Incorporation of Optimal Timeouts into Distributed Real-Time Load Sharing

Chao-Ju Hou, *Member, IEEE*, and Kang G. Shin, *Fellow, IEEE*

***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 usually 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 $T_{\rm out}^{\langle i \rangle}$ for node i as a hypothesis testing problem, and maximize the probability of detecting node failures subject to a pre-specified probability of falsely diagnosing a healthy node as faulty.**

The parameters needed for the calculation of $T_{\rm out}^{\langle i \rangle}$ are estimated on-line by node i using the Bayesian technique and are piggy-backed in its region-change broadcasts. The broadcast information is then used to determine $T_{\rm out}^{\langle i \rangle}$. If node n has not heard from node i for $T_{\rm out}^{\langle i \rangle}$ since its receipt of the latest broadcast from node i, it will consider node i failed, and will not consider any task transfer to node i until it receives a broadcast message from node i again. On the other hand, to further reduce the probability of incorrect diagnosis, each node n also determines its own timeout period $T_{\rm out}^{\langle n \rangle}$, and broadcasts its state not only at the time of state-region changes but also when it has remained within a broadcast interval throughout $T_{\rm out}^{\langle n \rangle}$.

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, as compared to the other algorithms either without any timeout mechanism or with a fixed timeout period.

***Index Terms*—Deadlines, real-time systems, load sharing, node failures, timeout, continuous-time Markov chains, hypothesis testing, randomization, Bayesian parameter estimation, performance evaluation.**

I. Introduction

The availability of inexpensive, high-performance processors and memory chips has made it attractive to use distributed computing systems for real-time applications. However, 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 minimization of the probability of tasks missing their deadlines, which is referred to as the *probability of dynamic failure*, P_{dyn} [1].

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 complete in time the task under some local scheduling discipline. The minimum-laxity-first-served (MLFS) discipline is shown in [2] 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 the deadline of one or more tasks in its queue is to be missed 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 [3]–[5], bidding/state probing [6]–[11], or aperiodic state-region change broadcasts [12]–[16]. The algorithms based on 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, introducing time and communication overheads, and may thus be detrimental to the timely completion of real-time tasks. Moreover, the performance of these algorithms is very sensitive to the variation of communication delay.

An algorithm that requires to update the state information only in case of state-region changes has 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.

Manuscript received February 24, 1992; revised December 21, 1992. This work was presented in part in the *Proc. 26th Hawaii Int. Conf. on Syst. Sci.*, Jan. 1993, pp. II603–610, © 1993 IEEE Computer Society. The work was supported in part by the NSF under Grants MIP-9203895 and MIP-9012549, and the ONR under Grants N00014-91-J-1115 and N00014-92-J-1080.

The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109–2122.

IEEE Log Number 9214054.

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

- The performance is susceptible to node failures. If node 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 [14] proposed the concept of the buddy sets and the preferred lists to reduce the undesirable effects of the first problem. In another paper [16], 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.

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 [16]. A timeout mechanism can be incorporated into this kind of LS algorithm as follows. 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, $T_{\text{out}}^{\langle i \rangle}$, since its receipt of node i's latest broadcast, and will thereafter not send its overflow task(s) to node i even if node i is observed (through the state information gathered in region–change broadcasts) to be capable of completing the task(s) in time. Obviously, the determination of $T_{out}^{\langle i \rangle}$ 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 the period $T_{\text{out}}^{\langle i \rangle}$:

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 oscillates within two adjacent broadcast thresholds, or remains in a broadcastthreshold interval.

The determination of $T_{\text{out}}^{\langle i \rangle}$ thus involves a trade-off between the performance improvement gained by reducing $T_{\text{out}}^{\langle i \rangle}$ (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 $T_{\text{out}}^{\langle i \rangle}$ by maximizing the probability of detecting node failures subject to a pre-specified 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 and has thus been silent for $T_{\text{out}}^{\langle n \rangle}$. That is, with a few extra timely broadcasts, the undesirable effect of incorrect diagnosis can be reduced while enabling fast detection of node failures.

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 $T_{\text{out}}^{\langle i \rangle}$) dynamically vary with the system load, the task attributes, and the initial state of the node. Thus, the calculation of $T_{\text{out}}^{\langle i \rangle}$ 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" (both external and transferred-in) task arrival rate and distributions of task execution time and laxity, and convey the estimated parameters to other nodes in its buddy set by piggy-backing them in state–region change broadcasts. This information will then be used by the other nodes to calculate $T_{\text{out}}^{\langle i \rangle}$.

The LS algorithm in [16] 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. One can, of course, include this timeout mechanism in other existing LS algorithms.

The rest of the paper is organized as follows. Section II outlines the LS algorithm and the proposed timeout mechanism. Section III and IV establishes a theoretical basis for the calculation of optimal $T_{\text{out}}^{\langle i \rangle}$. The HT formulation is treated in Section III, while the probability distribution needed in the HT formulation is derived in Section IV by applying the randomization technique to a continuous-time Markov chain which characterizes the state evolution. Section V discusses how the parameters needed for the timeout mechanism are estimated on–line by using the Bayesian technique. Section VI presents and discusses representative numerical examples, and the paper concludes with Section VII.

II. The Proposed Algorithm

We proposed in [16], [17] a decentralized, dynamic LS algorithm for distributed real-time systems without considering node failures. In this algorithm, we used the concept of buddy sets [14], time-stamped region-change broadcasts, and Bayesian decision theory to minimize the probability of transferring an overflow task to an "incapable" (of meeting the task deadline) node. In this section, we first state the assumptions made about the system under consideration, and summarize the proposed LS 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 was presented in [18]. We also assume that the underlying communication subsystem supports reliable broadcasting [19], [20] so that a nonfaulty node can correctly broadcast its state change to all other nonfaulty nodes in the system. Finally, each node is assumed to have a constant exponential failure rate λ_F. (This assumption is commonly used in reliability evaluation [21], [22].)

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 will be discussed in Section VI-A. This approximation is used to facilitate the derivation of $T_{\text{out}}^{\langle i \rangle}$ and the on-line estimation of parameters needed for the calculation of $T_{\text{out}}^{\langle i \rangle}$.

$\{p_i(j), 1 \le j \le E_{\max}\}$: 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. This distribution will be estimated on–line by each node i.
$\{\hat{p}_i(j), 1 \le j \le L_{\max}\}$: 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.[2] This distribution will also be estimated on-line by each node i.
CET_i: the cumulative task execution time (CET) on node i.
$\underline{T_Q} = (T_1; T_2; \cdots; T_{L_{\max}})$: the record for task execution times of the sorted queue on a node, where $T_j \triangleq e_1^j e_2^j \cdots e_{j+1}^j$ is an execution-time record of tasks with laxity $j \in \{1, \cdots, L_{\max}\}$ currently queued on a node, and $e_k^j \in \{0, \cdots, E_{\max}\}$, $1 \le k \le j+1$, is the execution time required by the kth task among those laxity-j tasks in the queue. ($e_k^j = 0$ if there are less than k laxity-j tasks in the queue.) the reason that T_j is expressed in the form $e_1^j e_2^j \cdots e_{j+1}^j$ is because a node can queue, under the MLFS discipline, at most $j+1$ tasks with laxity j, in which case all but the last laxity–j task require one unit of execution time and there are no tighter-laxity tasks queued at the node.
O_i: the observation of CET_i made by some node $j \ne i$.
$p_C(\cdot \mid O_i)$: the posterior distribution of CET_i given the observation O_i. This posterior distribution is constructed by each node $j \ne i$ with the state samples collected via region-change broadcasts.
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.
$T_{\text{out}}^{\langle i \rangle}$: 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 latest broadcast.

A. LS With Region-Change Broadcasts

For completeness the operations of a node's task scheduler which employs the LS algorithm described in [16], [17] are given in Fig. 1. 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 out, 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, to share loads system-wide, the buddy sets *overlap* with one another so that it is possible for a node to transfer its overflow task(s) to some other node(s) not in its own buddy set. That is, the overflow tasks within one buddy set are shared by capable nodes in the system, instead of overloading a few nodes within one buddy set [14].

Region-Change Broadcasts: the K_t state regions defined by $K_t - 1$ thresholds, $TH_1, TH_2, \cdots, TH_{K_t-1}$, 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 $1 \le k \le \lceil \frac{K_t}{2} \rceil - 1$.[3] The state information kept at each node is thus up–to–date as long as the broadcast delay is not significant.

Bayesian Analysis: The state information collected through region-change broadcasts may become outdated due to the delay in collecting it. That is, a node's observation O_i may be different from CET_i at the time of making a LS decision. In [16], 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 (16) 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[4] that $O_i = k$ if $TH_k \le CET_i < TH_{k+1}$, $0 \le k < K_t - 1$, where $TH_0 \Delta = 0$ and $TH_{K_t} \Delta = \infty$.

The only effect of the region-change broadcast delay is that the broadcast messages may not get delivered immediately and may thus become obsolete upon arrival at their receiver nodes. The correctness of all samples gathered is, however, not affected by the broadcast delay. Revise: Bayesian analysis 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 capability of completing a task with laxity ℓ in time, i.e., node

[2]We may include non real-time tasks in the task set by choosing $L_{\max}$ to be one time unit larger than the actual maximum laxity of real-time tasks and assuming that all non real-time tasks have laxity $L_{\max}$.

[3]The reason for not broadcasting the change of state region when a node's load crosses an odd-numbered threshold is to reduce the network traffic resulting from broadcasts.

[4] The reason for discretizing CET_i with O_i is to reduce the size of the observation space.

```
At each node $n$:
When a task $T_i$ with execution time $E_i$ and laxity $\ell_i$ arrives at node $n$:
    determine the position, $j_p$, in the task queue $Q^\dagger$ such that $\ell_{j_p-1} \le \ell_i \le \ell_{j_p}$;
    if current_time + $\sum_{k=1}^{j_p-1} E_k \ge \ell_i$ then
     begin
        receiver_node := table_lookup($\underline{O}$:observation, $\ell_i$:laxity);
        transfer task $T_i$ to receiver_node;
     end
    else
     begin
        queue task $T_i$ at position $j_p$;
        for $k = j_p + 1, length(Q)$
          begin
            if current_time + $\sum_{\ell=1}^{k-1} E_\ell \ge \ell_k$ then
             begin
                receiver_node := table_lookup($\underline{O}$:observation, $\ell_k$:laxity);
                dequeue and transfer $T_k$ to receiver_node;
             end
          end
        if current_CET crosses $TH_{2k}$, $1 \le k \le \lceil \frac{K_t}{2} \rceil - 1$, then
          begin /* region-change broadcasts: $TH_1, \cdots, TH_{K_t-1}$ are thresholds */
             broadcast (1) time-stamped $CET_n$'s, and (2) $\lambda_n$, $\{p_n(j)\}$, $\{\hat{p}_n(k)\}$ to all the other
              nodes in its buddy set;
             calculate $T_{out}^{<n>}$ and reset timeout_clock$_n$;
          end
     end
    $(\lambda_n, \{p_n(j)\}, \{\hat{p}_n(k)\})$ = parameter_update($E_i$, $\ell_i$, $t_i$:interarrival_time);

When a broadcast message arrives from node $i$:
    update observation of node $i$'s state, $O_i$;
    if node $i$ is disabled then
       enable node $i$;
    else
       record $(O_i, CET_i)$ pair needed for Bayesian decision analysis;
    calculate $T_{out}^{<i>}$ using $\lambda_i$, $\{p_i(j)\}$ and $\{\hat{p}_i(k)\}$, and reset timeout_clock$_i$;

At every clock tick:
    current_CET := current_CET − 1;
    if (current_CET crosses $TH_{2k}$, $1 \le k \le \lceil \frac{K_t}{2} \rceil - 1$) or (timeout_clock$_n$ expires) then
     begin
       broadcast (1) time-stamped $CET_n$'s, and (2) $\lambda_n$, $\{p_n(j)\}$, and $\{\hat{p}_n(k)\}$ to all the other
        nodes in its buddy set;
       calculate $T_{out}^{<n>}$ and reset timeout_clock$_n$;
     end
    if timeout_clock$_i$ expires then
       disable node $i$;

At every $T_p$ clock ticks: /* table update */
    update the table of loss-minimizing decisions by Bayesian decision analysis;
```

†The task queue Q is ordered by task laxities.

Fig. 1. Operations of the task scheduler on each node.

n chooses the node i with the largest value of

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

as the node for an overflow task with laxity ℓ to be transferred to.

B. Incorporation of the Timeout Mechanism into LS

As mentioned in Section I, 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 node. Some simple techniques could be used to determine whether **S1** or **S2** occurs: node n may determine whether node i failed or not by probing it at the time of making a LS decision, but in such a case, it has to wait for node i's response before making the LS decision. This could introduce unacceptably long delays to those tasks to be transferred, the negative effect of which increases significantly with communication delay [10]. On the other hand, node n may arbitrarily choose a fixed timeout period *a priori*. In this case, node n runs the risk of 1) hastily and falsely diagnosing a healthy node as failed if the timeout period chosen is too small and 2) failing to detect node failures in a timely manner if the chosen period is too large. Actually, as will be demonstrated in Section VI, the best value of $T_{\text{out}}^{\langle i\rangle}$ varies drastically with the attributes of tasks arriving at node i, and the state node i was initially in. (We will compare the performance of using the best $T_{\text{out}}^{\langle i\rangle}$ calculated against that of using some pre-specified timeout period in Section VI-D.) This calls for a timeout mechanism which dynamically adjusts $T_{\text{out}}^{\langle i\rangle}$ 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 submechanisms.

On-Line Parameter Estimation: Node i records on-line the inter-arrival 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, $\{p_i(j), 1 \le j \le E_{\max}\}$, and $\{\hat{p}_i(j), 1 \le j \le L_{\max}\}$. Application of the Bayesian technique to estimate these parameters will be discussed in Section V.

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 $T_{\text{out}}^{\langle i\rangle}$. A theoretical basis for determining $T_{\text{out}}^{\langle i\rangle}$ will be established in Sections III and IV by using the hypothesis testing (HT) and randomization techniques. Conceptually, the problem of determining $T_{\text{out}}^{\langle i\rangle}$ is first formulated as a HT problem by making a trade off 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 first modeling the state evolution of node i as a continuous-time Markov chain and then applying the randomization technique on the constructed Markov chain to derive the distribution of interest.

Node n considers node i failed if it has not heard from node i (via region-change broadcasts) for $T_{\text{out}}^{\langle i\rangle}$ 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 $T_{\text{out}}^{\langle n\rangle}$ at the time of broadcasting a state-region change. If node n has remained within a broadcastthreshold interval and has been silent for $T_{\text{out}}^{\langle n\rangle}$, it broadcasts an extra message to inform other nodes of its fault-free (or "I am alive") status.

III. Determination of the Optimal Timeout Period

In this section and the next section, we will establish a theoretical basis for the determination of $T_{\text{out}}^{\langle i\rangle}$. 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 II, we discussed how the on-line estimated parameters are broadcast. Estimation of λ_i, $\hat{p}_i(j)$'s, and $p_i(j)$'s is the subject of Section V.

The problem of determining $T_{\text{out}}^{\langle i\rangle}$ is formulated as a HT problem. The probability distribution needed to solve the HT problem is then derived using the randomization technique in Section V.

Recall that $T_{\text{out}}^{\langle i\rangle}$ is the timeout period after which node i will be diagnosed as failed by node $n \neq i$ if no broadcast message from node i has been received since the last broadcast. As mentioned earlier, there are two possible scenarios, **S1** and **S2**, that no broadcast message from node i will be received by node n within $T_{\text{out}}^{\langle i\rangle}$. The determination of $T_{\text{out}}^{\langle i\rangle}$ requires to make a trade-off between these two possibilities, 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 $Ob(t) \sim p_0$,
H_1: **node** i is faulty $Ob(t) \sim p_1$,

where $\sim$ denotes that p_0 and p_1 are the pdf of $Ob(t)$ under the hypothesis H_0 and H_1, respectively. p_0 and p_1 can be expressed as

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

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 [23], 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

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

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

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

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

$$\begin{aligned} P_D(\delta) &= E_1(\ \delta(Ob(t))) \\ &= P_1(\pi_1 \cdot p_1(Ob(t)) \geq \pi_0 \cdot p_0(Ob(t))) \\ &= P\left(p_0(Ob(t)=0) \leq \frac{\pi_1}{\pi_0}\right). \end{aligned} \tag{3.4}$$

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

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

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

IV. Derivation of $P(T_{nb} \geq t|$ Node i is Operational)

We now use the randomization technique [24]–[26] to calculate $P(T_{nb} \geq t\,|\,\text{node } i$ 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 $P(T_{nb} \geq t\,|\,\text{node } i$ is operational) using the randomization technique.

A. System Model

The state/CET evolution of a node is modeled as a continuous-time Markov chain $\{X(t), t \geq 0\}$ on a finite state space S. Transitions in the Markov chain are characterized by the generator matrix $Q = (q_{ij})$, where q_{ij}, $0 \leq i, j \leq N$, is the transition rate from state i to state j. The parameters needed in the model are $\lambda_i, \{p_i(j), 1 \leq j \leq E_{\max}\}$, and $\{\hat{p}_i(k), 1 \leq k \leq L_{\max}\}$, all of which are estimated on-line by each node i and piggy-backed in region-change broadcasts to the other nodes in its buddy set.

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 $P(T_{nb} \geq t\,|$ node i is operational) obtained from the corresponding $M^{[i]}/E_K/1$ model is very close to that obtained from $M^{[i]}/E_{K+1}/1$ model. In Section VI-B, $K \geq 5$ is shown to satisfy the above criterion for all combinations of task attributes studied. Each accepted/queued task contributes Km service stages with probability $p_i(m)$, $1 \leq m \leq E_{\max}$, and each service stage is consumed at (an exponential) rate K.

Definition of State: The state of node i is defined as $\underline{H} = (H_0; H_1; H_2; \cdots; H_{L_{\max}})$, where $H_j \triangleq h_1^j h_2^j \cdots h_{j+1}^j$ is a sequence of $j+1$ numbers with $h_k^j \in \{0, \cdots, KE_{\max}\}$ representing the number of service stages contributed by the kth laxity-j task in the node's queue. H_j can be viewed as a record of all laxity-j tasks currently queued on node i. 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 the equation found at the bottom of the page denote the laxity of the task currently under service, where

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

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 one remaining service stage. $L_{\text{now}}((0; 00; 440; 8000)) = 2$ indicates that the task to be served next is the one with two time units of laxity and four service stages if there are no new laxity-1 task arrivals before the next state transition.

Under the nonpreemptive 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.

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

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

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

or,

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

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 $\leq Kj$, i.e.,

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

or,

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

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;$\underline{4}$000) 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**. The state (0;48;000;8000) is allowed, while (0;48;000;1000) is not, because in the latter state the task with 3 time units of laxity (represented by 1) is currently in service, and thus, the task with 1 time unit of laxity and 8 service stages (represented by 8) cannot be queued.

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 [25] — 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. (Recall that some tasks originally queued on the node may have their laxities missed as a result of inserting a newly-arrived task into the sorted task queue, and must thus be transferred out.)

The Transition Caused by Task Acceptance: First, we consider the transition resulted from task acceptance. Assume that the system is in state $\underline{H}$, and will make a transition to state $\underline{H}'_{\ell,Km} \triangleq (H'_0; H'_1; \cdots; H'_\ell; \cdots; H'_{L_{\max}})$ upon acceptance of a task with laxity ℓ and execution time m, where $1 \leq \ell \leq L_{\max}$ and $1 \leq m \leq E_{\max}$. Then

1) $c'_j = c_j$ (or equivalently, $H'_j = H_j$), $1 \leq j \leq \ell - 1$, i.e., the CET contributed by tasks with laxity $\leq \ell - 1$ 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 ($\ell+1 \leq j \leq L_{\max}$) replaced by 0 (so $c'_j \leq c_j$). 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 $H'_\ell = h_1^\ell \cdots h_{last(H_\ell)}^\ell \; Km \; 0 \cdots 0$, 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 nonpreemptive policy) is

$$\begin{aligned} q_{\underline{H},\underline{H}'_{\ell,Km}} &= \lambda_i \hat{p}_i(\ell) p_i(m) \cdot \text{CheckCet}(\ell) \\ &\quad \prod_{\substack{t=\ell+1 \\ t \notin zero(\underline{C})}}^{L\max} \{\text{comp}(H_t, H'_t) \cdot \text{ Task\_Not\_Transfer } (t) \\ &\quad + (1 - \text{comp}(H_t, H'_t)) \cdot \text{ Task\_Transfer } (t)\}, \quad (4.1) \end{aligned}$$

where Check_Cet(ℓ), zero($\underline{C}$), comp(H_t, H'_t), Task_Not_Transfer(t), and Task_Transfer(t) are expressed at the bottom of the next page.

The physical meanings of (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, i.e., $K\ell \geq \sum_{j=0}^{\ell} c_j$, 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 $> \ell$ (i.e., no preemption).
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, i.e., $Kt \geq \sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-1} h_j^t$ if the laxity of the currently executing task $\leq t$, or, $Kt \geq \sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-1} h_j^t + h_1^{L_{\text{now}}(\underline{H})}$ if the laxity of the currently executing task $> t$.

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 if and only if both $\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-i-1} h_j^t \leq Kt$ and $\sum_{j=0}^{t-1} c'_j + \sum_{j=1}^{last(H_t)-i} h_j^t > Kt$ hold. The cases with $t < L_{\text{now}}(\underline{H})$ and $t = L_{\text{now}}(\underline{H})$ can be similarly reasoned about.

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 $\underline{H}'_{\ell,-1} = (H'_0; H'_1; \cdots; H'_\ell; \cdots; H'_{L_{\max}})$, with transition rate

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

where

1) if $(h_1^\ell - 1) \notin \{Km : 1 \leq m \leq E_{\max}\} \bigcup \{0\}$,

$$H'_\ell = (h_1^\ell - 1)h_2^\ell \cdots h_{\ell+1}^\ell, \text{and } H'_j = H_j \forall j \neq \ell;$$

2) if $(h_1^\ell - 1) \in \{Km : 1 \leq m \leq E_{\max}\} \bigcup \{0\}$,

$$\begin{aligned} H'_{L_{\max}} &= \underline{0}, \\ H'_j &= H_{j+1} = h_1^{j+1} \cdots h_{j+1}^{j+1}, \\ &\qquad \forall j \in [0, \ell-2] \bigcup [\ell, L_{\max} - 1], \\ H'_{\ell-1} &= \begin{cases} (h_1^\ell - 1)h_2^\ell \cdots h_\ell^\ell, & \text{if } h_1^\ell > 1, \\ h_2^\ell \cdots h_\ell^\ell 0, & \text{if } h_1^\ell = 1. \end{cases} \end{aligned}$$

The last nonzero transition rate is

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

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 (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. The transition (0;10;400;4800) $\rightarrow$ (0;10;400;48$\underline{4}$0) is not possible, because the newly-arrived task with laxity $\ell = 3$ (represented by the underlined 4) will not be accepted (i.e., Check_Cet$(\ell) = 0$, because $\sum_{j=0}^{\ell} c_j > K\ell$). The transition (0;10;400;4800) $\rightarrow$ (0;10;480;$\underline{4}$000) is not possible either,

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

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

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

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

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

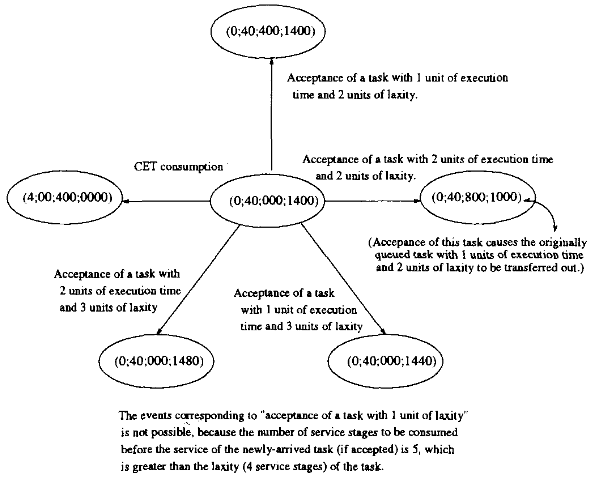
(0;40;400;1400)
Acceptance of a task with 1 unit of execution time and 2 units of laxity.
CET consumption
Acceptance of a task with 2 units of execution time and 2 units of laxity.
(4;00;400;0000)
(0;40;000;1400)
(0;40;800;1000)
(Accepance of this task causes the originally queued task with 1 units of execution time and 2 units of laxity to be transferred out.)
Acceptance of a task with 2 units of execution time and 3 units of laxity
Acceptance of a task with 1 unit of execution time and 3 units of laxity
(0;40;000;1480)
(0;40;000;1440)
The events corresponding to "acceptance of a task with 1 unit of laxity" is not possible, because the number of service stages to be consumed before the service of the newly-arrived task (if accepted) is 5, which is greater than the laxity (4 service stages) of the task.

Fig. 2. Possible state transitions from state $(0;40;000;1400)$ in a system model with $L_{max} = 3$, $E_{max} = 2$, and $K = 4$.

because the task queued with 3 time units of laxity and 4 service stages (represented by the underlined 4) must also be transferred (in addition to the task with 3 units of laxity and 8 service stages) after inserting the newly-arrived task. Similarly, the only possible transitions from state (0;40;000;1400) are to (4;00;400;0000), (0;40;400;1400), (0;40;800;1000), (0;40;000;1440), (0;40; 000;1480), and (0;40;000;1400) with transition rate K, $\lambda_i p_i(1)\hat{p}_i(2)$, $\lambda_i p_i(2)\hat{p}_i(2)$, $\lambda_i p_i(1)\hat{p}_i(3)$, $\lambda_i p_i(2)\hat{p}_i(3)$, and $-(K + \sum_{1\le m\le 2.2\le \ell\le 3}\lambda_i p_i(m)\hat{p}_i(\ell)$, respectively (Fig. 2). The transition (0;40;000;1400) $\rightarrow$ (0;4$\underline{4}$;000;1400) is not possible, because the task currently under service has laxity 3 (i.e., L_{now} ((0;40;000;1400)) = 3), and the newly-arrived task with $\ell = 1$ (represented by the underlined 4) will not be accepted under a nonpreemptive policy (i.e., $\sum_{j=0}^{\ell} c_j + h_1^{L_{\text{now}}(\underline{H})} > K\ell$).

B. 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 $[0,t]$. This technique was introduced in [24]–[26] as a method for computing transient probabilities of Markov processes with finite state spaces, and is summarized in the Appendix.

Recall that in the proposed LS algorithm, a node's states are divided into K_t disjoint subsets: $[0, TH_1]$, $(TH_1, TH_2], \cdots$, (TH_{K_t-1}, ∞), where $TH_k, 1 \le k \le K_t - 1$ are the thresholds of the node's CET. A node will broadcast to other nodes its change of state region whenever its state/CET crosses even-numbered thresholds, $TH_{2j}, 1 \le j \le \lceil \frac{K_t}{2} \rceil - 1$. We thus define $S_j = \{\underline{H} : K \cdot TH_{2(j-1)} \le \sum_{n=1}^{L_{\max}} (\sum_{k=1}^{n+1} h_k^n) < K \cdot TH_{2j}\}$ as the jth 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 $r_j(n,k), 0 \le k \le n+1$, 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, $P(T_{nb} \ge t|$ node i is operational and was in S_j during the last broadcast), $1 \le j \le \lceil \frac{K_t}{2} \rceil$, 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,

$$\begin{aligned} &P(T_{nb} \ge t| \text{ node } i \text{ is operational and was in} \\ &\quad S_j \text{ during the last broadcast}) \\ &\quad = \sum_{n=0}^{\infty} r_j(n, n+1) \cdot P(n \text{ state changes in time } t) \\ &\quad = \sum_{n=0}^{\infty} r_j(n, n+1) \cdot e^{-\Lambda t} (\Lambda t)^n / n! \end{aligned} \tag{4.4}$$

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

The error, e_m, resulting from the truncation of the infinite sum in (4.4) can be easily bounded as

$$\begin{aligned} e_m = &\sum_{n=m+1}^{\infty} e^{-\Lambda t}(\Lambda t)^n/n! \cdot r_j(n, n+1) \le 1 \\ &- \sum_{n=0}^{m} e^{-\Lambda t}(\Lambda t)^n/n!. \end{aligned} \tag{4.5}$$

The $\le$ in (4.5) results from the inequality $r_j(n, n+1) \le 1$. The value of m can be determined *a priori* for any given error tolerance.

$r_j(n,k)$ (and $r_j(n, n+1)$, in particular) can be easily calculated using the recursive approach proposed in [27] (and later studied in depth in [28]). 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

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

So,

$$r_j(n,k,\underline{H}) = \begin{cases} \sum_{\underline{\hat{H}} \in S} r_j(n-1, k-1, \underline{\hat{H}}) \cdot \mathcal{P}_{\underline{\hat{H}},\underline{H}}, & \text{if } \underline{H} \in S_j, \\ \sum_{\underline{\hat{H}} \in S} r_j(n-1, k, \underline{\hat{H}}) \cdot \mathcal{P}_{\underline{\hat{H}},\underline{H}}, & \text{if } \underline{H} \notin S_j, \end{cases} \tag{4.6}$$

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

$$\begin{aligned} r_j(0,1,\underline{H}) &= \begin{cases} 1, & \text{if } \underline{H} \in S_j \text{ and } \underline{H} \text{ is the state} \\ & \text{representation of } \underline{T_Q}, \\ 0, & \text{otherwise,} \end{cases} \\ r_j(0,0,\underline{H}) &= 0, \end{aligned} \tag{4.7}$$

where (4.7) 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, $r_j(n,k) = \sum_{\underline{H} \in S} r_j(n,k,\underline{H})$.

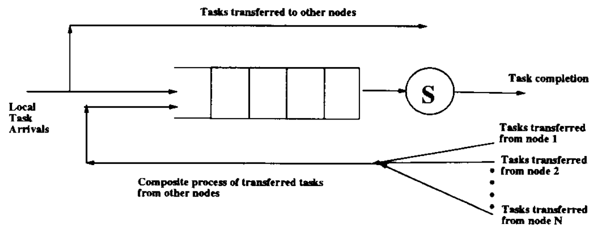
Tasks transferred to other nodes
Local Task Arrivals
S
Task completion
Tasks transferred from node 1
Tasks transferred from node 2
Tasks transferred from node N
Composite process of transferred tasks from other nodes

Fig. 3. A generic queueing model for each node.

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, (4.6) reduces to

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

V. Parameter Estimation

One key issue in applying the timeout mechanism is the on-line estimation of λ_i, $\{\hat{p}_i(j)\}$, and $\{p_i(j)\}$. All on-line estimated parameters will then be piggy-backed in region-change broadcasts to other nodes. We discuss in this section how each node collects samples and makes on-line estimation of these parameters.

A. On-Line Estimation of Composite Task Arrival Rate

The composite task arrival process at a node is composed of external task arrivals and transferred-in task arrivals, the latter of which is itself a composite process of transferred-in tasks from different nodes (see Fig. 3). One difficulty in estimating the composite task arrival rate is that the transferred-in task arrival process (and thus the composite arrival process) is not Poisson even if the external task arrival process is Poisson. This is because:

R1. The probability of sending a task to (or receiving a task from) a node depends on the state of both nodes, making the splitting process non-Poisson.

R2. Task transmission times may not be exponentially distributed, making the process of transferred-in tasks non-Poisson.

Furthermore, even if we assume the composite arrival process to exhibit behaviors similar to a Poisson process, the transferred-in task arrival rate from a node is not known due to the dynamic change of the system state, which calls for on-line estimation of the composite arrival rate.

Bayesian estimation is used for on-line computation of the composite task arrival rate on a node. We approximate the composite task arrival process to be Poisson (in spite of **R1** and **R2**) when the external task arrival process is Poisson. The rationale behind this approximation is the general result of renewal theory [29]: the superposition of increasingly many component processes (i.e., a reasonably large number of nodes) yields (in the limit) a Poisson process. To validate this Poisson approximation, we ran simulations, collected task inter-arrival times on-line under the proposed LS mechanism, and used two statistical testing methods, Kolmogorov–Smirnov and chi-square tests. The simulation results in Section VI-A show this approximation to be acceptable for those systems that are not very heavily loaded and composed of ≥ 12 nodes. We will also consider in Section VI-D the case of hyperexponential task inter-arrival times which represents a system, potentially with bursty task arrivals, and examine to what extent Bayesian estimation remains effective.

Bayesian estimation works as follows [30]. Each node

1) monitors and records its task inter-arrival times continuously,
2) uses the noninformative distribution $g_1(\lambda_i) = const$, and $f(t|\lambda_i) = \lambda_i e^{-\lambda_i t}$ as its prior distribution and likelihood function, respectively,
3) computes the posterior distribution given the time sample t_k with

$$f(\lambda_i|t_k) = \frac{g_k(\lambda_i) \cdot f(t_k|\lambda_i)}{\int_0^\infty g_k(\lambda_i) \cdot f(t_k|\lambda_i) d\lambda_i}, \tag{5.1}$$

4) uses the posterior distribution $f(\lambda_i|t_k)$ for the current sample t_k as the prior, $g_{k+1}(\lambda_i)$, for the next time sample t_{k+1}.

To make the method computationally manageable, it is desirable that both prior and posterior distributions belong to the same family of distributions. The major advantage of using a conjugate prior distribution in estimating λ_i (or any other parameters) is that if the prior distribution of λ_i belongs to this family, then for any sample size n and any values of the observed inter-arrival times, the posterior distribution of λ_i also belongs to the same family. Consequently, the calculation of (5.1) reduces essentially to updating the key parameters of a conjugate distribution. The interested readers are referred to [30] for a detailed account of this.

For the composite arrival rate λ_i with an exponential sampling function, one can show that the γ–distribution

$$\mathcal{G}(\lambda|\alpha, \beta) = \begin{cases} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}, & \text{for } \lambda > 0, \\ 0, & \text{otherwise.} \end{cases}$$

is its conjugate prior distribution, where $\Gamma(\alpha)$ is the gamma function such that $\Gamma(\alpha) = (\alpha - 1)!$ if α is integer. Specifically, given $\mathcal{G}(\lambda_i|\alpha = 1, \beta = t_1)$ as the prior γ–distribution, and given N_S inter-arrival time samples, $t_1, \cdots, t_{N_S}$, the posterior γ–distribution of λ_i becomes:

$$\mathcal{G}(\lambda_i|\alpha = N_S, \beta = \sum_{i=1}^{N_S} t_i).$$

We use the mean of λ_i w.r.t. the posterior distribution as the estimated value which can be expressed in terms of the time samples only, i.e.,

$$E(\lambda_i) = \frac{N_S}{\sum_{k=1}^{N_S} t_k}. \tag{5.2}$$

Thus, the load information provided by the N_S inter-arrival–time samples latest collected can be easily abstracted by updating the key parameters in the conjugate distribution.

B. On-Line Estimation of $p_i(j)$ and $\hat{p}_i(j)$

The other parameters needed for the timeout mechanism are $\{\hat{p}_i(j), 1 \leq j \leq L_{\max}\}$, and $\{p_i(j), 1 \leq j \leq E_{\max}\}$. The estimation techniques used to determine $\{\hat{p}_i(j)\}$ and $\{p_i(j)\}$ are virtually the same, so we will henceforth concentrate on $\{\hat{p}_i(j)\}$.

We treat each task arrival as an experiment whose outcome belongs to one of $L_{\max}$ mutually exclusive and exhaustive categories, and $\hat{p}_i(j)$ as the probability that the outcome belongs to the jth category ($1 \leq j \leq L_{\max}$). Note that $\sum_{j=1}^{L_{\max}} \hat{p}_i(j) = 1$. Suppose N_S independent experiment outcomes are available. Let $Y = (Y_1, \cdots, Y_{L_{\max}})$, where Y_j denotes the number of outcomes that belong to category j among these N_S outcomes. Then the likelihood function is a multinomial distribution with parameters N_S and $\boldsymbol{p}_i = (p_i(1), p_i(2), \cdots, p_i(L_{\max}))$ (i.e., see (5.3) at the bottom of the page). The conjugate family of distributions for the parameter p with a multinomial likelihood function is the Dirichlet distribution with parametric vector $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_{L_{\max}})$ (i.e., see the second equation at the bottom of the page) where $\alpha_0 = \sum_{i=1}^{L_{\max}} \alpha_i$. Specifically, each node assumes the non-informative distribution as the prior distribution of p, e.g., the prior distribution of f is the Dirichlet distribution with $\alpha_j = 1, 1 \leq j \leq L_{\max}$. After collecting N_S samples (i.e., after N_S task arrivals), and computing $(y_1, y_2, \cdots, y_{L_{\max}})$, the posterior distribution of $\boldsymbol{p}_i$, is updated as

$$\mathcal{D}(\boldsymbol{p}_i | (\alpha_1 + y_1, \alpha_2 + y_2, \cdots, \alpha_{L_{\max}} + y_{L_{\max}})).$$

We then use the mean of $\boldsymbol{p}_i$ w.r.t. the posterior distribution as the estimated value, i.e., for $1 \leq j \leq L_{\max}$,

$$E(p_i(j)) = \frac{\alpha_j + y_j}{\sum_{k=1}^{L_{\max}} (\alpha_k + y_k)}. \tag{5.4}$$

Again, the information provided by the most recent N_S task arrivals can be abstracted from the posterior distribution simply by updating the parameters.

VI. Numerical Examples

The proposed timeout mechanism is evaluated in the following sequence:

1) Validation of the Poisson approximation of the composite task arrival process. which facilitates
2) Discussion on the parameters considered/varied in performance evaluation.
3) Discussion on $T_{\text{out}}^{\langle i \rangle}$ (a) w.r.t. task attributes, and (b) w.r.t. the state in which a node was during its latest broadcast.
4) Performance evaluation: we first discuss the performance metrics used and their significance in real-time applications. Second, 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. Then, we study the negative impact of statistical fluctuation in external task arrivals on the proposed LS algorithm (Bayesian estimation in particular).

A. On the Poisson Approximation of Composite Task Arrivals

The composite task arrival rate is estimated on-line under the assumption that the composite task arrival process can be *approximated* to be Poisson.[5] This approximation is conjectured to become more realistic as the system size increases and/or as the system load gets lighter for the following reasons.

1) the superposition of increasingly many component processes yields (in the limit) a Poisson process. That is, as the system size gets larger, a node's state (CET) becomes less dependent on other nodes, the task transfer-out process at a node depends less on other nodes' states, and thus, the renewal assumption gets closer to reality.
2) In the case of Poisson external task arrivals, as the task transfer-out ratio gets small, so does the "disturbance" to the (originally) Poisson external arrival process caused by task transfers.

The validity of this approximation is checked by comparing the hypothesized exponential distribution and the sample cumulative distribution function. Given an estimate of the composite task arrivals being Poisson with arrival rate $\lambda = \frac{n}{\sum_{j=1}^{n} t_j}$, both the Kolmogorov–Smirnov (K-S) test and the chi-square test are used to determine if $t_1, \cdots, t_n$ represent random samples from an exponential distribution.

For completeness, we summarize below the steps of the K–S test used and discuss the data obtained from event-driven simulations. The chi-square test yields results very similar to the K–S test, thus we omit its details of verification steps and only summarize the results at the end of this section.

[5] The same assumption was also used in [10], [14] without any justification.

$$f(y|N_S, p = \begin{cases} \frac{N_S!}{y_1! \cdots y_{L_{\max}}!} p_i(1)^{y_1} p_i(2)^{y_2} \cdots p_i(L_{\max})^{y_{L_{\max}}}, & y \in \mathbf{N}^{L_{\max}}, y_j \geq 0, 1 \leq j \leq L_{\max}, \\ & \text{and } \sum_{j=1}^{L_{\max}} y_j = N_S, \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

$$\mathcal{D}(\mathbf{p}\ |\alpha) = \begin{cases} \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_{L_{\max}})} p_i(1)^{\alpha_1 - 1} \cdots & \mathbf{p} \in R^{L_{\max}}, p_i(j) > 0, \text{ for } 1 \leq j \\ \quad p_i(L_{\max})^{\alpha_{L_{\max}} - 1}, & \leq L_{\max}, \text{ and } \sum_{j=1}^{L_{\max}} p_i(j) = 1, \\ 0, & \text{otherwise,} \end{cases}$$

(The interested readers are referred to [31] and [32] for a detailed account of the Kolmogorov–Smirnov test and the chi-square test, respectively.) We first ran simulations and collect interarrival times on-line until $k = 100$ samples are obtained on each node. Second, we construct the sample (or empirical) distribution function $F_k(t)$ which is defined as the proportion of the observed samples which are less than or equal to t, i.e., let $t_{(1)} < t_{(2)} \cdots < t_{(k)}$ be the values of the order statistics of the sample, then

$$F_k(t) = \begin{cases} 0, & t < t_{(1)}, \\ i/k, & t_{(i)} \le t < t_{(i+1)}, i = 1, \cdots, k-1, \\ 1, & t = t_{(k)}. \end{cases} \tag{6.1}$$

Now we are interested in testing the following two hypotheses:

H_0 : $t_1, t_2, \cdots, t_k$ is a random sample drawn from an exponential distribution with parameter λ, i.e., $F(t) \triangleq \text{plim}_{k\to\infty} F_k(t) = F_\lambda(t)$, where plim denotes "probabilistic limit";

H_1: H_0 is not true; where $F_\lambda(t) = 1 - e^{-\lambda t}$ is the hypothesized exponential distribution. The test statistic D for the K–S test is defined as the maximum difference between $F_k(t)$ and $F_\lambda(t)$, i.e.,

$$D = \sup_{-\infty<t<\infty} |F_k(t) - F_\lambda(t))|.$$

If D is large or large differences exist between $F(t)$ and $F_\lambda(t)$, then we will reject the null hypothesis, H_0. To judge whether or not D is large enough to justify rejecting H_0, we compare D with the critical values D^* of the K–S test [31], [33]. For example, as the sample size $k < 40$, D^* can be calculated as $\frac{1.36}{\sqrt{k}}$ ($= 0.136$ in our case) with the significance level $\alpha_{ks} = 0.05$.[6] If $D > D^*$, we reject H_0; otherwise, we accept H_0 at the significance level α_{ks}.

It turns out that in the case of Poisson external task arrivals, we have $D < D^* = 0.136$ in the K–S test for all combinations of task attributes[7], when the number of nodes in the system ≥ 12, and/or the average task transfer–out ratio < 0.25 —this is always true in our simulations when the average external task arrival rate $\overline{\lambda^{ext}} = \frac{1}{N_n}\sum_{i=1}^{N_n} \lambda_i^{ext} \leq 0.8$. Similarly, we have in the chi-square test, $\chi^2(obs) < \chi^2(0.05) = 7.81$ (i.e., H_0 is accepted) under the conditions specified above, where $\chi^2(obs)$, as D does in Kolmogorov–Smirnov test, measures the deviation of the empirical distribution from the hypothesized distribution, and $\chi^2(0.05)$ is the corresponding critical value at the significance level of 0.05. See Tables I-A and I-B for numerical examples. Since both conditions are satisfied for the proposed LS algorithm, the approximation of exponential interarrival times is acceptable at the significance level $\alpha_{ks} = 0.05$ for the case of Poisson external task arrivals.

B. Parameters Considered/Varied

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

[6] α_{ks} is the probability that H_0 is falsely rejected.

[7] See the next subsection on the parameters varied.

TABLE I-A
VALIDATION OF THE POISSON ASSUMPTION

System size N_n	Average system load λ	Critical value D
8	0.2	0.084
	0.4	0.127
	0.6	0.187
	0.8	0.289
10	0.2	0.076
	0.4	0.092
	0.6	0.121
	0.8	0.203
12	0.2	0.063
	0.4	0.087
	0.6	0.104
	0.8	0.130
16	0.2	0.056
	0.4	0.081
	0.6	0.101
	0.8	0.117

TABLE I-B
VALIDATION OF THE POISSON ASSUMPTION

System size N_n	Average system load $\overline{\lambda}$	$\chi^2(obs)$
8	0.2	5.32
	0.4	6.49
	0.6	7.93
	0.8	8.06
10	0.2	4.68
	0.4	6.35
	0.6	7.42
	0.8	8.26
12	0.2	3.79
	0.4	4.23
	0.6	5.07
	0.8	6.12
16	0.2	2.86
	0.4	3.57
	0.6	4.35
	0.8	5.78

(a) With the Kolmogorov–Smirnov test: if $D < D^* = 0.136$, then the approximation is valid for the significance level 0.05. (b) With the chi-square test: if $\chi^2(obs) = \sum_{j=1}^{C} \frac{(o_i-n_i)^2}{n_i} < \chi^2(0.05) = 7.81$, then the approximation is valid for the significance level 0.05. Note that n_i and o_i are obtained as follows. We first break up the domain of interarrival times (i.e., $(0, \infty)$) into $C = 5$ categories. Under the assumption that $\lambda e^{-\lambda t}$ governs interarrival times, we determine the number, n_i, of t_i's that are expected to fall into category i. Second, we count the number, o_i, of the k=100 time samples obtained from the simulation which actually fall into category i.

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 deterministic 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 $\{e_1, \cdots, e_k\}_{\{p_{e_1}, \cdots, p_{e_k}\}}$

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 $\{p_{e_1}, p_{e_2}, \cdots, p_{e_k}\}$ is condensed to p. Similarly, $\{\ell_1, \ell_2, \cdots, \ell_n\}_{\{\hat{p}_{\ell_1}, \hat{p}_{\ell_2}, \cdots, \hat{p}_{\ell_n}\}}$ 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, $TH_1, \cdots, TH_{K_t - 1}$ used as reference points for broadcasts.

Both 16-node and 64-node regular[8]systems are used in our simulations. The size of buddy set is chosen to be 12, because increasing it beyond 10 was shown in [14] 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.[9]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 $P(T_{nb} \geq t|$ node is operational) thus derived is almost indistinguishable from that derived with $K \geq 6$ (Fig. 4).

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. The case with hyperexponential interarrival times represents a system potentially with bursty task arrivals, and is used to investigate the impact of statistical fluctuation in task arrivals on the performance. The squared coefficient of variation of hyperexponential arrivals (CV^2) is varied from 1 to 91. 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, $p_C(\cdot|O_i)$, given the observation O_i. It is, however, difficult to objectively determine an optimal combination of these design parameters that give accurate posterior distributions while incurring the least communication overhead. We already discussed one method in [16] that determines the design parameters. Although the set of parameters obtained through this method may not be globally optimal, our simulations have shown them to yield good results, as compared to other existing LS algorithms. The interested readers are referred to [16] for a detailed account of this. For the performance study described below, we tuned the design parameters using the method in [16] for each combination of system configuration and task set.

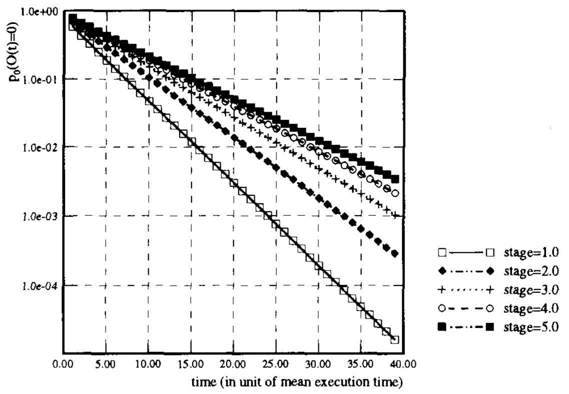

Fig. 4. $P(T_{nb} \geq t \mid$ node i is operational) derived w.r.t. shape parameter K. $\lambda_i = 0.8$, $\overline{ET} = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$ (mean $ET = 1.0$), and $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$. Node i has 4 state regions with each interval equal to 1 (except for the last interval), 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.

We present only those results that we believe are the most relevant, interesting, and/or representative. In spite of a large number of possible combinations of parameters, 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 and a given system configuration is valid over a wide range of combinations of execution time and laxity distributions.

C. Discussion of $T_{\text{out}}^{\langle i \rangle}$

$T_{\text{out}}^{\langle i \rangle}$ increases as $p_0(Ob(t) = 0) = P(T_{nb} \geq t|$ node i is operational) for a given t increases. Figs. 5, 6, and 7 illustrate how $p_0(Ob(t) = 0)$ (and thus, $T_{\text{out}}^{\langle i \rangle}$) varies markedly with the task arrival rate, the state of node i at the time of its latest broadcast, and the length of broadcast intervals, respectively.

As the composite task arrival rate increases, a node tends to cross its broadcast thresholds more often if there is a threshold nearby and to the right of the node's current state. Thus, in Fig. 5, the increase in λ_i yields a smaller $p_0(Ob(t) = 0)$ for a given t (e.g., the more likely a broadcast message is issued within time t). Similarly, as evidenced in the curves labeled as "initial state = 2.0" in Fig. 6 or in the curves labeled as "init. state = 5.0" in Fig. 7, the closer the initial state of a node is to a broadcast threshold, the more likely the node's state will cross the threshold, thus increasing the possibility of a region–change broadcast.

One interesting observation in Fig. 6 is that the probability of a node with initial state 4.0–5.0 broadcasting within time t is smaller than that of a node with initial state 0.0–1.0, when $t \leq 5$ (units of mean execution time), but becomes greater when $t > 5$. This is because a node with initial state 4.0–5.0 will not accept most of its arrived tasks and tends to consume its CET. On the other hand, task acceptance and completion activities may alternate on a node with initial state 0.0–1.0. Consequently, it is more likely for a node with initial state

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

[9]The information about the environment in which the task will execute, e.g., the task owner's current working directory, the privileges/attributes inherited by the task, I/O buffers and messages, etc., is transferred to the remote node. The physical transfer of task may thus require tens of communication packets, while a region–change broadcast would in all likelihood needs at most one packet.

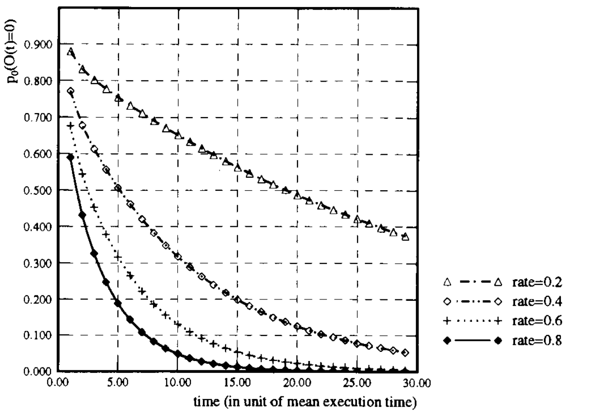

Fig. 5. $P(T_{nb} \geq t \mid \text{node } i \text{ is operational})$ w. r. t. task arrival rate λ_i. $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. Node i has 4 state regions determined by $TH_1 = 1.0, TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$. The state of node i is CET = 1.0 in the last broadcast.

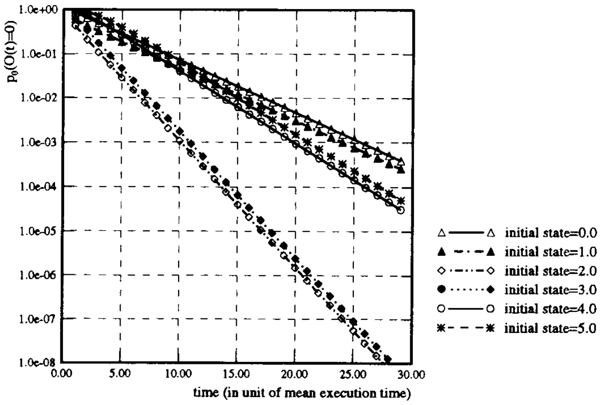

Fig. 6. $P(T_{nb} \geq t \mid \text{node } i \text{ is operational})$ w. r. t. the initial state node i is in. $\lambda = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. Node i has 4 state regions determined by $TH_1 = 1.0$, $TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$.

4.0–5.0 to reach the broadcast threshold after it consumes all its CET (e.g., after 5 units of mean execution time).

Fig. 7 also demonstrates how the size of each broadcast interval affects $p_0(Ob(t) = 0)$. As shown in the curves labeled "init. state = 1.0" in Fig. 7, the larger the broadcast interval is, the less likely a node's state will cross any broadcast threshold, thus resulting in a higher $p_0(Ob(t) = 0)$ for a given t.

Since $p_0(Ob(t))$ varies drastically with the task attributes and the initial state of a node, the on-line calculation of $T_{\text{out}}^{\langle i \rangle}$ is very important to the design of a timeout mechanism. Tables II and III give some numerical values of $T_{\text{out}}^{\langle i \rangle}$ for different task attributes, confidence intervals α_{ht}, and node failure rates λ_F.

D. Performance Evaluation

Performance Measures of Interest: Instead of using the mean task response time as a performance metric, we use two measures that 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. This measure is the key performance metric for the evaluation of LS algorithms for real-time applications.
- The probability of false alarm, P_F: the probability of falsely diagnosing a healthy node as failed. Since each node will refrain itself from sending tasks to the falsely-diagnosed nodes (as well as to the truly failed nodes), P_F is a measure in incorrectly limiting the LS capacity of a system. That is, 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 where a node n (1) considers node i failed if it has not heard from node i for $T_{fixed}^{\langle i \rangle}$ and (2) broadcasts its fault-free status if it has been silent for $T_{fixed}^{\langle i \rangle}$, where $T_{fixed}^{\langle i \rangle}$ 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 under the assumption that the parameter of interest is normally-distributed with unknown mean and variance.

Fig. 8 and Figs. 9–11 plot the curves of P_F and the curves of P_{dyn} for LS with different timeout periods w.r.t. different combinations of λ_F and system size N_n, respectively. 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, in Fig. 8 the case of $T_{fixed}^{\langle i \rangle} = 20$ performs best w.r.t. P_F for medium to heavy system loads ($\lambda_i^{ext} \geq 0.6$, where $5 < T_{\text{out}}^{\langle i \rangle} < 20$ as listed in Table III). For light to medium system loads ($0.2 \leq \lambda_i^{ext} < 0.6$), the case of $T_{\text{out}}^{\langle i \rangle}$ performs best w.r.t. P_F, because usually $T_{\text{out}}^{\langle i \rangle} > 20$ (Table III).
- 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 $T_{\text{out}}^{\langle i \rangle}$.
- The case with the on-line calculated $T_{\text{out}}^{\langle i \rangle}$ outperforms all the other fixed-timeout cases tested in reducing P_{dyn} over a wide range of system load. The case with $T_{fixed}^{\langle i \rangle} = 20$ is inferior to that with $T_{\text{out}}^{\langle i \rangle}$ for medium to heavy system loads due to its inability of early detection of a node

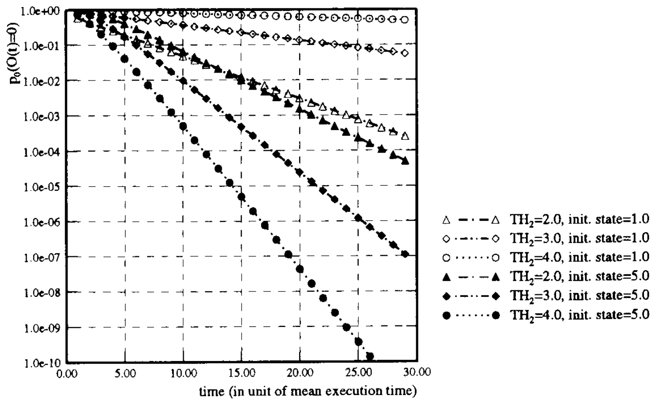

Fig. 7. $P(T_{nb} \geq t \mid$ node i is operational) w. r. t. the length of broadcast interval. $\lambda = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. Node i has 4 state regions determined by $TH_1 = \frac{1}{2}TH_2$, TH_2, and $TH_3 = \frac{3}{2}TH_2$.

failure, thus increasing the possibility of sending overflow tasks to a failed node. The case with $T^{\langle i \rangle}_{fixed} = 5$ is inferior to that with $T^{\langle i \rangle}_{\text{out}}$ because of the undesirable effects of false diagnosis (i.e., deterioration of LS capacity). Frequent 'I am alive' messages in case of $T^{\langle i \rangle}_{\text{out}}$ also consume communication bandwidth and compete with transferred tasks and/or regular broadcast messages for the use of communication medium when the system load ranges from medium to heavy. Thus, there is a definite performance advantage with on-line parameter estimation of task attributes and calculation of $T^{\langle i \rangle}_{\text{out}}$. The performance with $T^{\langle i \rangle}_{\text{out}}$ is, however, worse than the ideal case with immediate and perfect detection of node failures due to the fact that node n might keep sending 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.

- A smaller $T^{\langle i \rangle}_{fixed}$ is preferable as the system becomes more prone to node failures, especially for medium to heavy system loads (i.e., external task arrival rate ≥ 0.5). For example, the case of $T_{fixed} = 5$ outperforms the case of $T_{fixed} = 20$ in a more error-prone system (Fig. 10) as external task arrival rate ≥ 0.6. The performance improvement of frequent timeouts is, however, not as pronounced for a reliable system (Fig. 9). This can be explained by the fact that as the nodes in a system become more prone to node failures, the performance deterioration caused by false diagnoses will get better compensated by the performance improvement due to early detection of node failures.
- As shown in Fig. 10, vs. Fig. 11. the effects of false diagnoses become less pronounced for the case with a smaller $T^{\langle i \rangle}_{fixed}$ (e.g., $T^{\langle i \rangle}_{fixed} = 5$) as the system size gets large. This is due to the fact that a larger system has a larger processing capacity and is thus more resilient to the deterioration of LS capacity caused by false diagnoses.

Impact of Statistical Fluctuation in Task Arrivals on the Effectiveness of Bayesian Estimation: One issue in using Bayesian estimation is to what extent the proposed timeout

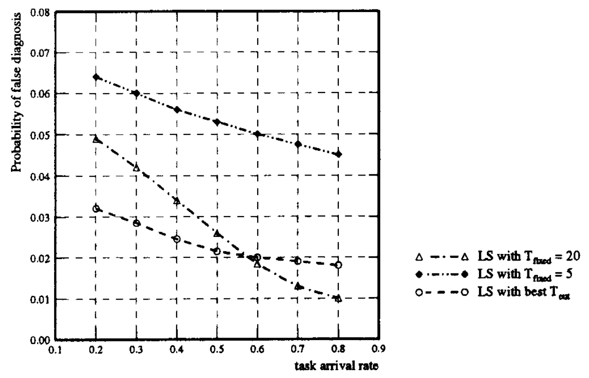

Fig. 8. Performance comparison w.r.t. P_F for different timeout periods in a 16–node ($N_n = 16$) system. External (local) task attributes for node i: $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{\frac{1}{3}}$, and $K = 5$. $\lambda_F = 10^{-2}$, $\mu_F = 0.1$, and $\alpha_{ht} = 0.05$. The task–transfer delay is assumed to be 10% of the execution time of the transferred task. Each node has 4 state regions determined by $TH_1 = 1.0$, $TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$.

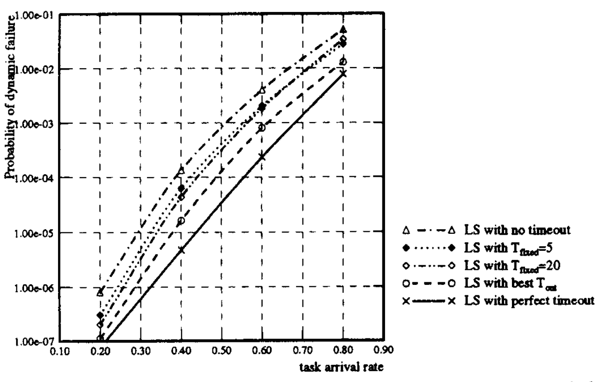

Fig. 9. Performance comparison w.r.t. P_{dyn} for different timeout periods in a reliable 16-node ($N_n = 16$) system. Local task attributes for node i: $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. $\lambda_F = 10^{-3}$, $\mu_F = 0.1$, and $\alpha_{ht} = 0.05$. The task–transfer delay is assumed to be 10% of the execution time of the transferred task. Each node has 4 state regions determined by $TH_1 = 1.0$, $TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$.

TABLE II
BEST TIMEOUT PERIODS W.R.T. THE INITIAL STATE, α_{ht}, AND λ_F. $\lambda_i = 0.8$, AND $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$. NODE i HAS 4 STATE REGIONS DETERMINED BY $TH_1 = 1.0$, $TH_2 = 2.0$, AND $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$.

Distribution of Task Laxity	Initial State	Best T^*_{out}			
		$\alpha_{ht} = 0.20$, $\lambda_F = 10^{-2}$	$\alpha_{ht} = 0.05$, $\lambda_F = 10^{-2}$	$\alpha_{ht} = 0.05$, $\lambda_F = 10^{-3}$	$\alpha_{ht} = 0.01$, $\lambda_F = 10^{-3}$
$\{1\}_1$	0	9.1	11.4	15.6	17.2
	1.0	8.0	9.9	14.4	15.7
	2.0	3.4	3.4	5.3	5.3
	3.0	4.8	4.8	7.1	7.1
$\{1,2,3\}_{1/3}$	0	9.1	11.4	15.6	17.2
	1.0	8.0	9.9	14.4	15.7
	2.0	5.8	6.2	10.4	10.5
	3.0	7.2	8.2	12.0	12.5
	4.0	8.1	9.5	13.1	13.8
	5.0	9.1	10.7	14.1	15.0
$\{1,2,3,4,5\}_{\{0.1,0.1,0.2,0.3,0.3\}}$	0	9.1	11.4	15.6	17.2
	1.0	8.0	9.9	14.4	15.7
	2.0	7.9	10.4	16.3	18.9
	3.0	10.3	14.4	19.3	22.8
	4.0	11.8	16.6	21.1	25.1
	5.0	11.8	18.1	22.3	26.6
	6.0	13.6	19.2	23.2	27.7

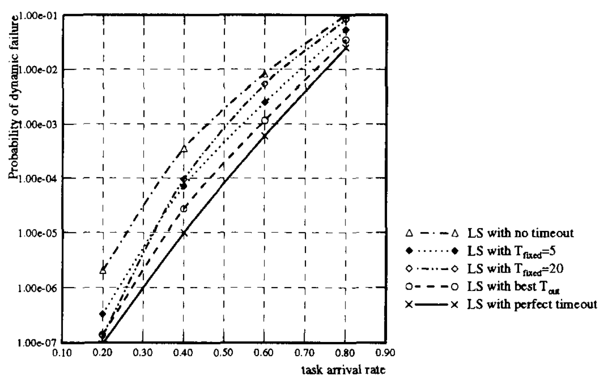

Fig. 10. Performance comparison w.r.t. P_{dyn} for different timeout periods in a noisy 16-node ($N_n = 16$) system. Local task attributes for node i: $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. $\lambda_F = 10^{-2}, \mu_F = 0.1$, and $\alpha_{ht} = 0.05$. The task-transfer delay is assumed to be 10% of the execution time of the transferred task. Each node has 4 state regions determined by $TH_1 = 1.0$, $TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2]$, $S_2 = (2, \infty)$.

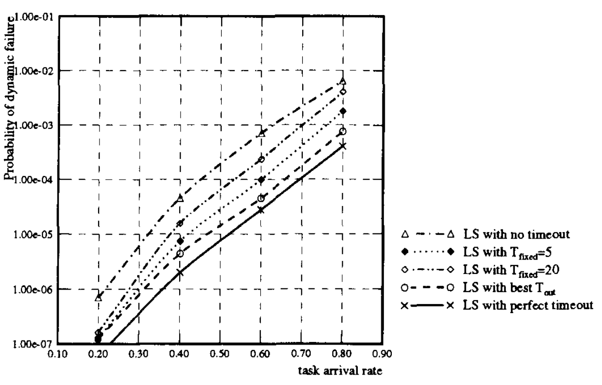

Fig. 11. Performance comparison w.r.t. P_{dyn} for different timeout periods in a noisy 64-node ($N_n = 64$) system. Local task attributes for node i: $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. $\lambda_F = 10^{-2}, \mu_F = 0.1$, and $\alpha_{ht} = 0.05$. The task-transfer delay is assumed to be 10% of the execution time of the transferred task. Each node has 4 state regions determined by $TH_1 = 1.0$, $TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$.

mechanism remains effective when the attributes of tasks arriving at a node randomly fluctuate. We examined this effect on the estimation of composite task arrival rates by simulating different task sets with hyperexponential external task inter-arrival times. Hyperexponential task arrivals represent a system potentially with bursty arrivals, and the degree of statistical fluctuation over a short period is modeled by varying the coefficient of variation (CV) of the inter-arrival times. Specifically, let T_t be the variable of task inter-arrival time. By Chebyshev's inequality,

$$P(|T_t - E(T_t)| \geq nE(T_t)) \leq \frac{CV^2}{n^2};$$

i.e., the smaller CV^2, the less likely T_t will deviate from its mean. Fig. 12 shows the simulation results under a heavy system load $\overline{\lambda_{ext}} = \lambda_i^{ext} = 0.8$ (where the performance is most sensitive to the variation of CV) with the window of the sample size $N_S = 30$. Also shown in Fig. 12 are the curves for the case with no timeout mechanism and the case with immediate, perfect detection of node failures. As CV gets larger, the sample–mean based estimate deviates more from the true composite arrival rate λ_i due to the fact that the variability effect of task burstiness cannot be completely smoothed out. This accounts, in part, for the performance degradation of the proposed LS algorithm. However, the proposed timeout mechanism remains effective for all the task sets tested. For example, in Fig. 12, the performance of LS with $T_{\text{out}}^{\langle i \rangle}$ is still 25% better than LS without any timeout mechanism. This suggests that within a wide range of task burstiness, the λ_i obtained from Bayesian estimation, although it might deviate from the true λ_i, is good for the calculation of $T_{\text{out}}^{\langle i \rangle}$.

VII. 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) calculat-

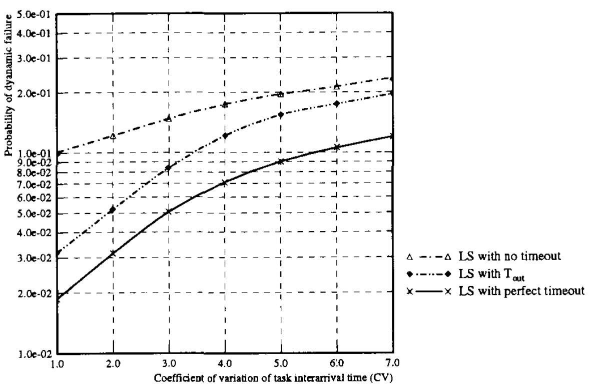

Fig. 12. P_{dyn} vs. CV of external task interarrival times in a 16–node ($N = 16$) system. Local task attributes for node i: $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1.0, 2.0, 3.0\}_{\frac{1}{3}}$, and $K = 5$. $\lambda_F = 10^{-2}$ and $\mu_F = 0.1$. The task–transfer delay is assumed to be 10% of the execution time of the transferred task. Each node has 4 state regions determined by $TH_1 = 1.0$, $TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$.

ing—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.

The validity of all approximations/assumptions used has been checked with simulations. For example, the Poisson approximation of composite task arrivals, which was used in this paper to facilitate the on-line estimation of parameters and the construction of node-level system model (and has also been used without justification in other LS algorithms, e.g., [14], [10], [11]), were checked with the Kolmogorov–Smirnov test. Our simulation results have indicated that this approximation holds at the significance level of 0.05 for a system with a reasonably large, (e.g., ≥ 12) number of nodes and with a small (e.g., ≤ 0.25) average task transfer-out ratio. The negative impact of statistical fluctuation in task arrivals on the proposed timeout mechanism (in particular, Bayesian estimation) is also shown to be tolerable within a wide range of task arrival burstiness; for example, the performance of LS with $T_{\text{out}}^{(i)}$ is still 25% better than LS without a timeout mechanism.

Optimizing the trade-offs involved with the timeout mechanism is an interesting design problem of its own. For example, there is a trade-off between the potential performance improvement gained by reducing the broadcastthreshold interval and the performance deterioration resulting from the traffic overhead of region-change broadcasts. This kind of optimization is a matter of our future inquiry.

Appendix A
Summary Of Randomization Technique

We summarize 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 $0, 1, \cdots, N$. Let $\Lambda \triangleq \max_{0 \leq i \leq N} q_i$, then there exists a discrete—time Markov chain $\{Y_n, n = 0, 1, \cdots\}$ 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

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

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

TABLE III
Best Timeout Periods w.r.t. the Task Characteristic and the Initial State of Node i. $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$. $\alpha_{ht} = 5 \times 10^{-2}$ and $\lambda_F = 10^{-2}$. Node i has 4 State Regions Determined by $TH_1 = 1.0$, $TH_2 = 2.0$, and $TH_3 = 3.0$. $S_1 = [0, 2.0]$, $S_2 = (2.0, \infty)$.

λ	L	Initial state	Best T_{out}
0.4	$\{1\}_1$	0	32.6
		1.0	29.8
		2.0	3.4*
		≥ 3.0	≥ 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
		≥ 5.0	≥ 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
		≥ 6.0	≥ 13.0
0.8	$\{1\}_1$	0	11.4
		1.0	9.9
		2.0	3.4*
		≥ 3.0	≥ 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
		≥ 5.0	≥ 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
		≥ 6.0	≥ 19.2

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

transitions in $(0, t]$, i.e.,

$$\begin{aligned} P(X(t) = i) &= P(Y_{N(t)} = i) \\ &= \sum_{n=0}^{\infty} P(Y_{N(t)} = i \mid N(t) = n) \cdot P(N(t) = n) \\ &= \sum_{n=0}^{\infty} P(Y_n = i) e^{-\Lambda t} (\Lambda t)^n / n!. \end{aligned} \tag{2}$$

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, $\{Y_n, n = 0, 1, ...\}$, 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.

Appendix B
List Of Symbols

Notation used throughout the paper

P_{dyn}: the probability of dynamic failure, or the probability of a task failing to complete before its deadline.

λ_i: the exponential composite task arrival rate at node i.

$\{p_i(j), 1 \leq j \leq E_{max}\}$: the distribution of composite task execution times on node i, where E_{max} is the maximum task execution time measured in clock ticks.

$\{\hat{p}_i(j), 1 \leq j \leq L_{max}\}$: the distribution of composite task laxities on node i measured in clock ticks, where L_{max} is the maximum laxity.

CET_i: the cumulative task execution time (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 $T_j \overset{\triangle}{=} e_1^j e_2^j ... e_{j+1}^j$ is a record of tasks with laxity $j \in \{1, \cdots, L_{max}\}$ currently queued on a node, and $e_k^j \in \{0, \cdots, E_{max}\}$, $1 \leq k \leq j+1$, is the time required to execute the k-th laxity—j task in the queue.

O_i: 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, $1 \leq k \leq K_t - 1$: the state (CET) thresholds for broadcasting region—change messages, where K_t is the number of state regions.

$T_{out}^{<i>}$: the timeout period; node i will be diagnosed as failed if no broadcast message from node i has been received for this period since its latest broadcast. λ_F: the exponential failure rate of a node. $Ob(t)$: the indicator variable for the event that a broadcast message is received within time t.

T_{nb}: the random variable representing the time to node i's next broadcast (measured relative to the time of node i's last broadcast).

Notation used in Section III

H_0 (H_1): the hypothesis that node i is operational (faulty).

π_0 (π_1): the unconditional probability that H_0 (H_1) is true.

p_0 (p_1): the probability density function of $Ob(t)$ under the hypothesis of H_0 (H_1).

$\delta(Ob(t)) \in \{0, 1\}$: the decision function of whether to accept H_0 or H_1 based on $Ob(t)$.

$P_F(\delta)$: the probability that H_0 is falsely rejected.

$P_M(\delta)$: the probability that H_1 is falsely rejected.

α_{ht}: the significance level used in the hypothesis test.

Notation used in Section IV

$\{X(t), t \geq 0\}$: the continuous—time Markov chain on a finite state—space S, which models the state evolution of a node.

$Q = (q_{ij})$: the generator matrix of the continuous–time Markov chain $\{X(t), t \geq 0\}$, where q_{ij}, $0 \leq i, j \leq N$, is the transition rate from state i to state j and $|S| = N + 1$. Note that $q_{ii} = -\sum_{j=0, j\neq i} q_{ij} \overset{\triangle}{=} -q_i$, and $0 \leq q_i \leq \infty$.

K: the rate and the shape parameter K of the Erlang distribution which models the deterministic consumption of CET on node i.

$\underline{H} = (H_0; H_1; H_2; ...; H_{L_{max}})$: the state of a node, where $H_j \overset{\triangle}{=} h_1^j h_2^j ... h_{j+1}^j$ is a sequence of $j + 1$ numbers with $h_k^j \in \{0, \cdots, KE_{max}\}$. h_k^j represents the number of service stages contributed by the k-th laxity-j task in the node queue.

$c_j \overset{\triangle}{=} \sum_{k=1}^{j+1} h_k^j$: the total number of service stages contributed by all laxity-j tasks.

$last(H_j)$: the index of the last nonzero entry in H_j, or, equivalently, the number of nonzero h_k^j's in H_j.

$L\text{now}(\underline{H})$: the laxity of the task currently in service.

$q_{\underline{H}, \underline{H}'_{\ell, Km}}$: the rate of the transition from $\underline{H}$ to $\underline{H}'_{\ell, Km}$ caused by queueing a newly-arrived task with ℓ time units of laxity and m units of execution time.

$q_{\underline{H}, \underline{H}'_{\ell, -1}}$: the rate of the transition from $\underline{H}$ to $\underline{H}'_{\ell, -1}$ caused by the consumption of 1 service stage of the task with laxity ℓ.

$\{Y_n, n = 0, 1, \cdots\}$: the discrete—time Markov chain abstracted from the continuous—time Markov chain $\{X(t), t \geq 0\}$ by randomization.

$\mathcal{P} = (\mathcal{P}_{ij})$: is the transition matrix of the discrete-time Markov chain $\{Y_n, n = 0, 1, \cdots\}$.

$\{N(t), t \geq 0\}$: the Poisson process abstracted from the continuous-time Markov chain $\{X(t), t \geq 0\}$ by randomization, such that $\{Y_{N(t)}, t \geq 0\}$ is probabilistically identical to $\{X(t), t \geq 0\}$.

Λ: the rate of the Poisson process $\{N(t), t \geq 0\}$.

S_j: the j-th state broadcast region. $S_j = \{\underline{H} : K \cdot TH_{2(j-1)} \leq \sum_{n=1}^{L_{max}} \{\sum_{k=1}^{n+1} h_k^n\} < K \cdot TH_{2j}\}$

$r_j(n, k)$, $0 \leq k \leq n+1$: the probability that $\{Y_n\}$ visits the states in S_j k times out of n state changes.

$r_j(n, k, \underline{H})$: the probability that $\{Y_n\}$ stays in S_j k times out of n state changes and the state visited during the last transition is state $\underline{H}$.

Notation used in Section V

$f(t \mid \lambda_i) = \lambda_i e^{-\lambda_i t}$: the likelihood function of interarrival times given λ_i.

$f(\lambda_i \mid t_k)$: the posterior distribution of λ_i given the sample of interarrival time t_k.

$\mathcal{G}(\lambda \mid \alpha, \beta)$: the γ-distribution of λ_i with parameters λ and β.

N_S: the size of statistical samples used to estimate λ_i, $\{\hat{p}_i(j)\}$, or $\{p_i(j)\}$.

$\boldsymbol{Y} = (Y_1, ..., Y_{L_{max}})$: the vector recording the numbers of laxity-j tasks in N_S task arrivals, where Y_j denotes the number of laxity-j tasks in N_S arrivals.

$\boldsymbol{p}_i$: $\boldsymbol{p}_i = (p_i(1), p_i(2), \cdots, p_i(L_{max}))$ is the vector of probabilistic parameters to be estimated.

$f(\boldsymbol{y} \mid N_S, \boldsymbol{p}_i)$: the likelihood function of $\boldsymbol{Y}$ among N_S outcomes given $\boldsymbol{p}_i$.

$\mathcal{D}(\boldsymbol{p}_i \mid \boldsymbol{\alpha})$: the Dirichlet distribution of $\boldsymbol{p}_i$ with parameter $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_{L_{max}})$.

Notation used in Section VI

$F_k(t)$: the empirical distribution function of task interarrival times defined as the proportion of the observed samples which are $\leq t$. $F_\lambda(t) = 1 - e^{-\lambda t}$: the hypothesized exponential distribution.

D: the test statistic for the Kolmogorov–Smirnov test.

α_{ks}: the significance level used in the Kolmogorov–Smirnov test, i.e., the probability that the test falsely rejects the hypothesized distribution.

N_n: the number of nodes in the system.

λ_i^{ext}: the external task arrival rate at node i.

$\overline{\lambda^{ext}}$: the average external task arrival rate. It is an index of system load.

μ_F: the exponential node recovery rate.

$\{e_1, ..., e_k\}_{\{p_{e_1}, ..., p_{e_k}\}}$: the execution time distribution of external tasks, i.e., an external task requires e_i units of execution time with probability p_{e_k} in the task set.

$\{\ell_1, \ell_2, ..., \ell_n\}_{\{\hat{p}_{\ell_1}, \hat{p}_{\ell_2}, ..., \hat{p}_{\ell_n}\}}$: the laxity distribution of external tasks, i.e., an external task has ℓ_i time units of laxity with probability $\hat{p}_{\ell_i}$ in the task set.

$T_{fixed}^{<i>}$: a fixed timeout period used by node i in the simulation.

CV: the coefficient of variation of the hyperexponential interarrival times of external tasks (used in the simulation).

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. Automat. Contr.*, vol. AC-30, pp. 357–366, Apr. 1985.

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

[3] C.-Y. H. Hsu and J. W.-S. Liu, "Dynamic load balancing algorithms in homogeneous distributed systems," in *IEEE Proc. 6th Int. Conf. on Distrib. Computing Syst.*, 1986, pp. 216–223.

[4] J. A. Stankovic, "Simulation of three adaptive, decentralized controlled, job scheduling algorithms," *Computer Networks*, vol. 8, pp. 199–217, 1984.

[5] A. B. Barak and A. Shiloh, "A distributed load–balancing policy for a multicomputer," *Software-Practice and Experience*, vol. 15, no. 9, pp. 901–913, 1985.

[6] K. Ramamritham, J. A. Stankovic and W. Zhao, "Distributed scheduling of tasks with deadlines and resource requirements," *IEEE Trans. Comput.*, vol. 38, no. 8, pp. 1110–1123, Aug. 1989.

[7] T. L. Casavant and J. G. Kuhl, "Analysis of three dynamic distributed load-balancing strategies with varying global information requirements," in *IEEE Proc. 7th Int. Conf. on Distrib. Computing Syst.*, pp. 185–192, 1987.

[8] H.-Y. Chang and M. Livny, "Distributed scheduling under deadline constraints: A comparison of sender–initiated and receiver–initiated approaches," *IEEE Real-time Systems Symposium*, pp. 175–181, 1986.

[9] J. A. Stankovic, K. Ramamritham, and S. Chang, "Evaluation of a flexible task scheduling algorithm for distributed hard real-systems," *IEEE Trans. Comput.*, vol. C-34, no. 12, pp. 1130–1141, Dec. 1985.

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

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

[12] M. Livny and M. Melman, "Load balancing in homogeneous broadcast distributed systems," in *Proc. ACM Comput. Network Performance Symp.*, 1982, pp. 47–55.

[13] A. Hać and X. Jin, "Dynamic load balancing in a distributed system using a decentralized algorithm," *IEEE Proc. 7th Int. Conf. on Distributed Computing Syst.*, Sept. 1987, pp. 170–184.

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

[15] K. G. Shin and Y.-C. Chang, "Load sharing in hypercube multicomputers for real-time applications," in *4th Conf. on Hypercube, Concurrent Comput., and Applicat.*, 1989, pp. 617–622.

[16] K. G. Shin and C.-J. Hou, "Design and evaluation of effective load sharing in distributed real-time systems," in *Proc. Third IEEE Symp. Parallel and Distrib. Processing*, Dec. 1991, pp. 670–677. Also to appear in *IEEE Trans. Parallel Distrib. Syst.*

[17] C. -J. Hou and K. G. Shin, "Load sharing with consideration of future task arrivals in heterogeneous distributed real-time systems," in *IEEE Proc. 12th Real-Time Syst. Symp.*, Dec. 1991, pp. 94–103. Also to appear in *IEEE Trans. Parallel Distrib. Syst.*

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

[19] P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Trans. Comput.*, vol. 37, no. 12, pp. 1654–1657, Dec. 1988.

[20] D. D. Kandlur and K. G. Shin, "Reliable broadcast algorithms for HARTS," *ACM Trans. Comput. Syst.*, vol. 9, no. 4, pp. 374–398, Nov. 1991.

[21] M. Engelhardt and L. J. Bain, "On the mean time between failures for repairable systems," *IEEE Trans. Reliability*, vol. R-35, no. 10, pp. 419–422, Oct. 1986.

[22] M. Alam and U. M. Al-Saggaf, "Quantitative reliability evaluation of repairable phased-mission systems using Markov approach," *IEEE Trans. Reliability*, vol. R-35, no. 10, pp. 498–503, Oct. 1986.

[23] H. V. Poor, *An Introduction to Signal Detection and Estimation.* Dowden & Culver, Inc., 1988.

[24] W. K. Grassmann, "Transient solutions in markovian queueing systems," *Comput. and Operat. Res.*, vol. 4, pp. 47–53, 1977.

[25] D. Gross and D. R. Miller, "The randomization technique as a modeling tool and solution procedure for transient markov processes," *Operat. Res.*, vol. 32, no. 2, pp. 343–361, Mar.–Apr. 1984.

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

[27] E. deSouza e Silva and H. R. Gail, "Calculating cumulative operational time distributions of repairable computer systems," *IEEE Trans. Comput.*, vol. C-35, no. 4, pp. 322–332, Apr. 1986.

[28] ——, "Calculating availability and performability measures of repairable computer systems using randomization," *J. ACM*, vol. 36, no. 1, pp. 171–193, Jan. 1989.

[29] P. J. Kuehn, "Approximate analysis of general queueing networks by decomposition," *IEEE Trans. Commun.*, vol. COM-27, pp. 113–126, 1979.

[30] M. H. DeGroot, *Optimal Statistical Decisions.* New York: McGraw-Hill, Inc., 1970.

[31] ——, *Probability and Statistics*, second ed. Reading, MA: Addison-Wesley, 1986.

[32] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1982.

[33] W. J. Conover, *Practical Nonparametric Statistics*. New York: John Wiley, 1971, p. 397.

Chao-Ju Hou (S'88–M'94) was born in Taipei, Taiwan, Republic of China. She received the B.S.E. degree in Electrical Engineering in 1987 from National Taiwan University, the M.S.E. degree in Electrical Engineering and Computer Science (EECS), the M.S.E. degree in Industrial and Operations Engineering, and the Ph.D. degree in EECS, all from The University of Michigan, Ann Arbor, in 1989, 1991, and 1993, respectively.

She is currently an assistant professor in the Dept. of Electrical and Computer Engineering at the University of Wisconsin, Madison. Her research interests are in the areas of distributed and fault-tolerant computing, real-time computing, queueing systems, estimation and decision theory, and performance modeling/evaluation. She is a member of IEEE Computer Society, ACM Sigmetrics, and Society of Woman Engineer.

Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in Electronics Engineering from Seoul National University, Seoul, Korea in 1970, and both the M.S. and Ph.D degrees in Electrical Engineering from Cornell University, Ithaca, New York in 1976 and 1978, respectively.

He has authored/coauthored over 240 technical papers (more than 100 of these in archival journals) and several book chapters in the areas of distributed real-time computing and control, fault-tolerant computing, computer architecture, robotics and automation, and intelligent manufacturing. In 1987, he received the Outstanding IEEE Transactions on Automatic Control Paper Award for a paper on robot trajectory planning. In 1989, he also received the Research Excellence Award from The University of Michigan. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called **HARTS**, to validate various architectures and analytic results in the area of distributed real-time computing.

From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, and International Computer Science Institute, Berkeley, CA. He is a Professor and Associate Chair of Electrical Engineering and Computer Science for the Computer Science and Engineering Division, The University of Michigan, Ann Arbor, Michigan.

Dr. Shin was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on Real-Time Systems, a Program Co-Chair for the 1992 *International Conference on Parallel Processing*, and served numerous technical program committees. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991–93, is a Distinguished Visitor of the Computer Society of the IEEE, an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, and an Area Editor of *International Journal of Time-Critical Computing Systems*.