H_j)-1} h_n^j \le Kj, \ \forall j \in [0, L\text{now}(\underline{H}) - 1].$$

For example, consider again the system model with $L_{max} = 3$, $E_{max} = 2$, and $K = 4$. The state (0;10;440;8000) is allowed, while (0;10;480;4000) is not, because the task with 3 time units of laxity and 4 service stages (represented by the underlined number 4) in the latter state violates **P3** and **P4**.

As indicated in **P2**, the size of the state space is bounded and is actually much less than the given bound because of **P1** and **P3**–**P4**. It, however, grows significantly as L_{max} or E_{max} or K increases, but as will be clearer later in this section, the generator matrix Q of the corresponding Markov chain is very sparse, so one can exploit the sparseness of Q — e.g., use the modified SERT algorithm proposed in [9] — to economically store sparse matrices, and to alleviate the computational difficulty.

Determination of transition rates: There are two task activities that cause state transitions: one is task acceptance by node i, and the other is CET consumption by node i. The task transfers resulted from the acceptance of a newly-arrived task under the MLFS scheduling discipline are figured in task acceptance.

A. The transition caused by task acceptance: Assume that the system is in state $\underline{H}$, and will make a transition to state $\underline{H}'_{\ell,Km} \triangleq (H'_0; H'_1; ...; H'_\ell; ...; H'_{L_{max}})$ upon acceptance of a task with laxity ℓ and execution time m, where $1 \le \ell \le L_{max}$ and $1 \le m \le E_{max}$. Then

(1) $c'_j = c_j$ (or equivalently, $H'_j = H_j$), $1 \le j \le \ell - 1$, i.e., the CET contributed by tasks with laxity $\le \ell - 1$ will not be affected by the acceptance of a task with laxity ℓ;

(2) H'_j equals H_j, perhaps with the last few entries ($\ell+1 \le j \le L_{max}$) replaced by 0 (so $c'_j \le c_j$). That is, the tasks originally queued with laxity $> \ell$ may have to be transferred out because of the insertion of a newly-arrived task.

(3) The number of nonzero entries in H_ℓ is not greater than ℓ, and $H'_\ell = h_1^\ell ... h_{last(H_\ell)}^\ell\ Km\ 0...0$, i.e., H'_ℓ consists of the nonzero entries in H_ℓ followed by the number Km (and possibly a few 0's to make the number of entries equal to $\ell + 1$).

(4) The corresponding transition rate (under the non-preemptive policy) is

$$q_{\underline{H},\underline{H}'_{\ell,Km}} = \lambda_i \hat{p}_i(\ell) p_i(m) \cdot \text{Check\_Cet}(\ell) \cdot \prod_{\substack{t=\ell+1 \\ t \notin zero(\underline{C})}}^{L_{max}} \Big\{ comp(H_t, H'_t) \cdot \text{Task\_Not\_Transfer}(t) + (1 - comp(H_t, H'_t)) \cdot \text{Task\_Transfer}(t) \Big\}, \tag{4.1}$$

where

$$\text{Check\_Cet}(\ell) \triangleq \begin{cases} \Delta^{\ge}(K\ell - \sum_{j=0}^{\ell} c_j), \\ \quad \text{if } L\text{now}(\underline{H}) \le \ell, \\ \Delta^{\ge}(K\ell - (\sum_{j=0}^{\ell} c_j + h_1^{L\text{now}(\underline{H})})), \\ \quad \text{if } L\text{now}(\underline{H}) > \ell; \end{cases}$$

$zero(\underline{C}) \triangleq$ the set of indices j such that $c_j = 0$;

$$comp(H_t, H'_t) \triangleq \begin{cases} 1, & \text{if } H_t = H'_t, \\ 0, & \text{otherwise;} \end{cases}$$

Task_Not_Transfer(t) $\triangleq$

$$\begin{cases} 1, & \text{if } t = L_{\text{now}}(\underline{H}) \text{ and } last(H_t) = 1, \\ \Delta\geq(Kt - (\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-1} h^t_j)), & \\ \quad \text{if } t = L_{\text{now}}(\underline{H}) \text{ and } last(H_t) > 1, & \\ \Delta\geq(Kt - (\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-1} h^t_j + h_1^{L_{\text{now}}(\underline{H})})), & \\ \quad \text{if } t < L_{\text{now}}(\underline{H}), & \\ \Delta\geq(Kt - (\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-1} h^t_j)), & \\ \quad \text{if } t > L_{\text{now}}(\underline{H}); & \end{cases}$$

Task_Transfer(t) $\triangleq$

$$\begin{cases} \Delta\geq(Kt - (\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H'_t)-1} h^t_j)) \times & \\ \quad \Delta>((\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H'_t)} h^t_j) - Kt), & \\ \quad \text{if } t = L_{\text{now}}(\underline{H}) \text{ and } last(H'_t) \geq 2, & \\ \Delta>(\sum_{j=0}^{t-1} c'_j + h^t_1 - Kt), & \\ \quad \text{if } t = L_{\text{now}}(\underline{H}) \text{ and } last(H'_t) = 1, & \\ 0, \quad \text{if } t = L_{\text{now}}(\underline{H}) \text{ and } last(H'_t) = 0, & \\ \Delta\geq(Kt - (\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H'_t)-1} h^t_j)) \times & \\ \quad \Delta>((\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H'_t)} h^t_j) - Kt), & \\ \quad \text{if } t > L_{\text{now}}(\underline{H}) \text{ and } last(H'_t) \neq 0, & \\ \Delta>(\sum_{j=0}^{t-1} c'_j - Kt), & \\ \quad \text{if } t > L_{\text{now}}(\underline{H}) \text{ and } last(H'_t) = 0, & \\ \Delta\geq(Kt - (\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H'_t)-1} h^t_j + h_1^{L_{\text{now}}(\underline{H})})) \times & \\ \quad \Delta>((\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H'_t)} h^t_j) + h_1^{L_{\text{now}}(\underline{H})} - Kt), & \\ \quad \text{if } t < L_{\text{now}}(\underline{H}) \text{ and } last(H'_t) \neq 0, & \\ \Delta>(\sum_{j=0}^{t-1} c'_j + h_1^{L_{\text{now}}(\underline{H})} - Kt), & \\ \quad \text{if } t < L_{\text{now}}(\underline{H}) \text{ and } last(H'_t) = 0; & \end{cases}$$

The physical meanings of Eq. (4.1) are given below:

(a) The first factor $\lambda_i \hat{p}_i(\ell) p_i(m)$ is the arrival rate of tasks with ℓ time units of laxity and m units of execution time on node i.

(b) The second factor Check_Cet(ℓ) accounts for the fact that a newly-arrived task with laxity ℓ will be queued/accepted on node i only if one of the following two conditions holds: (i) the CET contributed by tasks with laxity $\leq \ell$ is less than or equal to ℓ, if the laxity of the currently executing task $\leq \ell$; or (ii) the CET contributed by the tasks with laxity $\leq \ell$ and the task currently under service is $\leq \ell$ if the laxity of the currently executing task $> \ell$.

(c) The last factor accounts for the possible task transfers caused by the acceptance of the arrived task. Since only tasks with laxity $> \ell$ will be affected by the insertion of the newly-arrived task with laxity ℓ, $\prod$ is performed from $t = \ell+1$ to $t = L_{max}$ except for those t's with $c_t = 0$. The transition could occur with rate $\lambda_i \hat{p}_i(\ell) p_i(m)$ if, in addition to the conditions in Check_Cet(ℓ), one of the following conditions holds, $\forall t \in [\ell + 1, L_{max}]$:

(i) $H_t = H'_t$ and all tasks queued with laxity t can still be completed in time after the insertion of the arrived task.

(ii) H'_t equals H_t except with the last $(last(H_t) - last(H'_t))$ entries replaced by zero, i.e., a number $(last(H_t) - last(H'_t))$ of tasks with laxity t must be transferred out. For example, if $t > L_{\text{now}}(\underline{H})$, exactly i tasks with laxity t have to be transferred out iff both $\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-i-1} h^t_j \leq Kt$ and $\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-i} h^t_j > Kt$ hold.

B. The transitions caused by CET consumption: The deterministic consumption of CET at a pace of 1 per unit time is approximated as a K–Erlang distribution with rate K. Besides, at the end of each time unit (i.e., at the end of every K service stages), all laxities have to be decremented by 1 to account for the fact that the laxity of a task is measured w.r.t. the current time. Specifically, the system makes a transition from $\underline{H}$ to $\underline{H}'_{\ell,-1} = (H'_0; H'_1; ...; H'_\ell; ...; H'_{L_{max}})$, with transition rate

$$q_{\underline{H},\underline{H}'_{\ell,-1}} = \begin{cases} K, & \text{if } \ell = L_{\text{now}}(\underline{H}), \\ 0, & \text{otherwise,} \end{cases} \tag{4.2}$$

where

(1) if $(h^\ell_1 - 1) \notin \{Km : 1 \leq m \leq E_{max}\} \cup \{0\}$,

$$H'_\ell = (h^\ell_1 - 1)h^\ell_2 ... h^\ell_{\ell+1}, \text{ and } H'_j = H_j \ \forall j \neq \ell.$$

(2) if $(h^\ell_1 - 1) \in \{Km : 1 \leq m \leq E_{max}\} \cup \{0\}$,

$$\begin{aligned} H'_{L_{max}} &= \underline{0}, \\ H'_j = H_{j+1} &= h^{j+1}_1 ... h^{j+1}_{j+1}, \ \forall j \in [0, \ell - 2] \cup [\ell, L_{max} - 1], \\ H'_{\ell-1} &= \begin{cases} (h^\ell_1 - 1)h^\ell_2 ... h^\ell_\ell, & \text{if } h^\ell_1 > 1, \\ h^\ell_2 ... h^\ell_\ell 0, & \text{if } h^\ell_1 = 1. \end{cases} \end{aligned}$$

The last nonzero transition rate is

$$q_{\underline{H},\underline{H}} = -(\sum_{\ell,m} q_{\underline{H},\underline{H}'_{\ell,Km}} + \sum_{\ell} q_{\underline{H},\underline{H}'_{\ell,-1}}) \triangleq -q_{\underline{H}}. \tag{4.3}$$

The model constructed above is a continuous-time Markov chain, because (1) the residence time at each state is exponentially distributed, and (2) the next state the system will visit depends only on the current state and the task acceptance/completion activities occurred during the residence at the current state. The sparseness of Q comes from the fact that all the other entries (except for the transition rates in Eq. (4.1)–(4.3)) in Q are zero. For example, the only possible transitions from state (0;10;400;4800) in the system model with $L_{max} = 3$, $E_{max} = 2$, and $K = 4$ are to (0;40;480;0000), (0;14;400;4000), (0;18;000;4000), (0;10;440;4000), (0;10;480;0000), and (0;10;400;4800) with transition rate K, $\lambda_i p_i(1)\hat{p}_i(1)$, $\lambda_i p_i(2)\hat{p}_i(1)$, $\lambda_i p_i(1)\hat{p}_i(2)$, $\lambda_i p_i(2)\hat{p}_i(2)$, and $-(K + \sum_{1 \leq \ell, m \leq 2} \lambda_i p_i(m)\hat{p}_i(\ell))$, respectively.

4.2 Probability Calculation with the Randomization Technique

We now use the randomization technique to calculate the probability that a node does not broadcast any message in $[0, t]$, given it is operational in $[0, t]$. Before delving into the derivation, we summarize below some important results of the randomization technique.

Consider a continuous-time Markov chain $\{X(t), t \geq 0\}$ with generator matrix Q on a finite state space S of size $N+1$. For notational convenience, we enumerate the elements of S as $0, 1, \ldots, N$. Let $\Lambda \triangleq \max_{0 \leq i \leq N} q_i$, then there exists a discrete-time Markov chain $\{Y_n, n = 0, 1, \ldots\}$ and a Poisson process $\{N(t), t \geq 0\}$ with rate Λ, which are independent of each other, such that the process $\{Y_{N(t)}, t \geq 0\}$ has the same finite dimensional distributions as, and is thus probabilistically identical to, $\{X(t), t \geq 0\}$. In the equivalent process, the transition rate from state i is Λ, but only the fraction q_i/Λ of transitions are real and the remaining fraction $1 - \frac{q_i}{\Lambda}$ are fictitious transitions. In other words, $\{X(t), t \geq 0\}$ can be considered as a process which spends a time with an exponential rate Λ in state i and then makes a transition to state j with probability $\mathcal{P}_{ij}$, where

$$\mathcal{P}_{ij} = \begin{cases} 1 - \frac{q_i}{\Lambda} & \text{if } j = i, \\ \frac{q_{ij}}{\Lambda} & \text{if } j \neq i. \end{cases} \tag{4.4}$$

The transient probabilities, $P(X(t) = i)$, $0 \leq i \leq N$, of the continuous-time Markov chain $\{X(t), t \geq 0\}$ can now be easily obtained by conditioning on $N(t)$, the number of transitions in $(0, t]$, i.e.,

$$\begin{aligned} P(X(t) = i) &= P(Y_{N(t)} = i) \\ &= \sum_{n=0}^{\infty} P(Y_{N(t)} = i \mid N(t) = n) \cdot P(N(t) = n) \\ &= \sum_{n=0}^{\infty} P(Y_n = i) e^{-\Lambda t} (\Lambda t)^n / n!. \end{aligned} \tag{4.5}$$

In other words, a continuous-time Markov chain $\{X(t), t \geq 0\}$ on a finite state space S, after its randomization, can be viewed as a discrete-time Markov chain, $\{Y_n, n = 0, 1, \ldots\}$, subordinated to a Poisson process $\{N(t), t \geq 0\}$, and thus the transient probabilities can be easily computed using the discrete-time Markov chain.

We are now in a position of deriving $P(T_{nb} \geq t \mid$ node i is operational). Recall that in the proposed LS algorithm, a node's states are divided into K_t disjoint subsets by thresholds $TH_k, 1 \leq k \leq K_t - 1$. A node will broadcast to other nodes its change of state region whenever its state/CET crosses even-numbered thresholds, TH_{2j}, $1 \leq j \leq \lceil \frac{K_t}{2} \rceil - 1$. We thus define $S_j = \{\underline{H} : K \cdot TH_{2(j-1)} \leq \sum_{n=1}^{L_{max}} (\sum_{k=1}^{n+1} h_k^n) < K \cdot TH_{2j}\}$ as the j-th broadcast state region, where $TH_0 \triangleq 0$, the expression $\sum_{k=1}^{n+1} h_k^n$ is the number of service stages contributed by laxity-n tasks (i.e., c_n), and the expression between inequalities $\sum_{n=1}^{L_{max}} (\sum_{k=1}^{n+1} h_k^n)$ is simply the total number of service stages queued on the node.

Let $r_j(n, k), 0 \leq k \leq n+1$, be the probability that the discrete-time Markov chain, Y, obtained after the randomization of $X(t)$ visits k times the states in S_j out of n state changes. For example, $r_j(n, n+1)$ is the probability that Y always stays in S_j while there are n state changes. Then, $P(T_{nb} \geq t \mid$ node i is operational and was in S_j during the last broadcast), $1 \leq j \leq \lceil \frac{K_t}{2} \rceil$, is the probability that the underlying Markov chain always stays in S_j, no matter how many state changes have occurred in $[0, t]$. Thus,

$$\begin{aligned} &P(T_{nb} \geq t \mid \text{node } i \text{ was in } S_j \text{ in the last broadcast}) \\ &= \sum_{n=0}^{\infty} r_j(n, n+1) \cdot P(n \text{ state changes in time } t) \\ &= \sum_{n=0}^{\infty} r_j(n, n+1) \cdot e^{-\Lambda t} (\Lambda t)^n / n! \end{aligned} \tag{4.6}$$

where Λ is the rate of the Poisson process obtained after the randomization.

$r_j(n, k)$ (and $r_j(n, n+1)$, in particular) can be easily calculated using the recursive approach proposed in [10]. That is, let $r_j(n, k, \underline{H})$ be the probability that the underlying Markov chain Y are k times in S_j out of n state changes and the state visited in the last transition is state $\underline{H}$. $r_j(n, k, \underline{H})$ depends on

- $r_j(n-1, k-1, \hat{\underline{H}})$, $\forall \hat{\underline{H}} \in S$, if $\underline{H} \in S_j$, since we have to increment the number of states $\in S_j$ visited by one for the previous state change from $\hat{\underline{H}}$ to $\underline{H}$;
- $r_j(n-1, k, \hat{\underline{H}})$ $\forall \hat{\underline{H}} \in S$, if $\underline{H} \notin S_j$, since the number of states $\in S_j$ visited remains the same for the current state change from $\hat{\underline{H}}$ to $\underline{H}$.

So,

$$r_j(n, k, \underline{H}) = \begin{cases} \sum_{\hat{\underline{H}} \in S} r_j(n-1, k-1, \hat{\underline{H}}) \cdot \mathcal{P}_{\hat{\underline{H}}, \underline{H}}, & \text{if } \underline{H} \in S_j, \\ \sum_{\hat{\underline{H}} \in S} r_j(n-1, k, \hat{\underline{H}}) \cdot \mathcal{P}_{\hat{\underline{H}}, \underline{H}}, & \text{if } \underline{H} \notin S_j, \end{cases} \tag{4.7}$$

where $\mathcal{P}$ is the transition matrix of Y, and the initial conditions are

$$\begin{aligned} r_j(0, 1, \underline{H}) &= \begin{cases} 1, & \text{if } \underline{H} \in S_j, \\ 0, & \text{otherwise,} \end{cases} \\ r_j(0, 0, \underline{H}) &= 0, \end{aligned} \tag{4.8}$$

where Eq. (4.8) comes from the fact that given the CET was in S_j during the last broadcast, the node must be initially in a state $\in S_j$, and the k within the expression of $r_j(n, k, \underline{H})$ must be ≥ 1. Finally, $r_j(n, k) = \sum_{\underline{H} \in S} r_j(n, k, \underline{H})$.

Since we are interested in obtaining $r_j(n, n+1)$, we need only to compute $r_j(n, n+1, \underline{H}), \forall \underline{H} \in S_j$, as $r_j(n, n+1, \underline{H}) = 0$, $\forall \underline{H} \notin S_j$. Thus, Eq. (4.7) reduces to

$$r_j(n, n+1, \underline{H}) = \sum_{\hat{\underline{H}} \in S_j} r_j(n-1, n, \hat{\underline{H}}) \cdot \mathcal{P}_{\hat{\underline{H}}, \underline{H}} \quad \forall \underline{H} \in S_j.$$

5 Numerical Examples

The proposed timeout mechanism is evaluated in the following sequence:

1. Discussion on the parameters considered/varied in performance evaluation.
2. Discussion on $T_{out}^{<i>}$ (a) w.r.t. task attributes, and (b) w.r.t. the state in which a node was during its latest broadcast.
3. Performance evaluation: We comparatively evaluate (a) LS with no timeout mechanism, (b) LS with fixed timeouts, (c) LS with the calculated best timeouts, and (d) LS with immediate detection of each node failure upon its occurrence.

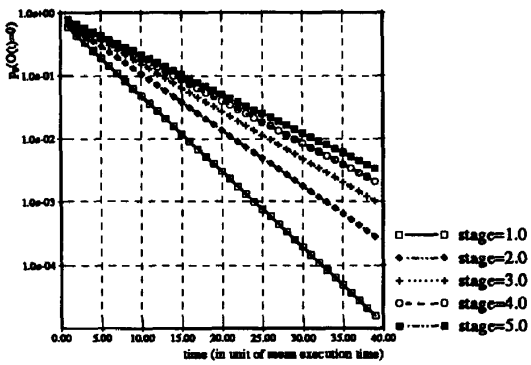

Figure 1: $P(T_{nb} \geq t \mid$ node i is operational) derived w. r. t. shape parameter K.

5.1 Parameters Considered/Varied

The performance of LS algorithms depends on a large number of parameters which are classified into the following four groups:

(1). Parameters of the distributed system, such as the number of nodes in the system N_n, the size of buddy set, node failure rate λ_F, node recovery rate μ_F, and the communication delay which consists of task-transmission delay and medium-queueing delay.

(2). Parameters of the (node-level) system model, such as the shape parameter K used to approximate the CET consumption as a K-Erlang distribution.

(3). Characteristic parameters of the task set, such as the external task arrival rate λ_i^{ext}, the laxity distribution of external tasks, and the distribution of execution time required by external tasks, on each node i. For all results presented below, we use $\{e_1, ..., e_k\}_{\{p_{e_1}, ..., p_{e_k}\}}$ to denote the task set in which an external task requires execution time e_i with probability p_{e_i}, $\forall i$. If $p_{e_i} = p \ \forall e_i$, then $\{p_{e_1}, p_{e_2}, ..., p_{e_k}\}$ is condensed to p. Similarly, $\{\ell_1, \ell_2, ..., \ell_n\}_{\{p_{\ell_1}, p_{\ell_2}, ..., p_{\ell_n}\}}$ is used to describe the laxity distribution of external tasks.

(4). Design parameters of the proposed LS algorithm, such as the number, K_t, and values of thresholds, TH_1,..., TH_{K_t-1} used as reference points for broadcasts.

A 16-node regular[3] system is used in our simulations. The size of buddy set is chosen to be 12, because increasing it beyond 10 was shown in [4] to be ineffective. Both node failure and recovery rates are assumed to be exponential with λ_F varying from 10^{-2} to 10^{-4} and μ_F being fixed at 10^{-1}. Broadcast messages compete with task transfers for the communication medium. No priority mechanism regulates the transmission over the medium (i.e., a FCFS rule is assumed). The task-transmission delay is varied from 10% to 50% of the corresponding task execution time. The broadcast-message-transmission delay is assumed to be negligible. The queueing delay which is experienced by both broadcast messages and transferred tasks and which dynamically changes with system traffic is modeled as a linear function of the number of tasks/messages queued in the medium. The shape parameter K is chosen to be 5, since $P(T_{nb} \geq t \mid$ node is operational) thus derived is almost indistinguishable from that derived with $K \geq 6$ (Fig. 1).

The simulation was carried out for both exponential and hyperexponential task arrivals while varying the external task arrival rate per node, λ_i^{ext}, from 0.2 to 0.9, the ratio of $\frac{e_{j+1}}{e_j}$ $(1 \leq j \leq k-1)$ from 2 to 5, and the ratio of of $\frac{\ell_{j+1}}{\ell_j}$ $(1 \leq j \leq n-1)$ from 2 to 4. For convenience, all time-related parameters are expressed *in units of average task execution time.*

The design parameters, K_t and TH_k's, may affect the accuracy of the posterior CET distributions, $p_C(\cdot \mid O_i)$, given the observation O_i. We already discussed one method in [5] that determines the design parameters. For the performance study below, we tuned the design parameters using the method described in [5] for each combination of system configuration and task set.

We present only those results that we believe are the most relevant, interesting, and/or representative. In the results reported below, the parameters are specified as (unless otherwise stated): $\lambda_i = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. Node i has 4 state regions with each interval equal to 1 (except for the last interval), i.e., $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$. The state of node i is CET=1.0 in the last broadcast. $\alpha_{ht} = 0.05$. The task-transfer delay is set to 10% of the execution time of the transferred task. In spite of a large number of possible combinations of parameters, the conclusion drawn from the performance curves for a task set with the given task execution and laxity distributions and a given system configuration is valid over a wide range of combinations of execution time and laxity distributions.

5.2 Discussion of $T_{out}^{<i>}$

Figs. 2 and 3 illustrate how $p_0(O(t) = 0) = P(T_{nb} \geq t \mid$ node i is operational) (and consequently, $T_{out}^{<i>}$) varies markedly with the task arrival rate and the initial state of node i at the time of its latest broadcast, respectively. As the composite task arrival rate increases, a node tends to cross its state/CET boundaries more (less) often if the closest boundary is to the right (left) of the node's current state. Thus, in Fig. 2, the increase in λ_i yields a smaller $p_0(O(t) = 0)$ for a given t. Similarly, as evidenced in Fig. 3, the closer the initial state of a node is to a broadcast threshold, the more likely the node's state will cross the boundary, thus increasing the possibility of a region-change broadcast.

Since $p_0(O(t))$ varies drastically with the task attributes and the initial state of a node, the on-line calculation of $T_{out}^{<i>}$ is very important to the design of a timeout mechanism. Table 1 give some numerical values of $T_{out}^{<i>}$ for different task attributes and initial states.

[3] A system is regular if the degrees of all nodes are identical.

5.3 Performance Evaluation

Performance Measures of Interest: Instead of using the mean task response time as a performance metric, we use two measures which are more relevant to fault-tolerant real-time performance:

- The probability of dynamic failure, P_{dyn}: the probability of tasks failing to complete before their deadlines.
- The probability of false alarm, P_F: the probability of falsely diagnosing a healthy node as failed. A larger P_F will leave a node with fewer candidate nodes for task transfers, thus deteriorating the LS performance.

Performance comparison among LS with different timeout periods: Using trace-driven simulations, we comparatively evaluate the performance improvement achievable with the on-line calculated best timeout mechanism. We compare the proposed LS algorithm with the best timeout period against the case of using a fixed timeout period, $T_{fixed}^{<i>}$, where $T_{fixed}^{<i>}$ is a constant selected independently of node i's task attributes and state. We also compare the proposed timeout mechanism with two baseline mechanisms. The first baseline assumes no timeout mechanism, while the second is an ideal case where (1) each node immediately detects the failure of another node upon its occurrence and (2) no false alarm occurs.

For each combination of task set and system configuration, the simulation ran until it reached a 95% level of confidence in the numerical results for a maximum error of 2% within the specified probability (P_{dyn} or P_F). The number of simulation runs needed to achieve the above confidence level is calculated by the Student-t test.

Fig. 4 plots P_F curves for LS with different timeout periods. Fig. 5 plots P_{dyn} curves for LS with different timeout periods w.r.t. λ_i^{ext}. From these figures, we make the following observations:

- In general, P_F decreases as (1) task arrivals/transfers get more frequent (i.e., as the system load increases), and (2) the timeout periods get larger. Thus, the case of $T_{fixed}^{<i>} = 20$ performs best w.r.t. P_F for medium to heavy system loads ($\lambda_i^{ext} \geq 0.6$) where $5 < T_{out}^{<i>} < 20$ (Table 1). For light to medium system loads ($0.2 \leq \lambda_i^{ext} \leq 0.6$), the case of $T_{out}^{<i>}$ performs best w.r.t. P_F, because each node in a lightly or medium loaded system usually stays in the broadcast region $S_1 = [0, 2]$ where $T_{out}^{<i>} > 20$ (Table 1).
- The assumed 5% chance of incorrect diagnosis ($\alpha_{ht} = 0.05$ in the HT formulation) is reduced with a few extra, timely messages broadcast by each node to inform other nodes of its fault-free status after a silence for $T_{out}^{<i>}$.
- The case with the on-line calculated $T_{out}^{<i>}$ outperforms all the other fixed-timeout cases tested in reducing P_{dyn} over a wide range of system load. The case with $T_{fixed}^{<i>} = 20$ is inferior to that with $T_{out}^{<i>}$ for medium to heavy system loads due to its inability of early detection of a node failure. The case with $T_{fixed}^{<i>} = 5$ is inferior to that with $T_{out}^{<i>}$ because of the undesirable effects of false diagnosis. Frequent 'I am alive' messages in case of $T_{out}^{<i>}$ also consume communication bandwidth and compete with those tasks being transferred for the use of communication medium. Thus, there is a definite performance advantage with on-line parameter estimation of task attributes and calculation of $T_{out}^{<i>}$. The performance with $T_{out}^{<i>}$ is, however, worse than the ideal case due to the fact that node n might send its overflow task to a failed node i during the period between the occurrence of node i's failure and its detection by node n.

6 Concluding Remarks

We have proposed a timeout mechanism which, when there are node failures, can be incorporated into LS with aperiodic state-change broadcasts. By (1) on-line collection/estimation of parameters relevant to task attributes, and (2) calculating — based on the observation and the estimated task attributes in the latest broadcast — the best timeout period used to diagnose a silent node as failed, the probability of dynamic failure can be significantly reduced, as compared to LS without any timeout mechanism or with a fixed timeout mechanism.

References

[1] J. Hong, X. Tan, and D. Towsley, "A performance analysis of minimum laxity and earliest deadline scheduling in a real-time system," *IEEE Trans. on Computers*, vol. C-38, no. 12, pp. 1736–1744, December 1989.

[2] 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. 1110–1123, August 1989.

[3] 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.

[4] 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.

[5] K. G. Shin and C.-J. Hou, "Design and evaluation of effective load sharing in distributed real-time systems," *Proc. of Third IEEE Symposium on Parallel and Distributed Processing*, pp. 670–677, December 1991.

[6] C.-J. Hou and K. G. Shin, "Load sharing with consideration of future task arrivals in heterogeneous distributed real-time systems," *IEEE Proc. of 12th Real-Time System Symposium*, pp. 94–103, December 1991.

[7] P. Ramanathan, D. D. Kandlur, and K. G. Shin, "Hardware assisted software clock synchronization for homogeneous distributed systems," *IEEE Trans. on Computers*, vol. C-39, no. 4, pp. 514–524, 1990.

[8] P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Trans. on Computers*, vol. C-37, no. 12, pp. 1654–1657, 1988.

[9] B. Melamed and M. Yadin, "Randomization procedures in the computation of cumulative-time distributions over discrete state markov processes," *Operations Research*, vol. 32, no. 4, pp. 926–944, July–August 1984.

[10] E. de Souza e Silva and H. R. Gail, "Calculating cumulative operational time distributions of repairable computer systems," *IEEE Trans. on Computers*, vol. C-35, no. 4, pp. 322–332, April 1986.

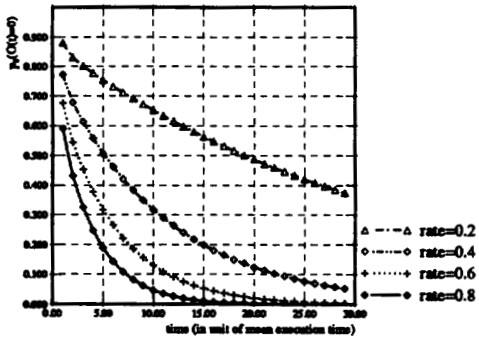

Figure 2: $P(T_{nb} \geq t \mid$ node i is operational) w. r. t. task arrival rate λ_i.

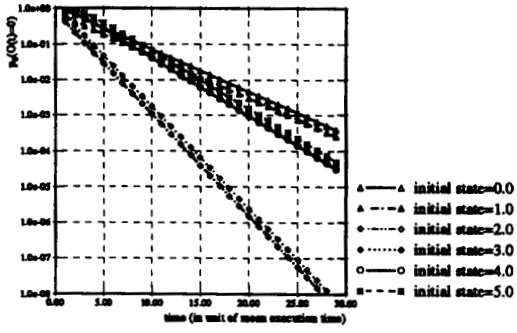

Figure 3: $P(T_{nb} \geq t \mid$ node i is operational) w. r. t. the initial state node i is in.

λ	L	Initial state	Best T_{out}
0.4	$\{1\}_1$	0	32.6
		1.0	29.8
		2.0	3.4*
		$\geq$ 3.0	$\geq$ 4.8*
	$\{1,2,3\}_{1/3}$	0	32.6
		1.0	29.8
		2.0	4.6*
		3.0	6.4
		4.0	7.8
		$\geq$ 5.0	$\geq$ 9.1
	$\{1,2,3,4,5\}_{\{0.1,0.1,0.2,0.3,0.3\}}$	0	32.6
		1.0	29.8
		2.0	5.5
		3.0	8.4
		4.0	10.3
		5.0	11.8
		$\geq$ 6.0	$\geq$ 13.0
0.8	$\{1\}_1$	0	11.4
		1.0	9.9
		2.0	3.4*
		$\geq$ 3.0	$\geq$ 4.8*
	$\{1,2,3\}_{1/3}$	0	11.4
		1.0	9.9
		2.0	6.2
		3.0	8.2
		4.0	9.5
		$\geq$ 5.0	$\geq$ 10.7
	$\{1,2,3,4,5\}_{\{0.1,0.1,0.2,0.3,0.3\}}$	0	11.4
		1.0	9.9
		2.0	10.4
		3.0	14.4
		4.0	16.6
		5.0	18.1
		$\geq$ 6.0	$\geq$ 19.2

(in units of mean task execution time. * indicates $e^{\lambda_F t} - 1$ dominates the determination of T_{out}.)

Table 1: Best timeout periods w. r. t. the task characteristic and the initial state of node i. $\alpha_{ht} = 5 \times 10^{-2}$ and $\lambda_F = 10^{-2}$.

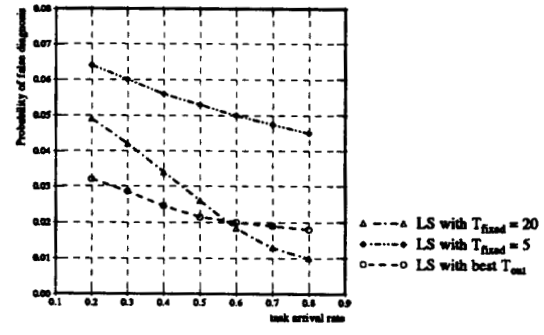

Figure 4: Performance comparison w.r.t. P_F for different timeout periods in a 16-node ($N_n = 16$) system. $\lambda_F = 10^{-2}$, $\mu_F = 0.1$, and $\alpha_{ht} = 0.05$.

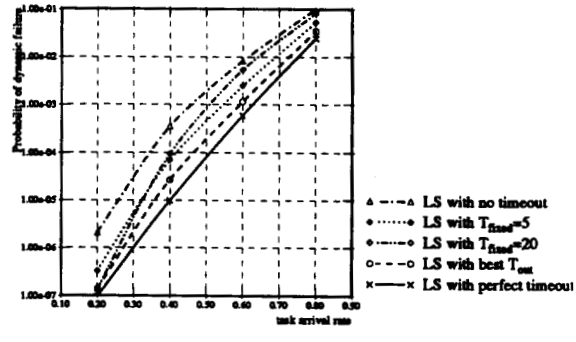

Figure 5: Performance comparison w.r.t. P_{dyn} for different timeout periods in a 16-node ($N_n = 16$) system. $\lambda_F = 10^{-2}$, $\mu_F = 0.1$, and $\alpha_{ht} = 0.05$.