Load Sharing with Consideration of Future Task Arrivals in Heterogeneous Distributed Real-Time Systems

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

Abstract— **In a heterogeneous distributed real-time system, some nodes may experience more task arrivals than others, or tasks arrived at some nodes may have tighter laxities than those arrived at other nodes. In such an environment, transferring an unguaranteed task at a node to another node currently with the most abundant resources is not necessarily the best decision. We propose a new load sharing (LS) algorithm for real-time applications which takes into account the effect of future task arrivals on locating the best receiver for each unguaranteed task.**

Upon arrival of a task at a node, the node first checks whether or not it can complete the task in time using the minimum-laxity-first-served discipline. If the node cannot guarantee the arrived task or some of existing guarantees were to be invalidated as a result of inserting the task into its queue, then the node must locate a remote node to which each unguaranteed task will be transferred. The proposed LS algorithm minimizes not only the probability of transferring an unguaranteed task $\mathcal{T}$ to an incapable node with Bayesian analysis, but also the probability that a remote node fails to guarantee $\mathcal{T}$ because of future arrivals of tighter-laxity tasks with queueing analysis. All parameters needed for a node's LS decision are collected/estimated on-line using time-stamped region-change broadcasts and Bayesian estimation. By using time-stamped region-change broadcasts, the collected state information, albeit obsolete, can be used to estimate other nodes' states. Use of Bayesian estimation makes the proposed LS algorithm adaptive to dynamically varying workloads with little computational overhead.

Our simulation results show that the proposed LS algorithm outperforms other existing LS algorithms in minimizing the probability of 1) dynamic failure, 2) task collisions, and 3) excessive task transfers. The performance improvement by the proposed policy over others becomes more pronounced as the degree of system heterogeneity increases.

Index Terms— **Deadlines, real-time systems, load sharing, location policy, queueing analysis, Bayesian parameter estimation, performance evaluation.**

Manuscript received January 9, 1992; revised May 17, 1993. This work was supported in part by the NSF under Grant DMC-8721492 and the ONR under Contract N00014-92-J-1080. An earlieer version of this paper appeared in the *Proceedings of the IEEE 13th Real-Time Systems Symposium*, 1992, pp. 146–155.

C.-J. Hou is with the Department of Electrical and Computer Engineering, University of Wisconsin at Madison, Madison, WI 53706–1691 USA; e-mail: jhou@ece.wisc.edu.

K. G. Shin is 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 USA; e-mail: kgshin@alps.eecs.umich.edu.

IEEE Log Number 9401105.

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 at the nodes in the system and/or the processing power may vary from node to node, thus overloading some nodes temporarily while leaving others idle or underloaded. 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} [2], [3]. Upon arrival at a node of a real-time task with laxity ℓ (where the *laxity* is defined as the latest time a task must start its execution in order to meet its deadline), the node determines whether or not it can guarantee (complete in time) the task under some local scheduling discipline.

The minimum-laxity-first-served (MLFS) discipline is shown in [4] to, on average, outperform others in reducing P_{dyn}, and is hence commonly used as the local scheduling discipline. If a node cannot guarantee a task or some of its existing guarantees are to be invalidated as a result of inserting the task into its schedule, it has to determine candidate receiver(s) for the task(s) to be transferred. Two issues need to be considered when choosing the receiver of an unguaranteed task $\mathcal{T}$: a) the probability of transferring $\mathcal{T}$ to an "incapable node" must be minimized. By "incapable node," we mean a node whose resource surplus is not sufficient to complete $\mathcal{T}$ in time; b) excessive task transfers resulting from task collisions must be avoided. A *task collision* is said to occur if the guarantee of one or more tasks queued at a node are to be invalidated due to the arrival of a new tighter-laxity task. Eager *et al.* [5] referred to the policy which determines whether or not real-time tasks can be guaranteed locally as the *transfer policy*, and the policy which chooses the best receiver of $\mathcal{T}$ as the *location policy*.

Most previous work concentrates on (a), and chooses the most desirable receiver based on the state information collected from periodic/aperiodic state broadcasts [1], [6]–[9] or state probing/bidding [10]–[12]. Moreover, implied in this work (perhaps except for [11], [12]) is the assumption of homogeneous workload distribution among nodes. Under such an assumption, a node which has the most resource surplus or

can complete $\mathcal{T}$ in time is chosen as the receiver of $\mathcal{T}$ without considering *future* task arrivals. This assumption does not always hold, because the distribution that governs task arrivals at different nodes may vary greatly over time and thus the workload distribution is not homogeneous among the nodes. In such a case, whether or not those LS algorithms developed for homogeneous real-time systems will also offer satisfactory performance for heterogeneous systems is questionable. For example, a node with a large composite task arrival rate may become easily overloaded/incapable as a result of a high local arrival rate or the simultaneous transfer of tasks from multiple nodes to the same node. Transferring an "overflow" task $\mathcal{T}$ to such a node may not be a good decision, even if the node were idle/underloaded at the time of locating the receiver of $\mathcal{T}$. Those tasks subsequently arrived at the node may have to be transferred as a result of its acceptance of $\mathcal{T}$, which may result in excessive task transfers. Moreover, this node may also have to transfer $\mathcal{T}$ again due to the subsequent arrival of a tighter-laxity task under the MLFS scheduling discipline. (Tighter-laxity tasks arrived after the arrival of $\mathcal{T}$ but prior to its execution may invalidate the guarantee of $\mathcal{T}$.)

For the reasons above, we must consider the following two issues in guaranteeing tasks on a heterogeneous system.

- **G1)** Minimization of the probability of transferring an unguaranteed task $\mathcal{T}$ to an incapable node, i.e., the receiver of $\mathcal{T}$ is one of those nodes which are observed/estimated to have sufficient resource surplus to guarantee $\mathcal{T}$.
- **G2)** Avoidance of task collisions and/or excessive task transfers, and minimization of the possibility of a task's guarantee being invalidated due to future tighter-laxity task arrivals.

Note that **G2** need not be considered in homogeneous systems since the possibility of a task's guarantee being invalidated by future task arrivals is the same for all candidate receivers. The performance of LS can, however, be improved significantly by incorporating **G2** into LS decisions for heterogeneous systems. Consideration of **G2** is thus the main theme of this paper.

The idea of not necessarily transferring a job to the station currently with the most resource surplus was first proposed by Yum and Schwartz [13, 14] for routing messages in computer communication networks. Stankovic and Ramamritham [11, 12] considered the effect of future task arrivals on the guarantee of transferred-in tasks by (1) exchanging the information containing the percentage of free time among the nodes during the next *window*, the length of which is a tunable design parameter, and (2) using many parameters computed/estimated on-line to determine whether tasks will be transferred or not. They used heuristics and/or exhaustive search for on-line estimation/determination of parameters, and the effectiveness of their approach was evaluated via simulation of a small, six-node system without analytic modeling. By contrast, we shall take **G2** into account using a well-defined analytic framework, and the parameters needed for **G2** are updated on-line with Bayesian estimation theory.

In another paper [1], we proposed a decentralized, dynamic LS algorithm which achieves **G1** in the presence of nonnegligible communication delays by using the concept of buddy sets and region-change broadcasts [8], and Bayesian decision theory. The LS algorithm proposed in [1] will be used as an example of taking **G2** into account. Likewise, one can include **G2** in other existing LS algorithms.

The rest of the paper is organized as follows. Section II outlines the proposed LS algorithm, and formally defines **G1** and **G2** to be considered for guaranteeing real-time tasks. Section III gives a brief description of how **G1** can be achieved in the presence of nonnegligible communication delays [1]. The way each node broadcasts its time-stamped state and on-line estimated parameters (needed for **G2**) is also discussed in Section III. Section IV addresses the theoretical basis for **G2**. Section V discusses how the parameters needed for **G2** are estimated on-line using Bayesian estimation theory. Section VI presents representative numerical examples, and the paper concludes with Section VII.

II. The Proposed Algorithm

In this section, we outline the proposed algorithm, and formally define the two issues, **G1** and **G2**, in guaranteeing real-time tasks. The operations of a node's task scheduler which employs the proposed algorithm are sketched in Fig. 1. To facilitate problem formulation and analysis, we define the following notation.

- $\{p_i(j), j = 1, \cdots, E_{\max}\}$: the distribution of composite (both external and transferred) task-execution time on node i, where $E_{\max}$ is the maximum task-execution time. This distribution will be estimated on-line by each node i.
- $\{\hat{p}_i(j), j = 1, \cdots, L_{\max}\}$: the distribution of composite task laxity on node i, where $L_{\max}$ is the maximum laxity. This distribution will also be estimated on-line by each node i.
- $CET_i(\ell)$: the cumulative execution time (*CET*) on node i contributed by tasks with laxity $\leq \ell$ under the MLFS discipline.
- $O_i(\ell)$: the observation about $CET_i(\ell)$ made by some node $j \neq i$.
- $p_{C_i}(\cdot \mid O_i(\ell))$: the posterior distribution of $CET_i(\ell)$ given the observation $O_i(\ell)$. This posterior distribution is constructed by each node $j \neq i$ with the state samples collected via time-stamped region-change broadcasts. More on this will be discussed in Section III.
- $\overline{V}_{i,\ell}$: the event that future tighter-laxity task arrivals at node i do not invalidate the existing guarantee of a task with laxity ℓ.
- $G_{i,\ell}$: the event that a task with laxity ℓ can be guaranteed by node i even in the presence of future tighter-laxity task arrivals.

The proposed LS algorithm which achieves both **G1** and **G2** works as follows: upon arrival of a task with laxity d at node n, the node checks whether or not it can complete the task in time under the MLFS scheduling discipline, i.e., $CET_n(d) \leq d$. If it can, the task is accepted and queued at node n for execution. If the task cannot be guaranteed locally or some of existing guarantees are to be invalidated by inserting the task into the node's schedule, the node looks up

```
At each node n:
When a task T_i with execution time E_i and laxity ℓ_i arrives at node n:
    determine the position, j_p, in the task queue Q† such that ℓ_{j_p-1} ≤ ℓ_i ≤ ℓ_{j_p};
    if current_time + Σ_{k=1}^{j_p-1} E_k ≥ ℓ_i then
     begin
        receiver_node := table_lookup(Q:observation, ℓ_i:laxity)‡;
        transfer task T_i to receiver_node;
        change the recorded E_i to zero;
     end
    else
     begin
        queue task T_i at position j_p;
        for k = j_p + 1, length(Q)
          begin
            if current_time + Σ_{ℓ=1}^{k-1} E_ℓ ≥ ℓ_k then
             begin
                receiver_node := table_lookup(Q:observation, ℓ_k:laxity);
                dequeue and transfer T_k to receiver_node;
                change E_k to 0, and modify {p_i(j)}; /* Section 5 */
             end
          end
        if current_CET crosses TH_2k, 1 ≤ k ≤ ⌈K/2⌉ − 1, then
            /* region-change broadcasts: TH_1, ..., TH_{K-1} are thresholds */
            broadcast (1) time-stamped CET_n(ℓ)'s, ℓ ∈ [0, L_max], and (2) λ_n, {p_n(j)},
            and {p̂_n(k)} to all the other nodes in its buddy set;
     end
    (λ_n, {p_n(j)}, {p̂_n(k)}) = parameter_update(E_i, ℓ_i, t_i:interarrival_time);
     /* Section 5 */

When a broadcast message arrives from node i, 1 ≤ i ≤ N:
    update observation of node i's state, O_i(ℓ), ℓ ∈ [0, L_max];
    record (O_i(ℓ), CET_i(ℓ)), ℓ ∈ [0, L_max] pair needed for
        constructing probability distributions;
    record λ_i, {p_i(j)}, and {p̂_i(k)};

At every clock tick:
    current_CET := current_CET − 1;
    if current_CET crosses TH_2k, 1 ≤ k ≤ ⌈K/2⌉ − 1 then
        /* region-change broadcasts*/
        broadcast (1) time-stamped CET_n(ℓ)'s, ℓ ∈ [0, L_max], and (2) λ_n, {p_n(j)}, and
        {p̂_n(k)} to all the other nodes in its buddy set;

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

†The task queue Q is ordered by task laxities.
‡table_lookup is where the proposed location policy takes effect. The theoretical base is discussed in Section 3 and 4.

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

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)[1] in a small set, called a *buddy set*, of nodes in its physical proximity. The list of LS decisions is updated periodically based on both Bayesian and queueing analyses as described below.

Bayesian Analysis: is used to minimize the probability of transferring an unguaranteed task $\mathcal{T}$ with laxity ℓ to an incapable node i given the observation at the time of locating the receiver of $\mathcal{T}$. The state information collected through state broadcasting/probing may become outdated due to the delays in collecting it. That is, a node's observation $O_i(\ell)$ may be different from $CET_i(\ell)$ at the time of making a LS

[1] If multiple tasks have to be transferred out (as a result of their guarantees being invalidated by the insertion of the newly arrived task), the observation about other nodes will be updated before making successive LS decisions. That is, if an unguaranteed task with laxity ℓ and execution time m is transferred to node i, then $CET_i(\ell)$ will be updated as $CET_i(\ell) + m$ before choosing candidate nodes for other tasks to be transferred.

decision. We countered this problem in [1], [8] by using buddy sets, time-stamped region-change broadcasts, and Bayesian decision analysis, all of which are summarized in Section III for completeness. Each node broadcasts a time-stamped message, informing all the nodes in its buddy set of a state-region change and all its on-line estimated parameters. Upon receiving a broadcast message from node i, each node in node i's buddy set updates its observation of node i, and records the statistical samples which will be used to construct/update the posterior distribution, $p_{C_\ell}(\cdot \mid O_i(\ell))$, with Bayesian analysis. Each node estimates node i's true state based on its (perhaps outdated) observation via this posterior distribution of $CET_i(\ell)$ given $O_i(\ell)$. That is, each node—instead of hastily believing its observation about node i, $O_i(\ell)$—estimates $CET_i(\ell)$ based on $O_i(\ell)$, and determines node i's LS capability via $p_{C_\ell}(\cdot \mid O_i(\ell))$. The sufficient condition for node i to be capable of guaranteeing a task with laxity ℓ is $CET_i(\ell) \leq \ell$, the probability of which can be calculated as:

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

Queueing Analysis: is used to minimize the probability of a task $\mathcal{T}$'s guarantee being deprived by subsequent tighter-laxity task arrivals during the period between the transfer (to node i) and the execution or the laxity of $\mathcal{T}$, whichever occurs first. We calculate this probability by:

$$P\big(\overline{V}_{i,\ell} \mid CET_i(\ell) \leq \ell\big) = \sum_{k=0}^{\ell} P\big(\overline{V}_{i,\ell} \mid CET_i(\ell) = k\big) \cdot p_{C_\ell}(k \mid O_i(\ell)),$$

where $p_{C_\ell}(k \mid O_i(\ell))$ is constructed in **G1**, and $P(\overline{V}_{i,\ell} \mid CET_i(\ell) = k)$ relates the effect of future tighter-laxity task arrivals to the guarantee of $\mathcal{T}$ with laxity ℓ. After $\mathcal{T}$ is transferred to node i, it has to wait for the execution of all the tasks which constitute $CET_i(\ell)$. Tag these tasks as "primary" tasks. During the execution of primary tasks, "secondary" tighter-laxity tasks may arrive, and have to be executed (or transferred out if they cannot be guaranteed by node i) before $\mathcal{T}$. Similarly, there may be more tighter-laxity task arrivals during the execution of "secondary" tighter-laxity tasks, and so on. Let X denote the total execution time contributed by the tighter-laxity tasks arrived at node i after the transfer of $\mathcal{T}$ but prior to the execution, or the laxity, of $\mathcal{T}$, whichever occurs first. $\mathcal{T}$ will be guaranteed by node i in the presence of future task arrivals if $X \leq \ell - CET_i(\ell)$. We will derive $P\big(\overline{V}_{i,\ell} \mid CET_i(\ell) = k\big)$ in Section IV using queueing analysis.

The parameters needed in calculating $P\big(\overline{V}_{i,\ell} \mid CET_i(\ell) \leq \ell\big)$ are the composite task arrival rate λ_i, the distribution of task-execution time $\{p_i(.)\}$, and the distribution of task laxity $\{\hat{p}_i(.)\}$ on node i. Since the system state changes dynamically with time, these parameters have to be measured/estimated on-line by node i, and *piggybacked* in region-change broadcast messages to node n. Each node i records the interarrival time, the execution time, and the laxity of each task upon its arrival, and applies Bayesian estimation to determine the composite task arrival rate and the distributions of task-execution time and task laxity. Bayesian parameter estimation will be discussed in Section V.

III. State Estimation With Outdated Information

We proposed in [1] a decentralized, dynamic LS algorithm that achieves **G1** by using the buddy sets, time-stamped region-change broadcasts, and Bayesian decision analysis. We summarize below the strategies used to achieve **G1** for completeness and will incorporate **G2** in the proposed algorithm in Section IV.

Buddy Sets: Each node communicates with, maintains the state information of, and transfers unguaranteed tasks to, the nodes in its buddy set *only*. The communication overheads resulting from broadcasts/task transfers are thus reduced.

Time-Stamped Region-Change Broadcasts: The K state regions defined by $K-1$ thresholds, $TH_1, TH_2, \cdots, TH_{K-1}$, are used to characterize the workload of each node. Each node i broadcasts a time-stamped message, informing all the other nodes in its buddy set of its state-region change and all its on-line estimated parameters, whenever its *CET* crosses TH_{2k} for some k, where $1 \leq k \leq \lceil \frac{K}{2} \rceil - 1$. The state information kept at each node is thus up-to-date as long as the broadcast delay is not significant. The reason for not broadcasting the change of state region whenever a node's load crosses an odd-numbered threshold is to reduce the network traffic resulting from region-change broadcasts. Moreover, the reason for not combining two adjacent state regions into one and then broadcasting the change of state region whenever a node's *CET* crosses any threshold is to include finer state information in each broadcast and thus construct more accurate posterior distributions.

Bayesian Decision Analysis: To achieve **G1** in the presence of nonnegligible communication delays, each node has to estimate node i's capability based on the (perhaps obsolete) observation $O_i(\ell)$ through the posterior distribution of $CET_i(\ell)$, $p_{C_\ell}(\cdot \mid O_i)$. $p_{C_\ell}(\cdot \mid O_i)$ is constructed/updated with Bayesian analysis as follows. Each time-stamped broadcast message contains two sets of information:

1) Node number i, $CET_i(\ell)$, and the time t_0 when this message was sent.
2) On-line estimated task characteristics: λ_i, $\{p_i(j), 0 \leq j \leq E_{\max}\}$, and $\{\hat{\mathbf{p}}_i(k), 1 \leq k \leq L_{\max}\}$.[2]

When the message broadcast by node i arrives at node n, node i's $CET_i(\ell)$ at t_0, can be recovered by node n. Node n can also trace back to find its observation about node i, $O_i(\ell)$, at time t_0. This observation $O_i(\ell)$ is what node n thought (observed) about node i when node i actually has $CET_i(\ell)$. $O_i(\ell)$'s along with $CET_i(\ell)$'s are used by node n to compute/update the posterior distribution, $p_{C_\ell}(\cdot \mid O_i(\ell))$, given the observation $O_i(\ell)$, once every T_p units of time. Any inconsistency between $CET_i(\ell)$ and node n's observation of $CET_i(\ell)$, $O_i(\ell)$, is captured by this probability distribution. Besides, $CET_i(\ell)$ sent by node i at time t_0 is transformed into

[2]This information is used to calculate the criterion for **G2**, which will be discussed in Section IV.

node n's new observation[3], $O_i(\ell)$, about node i at the time node n receives this message by the rule that $O_i(\ell) = k$ if $TH_k \leq CET_i(\ell) < TH_{k+1}$, $k \geq 0$, and $TH_0 \triangleq 0$.

The only effect of the region-change broadcast delay is that messages may not arrive at a node immediately after their broadcast and may thus become obsolete upon their arrival at other nodes. The correctness of all samples gathered is, however, not affected by the broadcast delay. The undesirable effects of the delay in broadcasting region—change messages/transferring tasks are thus eliminated by using these posterior distributions. Another advantage of using Bayesian analysis is that the resulting algorithm is very robust (as compared to the other algorithms reported in [8], [15], [16]) to the variation of tunable design parameters, such as the number and values of thresholds, $TH_1, \cdots, TH_{K-1}$, and the probability update interval T_p. See [1] for a detailed account.

IV. Consideration of Future Task Arrivals

In case of heterogeneous task arrivals (e.g., different task arrival rates, distributions of task laxity, or distributions of task execution time) under the MLFS scheduling policy, transferring an unguaranteed task of laxity ℓ to the node with the least *CET* may not be a good choice if that node happens to have a large composite task arrival rate or most tasks arrived at that node happen to have tighter laxities than the transferred-in task. Task collisions may thus occur and excessive task transfers may ensue. The main intent of **G2** is to alleviate this problem.

In this section, we will establish a theoretical basis for **G2**. The parameters needed for **G2** are λ_i, $\{\hat{p}_i(j), 1 \leq j \leq L_{\max}\}$, $\{p_i(j), 0 \leq j \leq E_{\max}\}$, and $p_{C_\ell}(\cdot \mid O_i(\ell))$, which we assume are all available at node n, in which there is an unguaranteed task $\mathcal{T}$ of laxity ℓ to transfer. In Section III, we discussed how (i) these on-line estimated parameters are broadcast, (ii) observations about node i, $O_i(\ell)$, $1 \leq \ell \leq E_{\max}$, are updated, and (iii) the posterior distribution of $CET_i(\ell)$, $p_{C_\ell}(\cdot \mid O_i(\ell))$, given the observation $O_i(\ell)$ is computed/updated. Estimation of λ_i, $\hat{p}_i(j)$'s, and $p_i(j)$'s will be discussed in Section V.

Recall that $G_{i,\ell}$ denotes the event that an unguaranteed task $\mathcal{T}$ with laxity ℓ is estimated to be guaranteed by node i in the presence of future task arrivals, and $\overline{V}_{i,\ell}$, the event that future tighter-laxity task arrivals at node i will not invalidate the guarantee of $\mathcal{T}$. So,

$$\begin{aligned} P(G_{i,\ell} \mid O_i(\ell)) &= \sum_{k=0}^{\infty} [\, P(CET_i(\ell) \leq \ell) \cdot P(\overline{V}_{i,\ell} \mid CET_i(\ell) = k) \,] \\ &\qquad \times p_{C_\ell}(k \mid O_i(\ell)) \\ &= \sum_{k=0}^{\ell} P(\overline{V}_{i,\ell} \mid CET_i(\ell) = k) \cdot p_{C_\ell}(k \mid O_i(\ell)). \end{aligned} \tag{4.1}$$

The key issue here is how to derive $P(\overline{V}_{i,\ell} \mid CET_i(\ell) = k)$. Given $CET_i(\ell) \leq \ell$ at the time (say, time 0) of locating the receiver of an unguaranteed task $\mathcal{T}$, the following two conditions may occur:

- **C1)** Tighter-laxity tasks may arrive after the arrival of $\mathcal{T}$ at node i, and have to be executed before $\mathcal{T}$ under the MLFS policy, thus increasing $CET_i(\ell)$.
- **C2)** The tasks constituting $CET_i(\ell)$ and/or some tighter-laxity tasks arrived later than $\mathcal{T}$ may get their existing guarantees invalidated due to the subsequent tighter-laxity task arrivals, and thus have to be transferred.

C2 violates the work conservation law commonly assumed in queueing analysis. To remedy this violation, we take into account the effect of **C2** on $CET_i(\ell)$ by adding $p_i(0)$ in the distribution of execution time, $\{p_i(j), 1 \leq j \leq E_{\max}\}$. That is, when collecting statistics for the execution time distribution, each node i considers and records those tasks arrived at node i but eventually transferred out of node i as having null execution time, i.e., $p_i(0)$ is the fraction of tasks arrived at node i that will eventually be transferred out. Moreover, each future tighter-laxity task arrived at node i is estimated[4] to contribute j units of execution time with probability $p_i(j)$, $0 \leq j \leq E_{\max}$, and must be executed before $\mathcal{T}$, where $j = 0$ represents the case when the guarantee of a task is deprived by subsequent tighter-laxity task arrivals. For example, in Fig. 2(a), the task $\mathcal{T}(5, 3)$ with execution time 3 and laxity 5 cannot be guaranteed upon its arrival, and is thus treated by node i (in collecting its statistics) as a task $\mathcal{T}^*(5, 0)$ with execution time 0 and laxity 5.

With this modification to $p_i(j)$'s, work conservation, which states that no tasks depart from node i before they are completely served, can be virtually retained in the subsequent derivation. Those tasks which have their guarantees invalidated by tighter-laxity task arrivals are viewed as receiving zero unit of service time before they are transferred out of node i. On the other hand, because of this modification, our analysis does not model exactly the original queueing system of interest. However, since one cannot exactly predict the order of future task arrivals and their attributes and thus cannot know precisely whether or not a task will be transferred out of the task queue, one has to resort to some statistical measure (e.g., $p_i(0)$) to take into account the effect of tasks being kicked out of the queue on calculating the distribution of X. Moreover, as our simulation results in Section VI-B indicate, the performance of the proposed LS algorithm does significantly improve (by almost an order of magnitude) in reducing P_{dyn} even with the approximate analysis. This is because the approximate distribution of X suffices to be used as an index of the likelihood of future tighter-laxity task arrivals at a node and its corresponding effect on the node's LS capability.

Now, we want to derive $P(\overline{V}_{i,\ell} \mid CET_i(\ell) = k)$ subject to **C1**. Recall that X represents the total execution time

[3] The reason for transforming $CET_i(\ell)$ into $O_i(\ell)$ is to reduce the size of the observation space.

[4] Since we cannot really know the particular laxity and service requirements of future task arrivals in a dynamically changing distributed system with LS, we resort to the statistical measures, $p_i(j)$'s and $\hat{p}_i(j)$'s, based on the data gathered/estimated from the past to represent the attributes of future task arrivals on node i. $p_i(j)$'s reflect whether node i tends to receive long or short tasks; $\hat{p}_i(j)$'s reflect whether node i tends to receive tight or loose tasks.

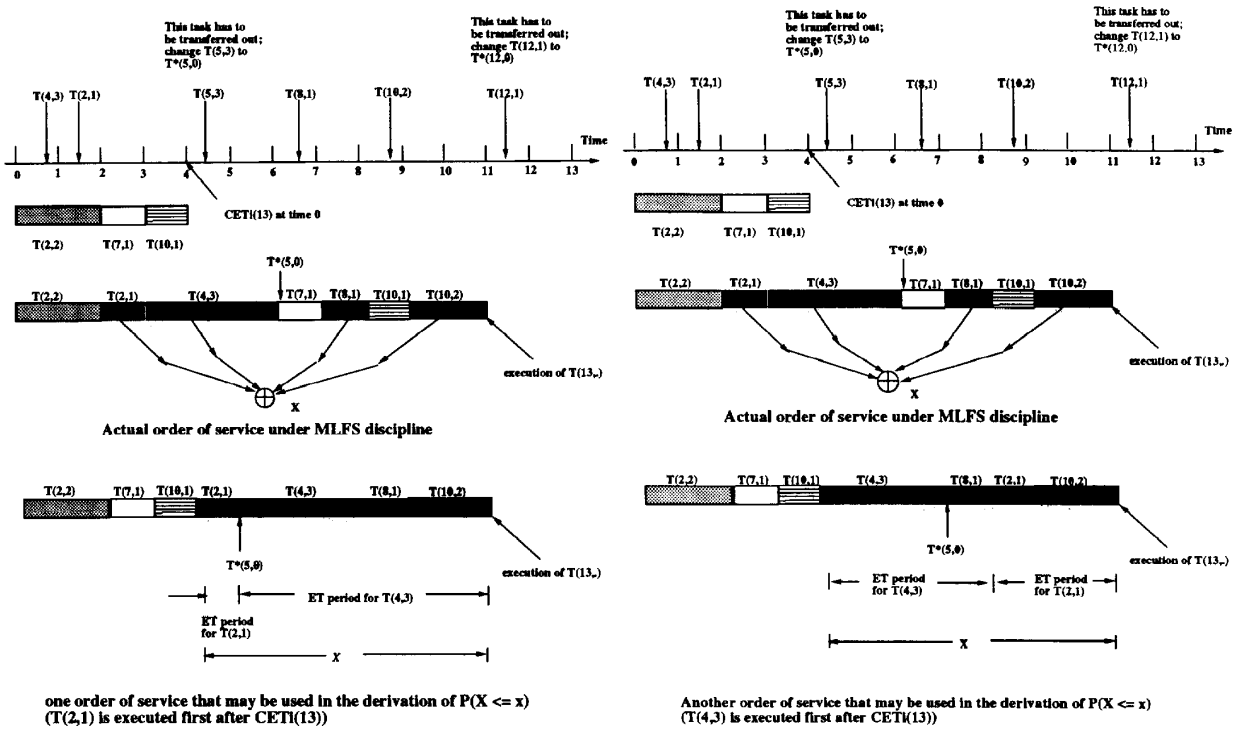


Fig. 2. (a) Future tighter-laxity task arrivals seen by a task $\mathcal{T}$ with $\ell = 13$. A task T with laxity x and execution time y is written as $T(x, y)$.This example shows 1) the independence of X from the execution order of tasks; 2) the definition and property of the ET period. (b) Future tighter-laxity task arrivals seen by a task $\mathcal{T}$ with $\ell = 13$.

contributed by tighter-laxity task arrivals at node i since time 0 or $\mathcal{T}$'s arrival but prior to $\mathcal{T}$'s execution (Fig. 2). If the distribution of X is known, then one can compute

$$P(\overline{V}_{i,\ell} | CET_i(\ell) = k) = P(X \leq \ell - k). \tag{4.2}$$

Since 1) node i stays busy (and cannot start execution of $\mathcal{T}$) as long as there are tasks with laxity $\leq \ell$ queued in front of it, and 2) the property of work conservation has been virtually retained with the way of collecting/estimating $p_i(j)$'s, e.g., no task will leave the system without getting serviced (but possibly with their null service time, the probability of which is estimated to be $p_i(0)$), we have virtually transformed the original system into a work-conserving one, and thus can use the well-known result of work-conserving systems [17], [18], which states the amount of work present in a work-conserving system does not depend on the service order of the customers. In our context, the amount of work seen by $\mathcal{T}$ prior to its execution, which is the sum of $CET_i(\ell)$ at time 0 (k in (4.2)) and X, is independent of the service order of the tasks constituting k and X. Fig. 2 shows two examples of this independence. Note that in Fig. 2 a task with laxity x and execution time y is expressed as $T(x, y)$; the black blocks represent the tighter-laxity tasks arrived after the arrival of $\mathcal{T}$ but prior to the execution of $\mathcal{T}$; the other blocks represent the tighter-laxity tasks queued before the arrival of $\mathcal{T}$.

With the observation that the amount of work, $CET_i(\ell)$ at time 0 (i.e., at time k in (4.2) plus X is independent of the execution order of tasks constituting k and X, we can permute the execution order of tasks with laxity $\leq \ell$ on node i so that those tasks contributing to $CET_i(\ell)$ may be executed first (Figs. 2 and 3). Then, we condition X on the number of tighter-laxity arrivals during $CET_i(\ell)$. Let $\mathcal{S}^{\langle 1 \rangle}$ denote the set of tighter-laxity tasks arrived during $CET_i(\ell)$. Each task $\mathcal{T}_m \in \mathcal{S}^{\langle 1 \rangle}$ will contribute to X with tighter-laxity task arrivals during its execution. Denote the set of these arrivals as $\mathcal{S}^{\langle 2 \rangle}_{\mathcal{T}_m}$. Furthermore, each task $\mathcal{T}_n \in \mathcal{S}^{\langle 2 \rangle}_{\mathcal{T}_m}$ will also contribute to X with subsequent tighter-laxity task arrivals during its execution, which are represented by $\mathcal{S}^{\langle 3 \rangle}_{\mathcal{T}_m, \mathcal{T}_n}$, and so on. This relation holds recursively for all m and n. We can thus view each tighter-laxity task arrived during $CET_i(\ell)$ as essentially generating its own execution time (ET) period. Examples of the ET period are shown in Figs. 2 and 3. Fig. 4 lists the corresponding $\mathcal{S}^{\langle i \rangle}_{\mathcal{T}_{m_1}, \mathcal{T}_{m_2}, \cdots, \mathcal{T}_{m_{i-1}}}$. By the Markovian property[5] of the task arrival process, all ET periods have the same distribution. We characterize the ET period by its cumulative density function (CDF), $G(t)$, $t \geq 0$, whose expression can be derived using the above recursive relation and will be discussed in the next subsection. Then, we have

$$P(X \leq x \mid CET_i(\ell) = k)$$

[5] Validity of this approximation about composite task arrivals at each node will be discussed in Section V.

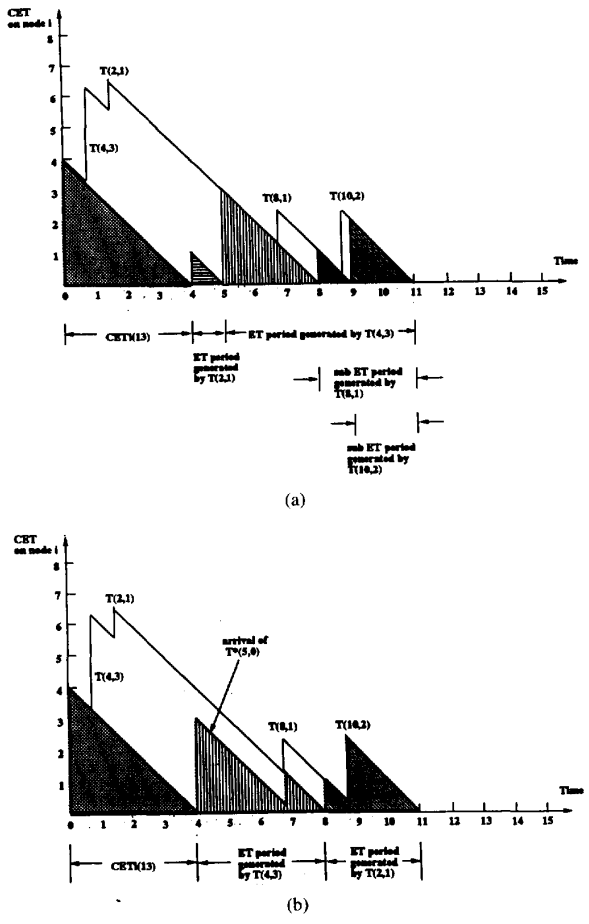


Fig. 3. ET periods for Fig. 2. Y-axis indicates the *CET* contributed by those tasks with laxity ≤ 13 on node i prior to the execution of task $\mathcal{T}$.

$$= \sum_{j=0}^{\infty} \left\{ \frac{(\lambda_h k)^j e^{-\lambda_h k}}{j!} \cdot G^{(j)}(x) \right\}, \quad (4.3)$$

where $G^{(j)}(x)$ is the CDF of the j-fold convolution of $G'(x) \triangleq \frac{dG(x)}{dx}$, and $\lambda_h = \sum_{n=1}^{\ell-1} \lambda \hat{p}_i(n)$ is the arrival rate of tasks with laxity $\leq \ell - 1$. The first factor inside the braces is the probability of j task arrivals with laxity $< \ell$ within $CET_i(\ell) = k$, and the second factor, $G^{(j)}(x)$, is the probability distribution of the sum of j ET periods.

A. Expression for $G(t)$

$G(t)$ is the CDF of the ET period generated by a tighter-laxity task arrived during $CET_i(\ell)$. By work conservation and the fact that a nonzero ET period continues to exist as long as there are unfinished tighter-laxity tasks arrived after the beginning of the ET period (e.g., $\mathcal{S}^{\langle i \rangle}_{\mathcal{T}_{m_1}, \mathcal{T}_{m_2}, \cdots, \mathcal{T}_{m_{i-1}}}$), $G(t)$ has the same distribution as the well-known busy period of an $M/G/1$ queue. (A *busy period* is defined to begin with the service (or equivalently, arrival) of a customer at an idle node and end when the node becomes idle again.) So, $G(t)$ can be

$\mathcal{S}^{<1>} = \{\mathcal{T}(4,3), \mathcal{T}(2,1)\}$,
$\mathcal{S}^{<2>}_{\mathcal{T}(2,1)} = \{\mathcal{T}^*(5,0)\}$, $\mathcal{S}^{<3>}_{\mathcal{T}(2,1),\mathcal{T}^*(5,0)} = \phi$,
$\mathcal{S}^{<2>}_{\mathcal{T}(4,3)} = \{\mathcal{T}(8,1)\}$, $\mathcal{S}^{<3>}_{\mathcal{T}(4,3),\mathcal{T}(8,1)} = \{\mathcal{T}(10,2)\}$,
$\mathcal{S}^{<4>}_{\mathcal{T}(4,3),\mathcal{T}(8,1),\mathcal{T}(10,2)} = \phi$.

(a) $\mathcal{S}^{<i>}_{\mathcal{T}_{m_1}, \mathcal{T}_{m_2}, \cdots, \mathcal{T}_{m_{i-1}}}$ corresponding to the order of service in Fig. 2(a).

$\mathcal{S}^{<1>} = \{\mathcal{T}(4,3), \mathcal{T}(2,1)\}$,
$\mathcal{S}^{<2>}_{\mathcal{T}(4,3)} = \{\mathcal{T}^*(5,0), \mathcal{T}(8,1)\}$, $\mathcal{S}^{<3>}_{\mathcal{T}(4,3),\mathcal{T}^*(5,0)} = \phi = \mathcal{S}^{<3>}_{\mathcal{T}(4,3),\mathcal{T}(8,1)}$,
$\mathcal{S}^{<2>}_{\mathcal{T}(2,1)} = \{\mathcal{T}(10,2)\}$, $\mathcal{S}^{<3>}_{\mathcal{T}(2,1),\mathcal{T}(10,2)} = \phi$.

(b) $\mathcal{S}^{<i>}_{\mathcal{T}_{m_1}, \mathcal{T}_{m_2}, \cdots, \mathcal{T}_{m_{i-1}}}$ corresponding to the order of service in Fig. 2(b).

Fig. 4. $\mathcal{S}^{\langle i \rangle}_{\mathcal{T}_{m_1}, \mathcal{T}_{m_2}, \cdots, \mathcal{T}_{m_{i-1}}}$ corresponding to the order of service in Fig. 3.

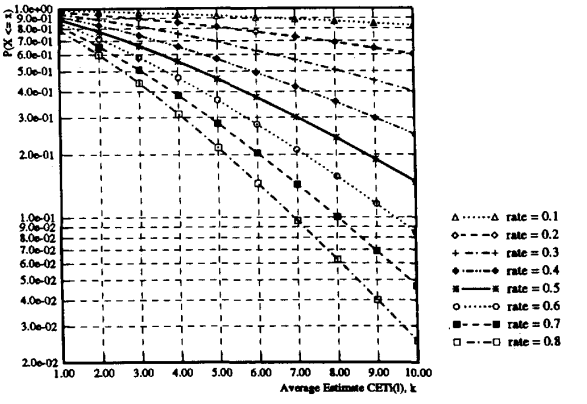


Fig. 5. $P(X \leq 2 \mid CET_i(\ell) = k)$ for different values of k. $\lambda_h = 0.8\lambda$. Task execution time is exponentially distributed with 1.0.

readily expressed as [19]:

$$G(t) = \sum_{n=1}^{\infty} \int_0^t e^{-\lambda_h t} \frac{(\lambda_h t)^{n-1}}{n!} b^{(n)}(t)\, dt, \quad (4.4)$$

where $b^{(n)}(t)$ is the n-fold convolution of $b(t)$, and $b(t)$ is the probability density function (PDF) of task-execution time expressed as a continuous-time function, i.e.,

$$b(t) = \sum_{j=0}^{E_{\max}} p_i(j) \cdot \delta(t - j);$$

here $\delta(.)$ is the impulse function.

Equation (4.4) is an explicit expression of $G(t)$ in terms of known (or on-line estimated) quantities. The problem that arises in the infinite summation of both (4.4) and (4.3) can be solved by properly truncating high-order terms, and the error thus induced can be bounded by some predetermined value.

Figs. 5 and 6 give two numerical examples of $P(X \leq x \mid CET_i(\ell) = k)$, the former fixes x at 2 and the latter fixes k at 3. As expected, $P(X \leq x \mid CET_i(\ell) = k)$ decreases for a fixed x, as the composite arrival rate of tighter-laxity tasks (λ_h) and/or the *CET* contributed by the queued tighter-laxity tasks (k) increases. Also, $P(X \leq x \mid CET_i(\ell) = k)$ increases with an increase in x (or a decrease in λ_h) for a fixed k. Each node n can (1) use Eqs. (4.1)–(4.3) to compute the probability that

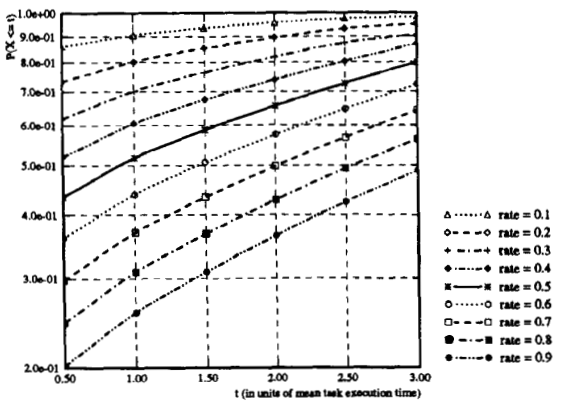


Fig. 6. Conditional probability distribution of X given the estimated *CET* at node i is $k = 3$. $\lambda_h = 0.8\lambda$. Task execution time is exponentially distributed with 1.0.

an unguaranteed task $\mathcal{T}$ with laxity ℓ is guaranteed by node i with consideration of node i's future task arrivals, based on node n's observation about node i, $O_i(\ell)$, and (2) choose the node i with the largest $P(G_{i,\ell} \mid O_i(\ell))$ when transferring $\mathcal{T}$.

V. Parameter Estimation

One key issue in applying the proposed adaptive LS algorithm is the on-line estimation of the composite task arrival rate, λ_i, the distribution of composite task laxity, $\{\hat{p}_i(j)\}$, and the distribution of composite task execution time, $\{p_i(j)\}$. All on-line estimated parameters will then be conveyed to other nodes via region-change broadcasts. 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 the local (external) task arrivals and transferred-in task arrivals, the latter of which is itself a composite process of local and transferred-in tasks from different nodes (see Fig. 7). One difficulty in estimating the composite task arrival rate is that the transferred-in task arrival process (and thus the composite arrival process) may not be Poisson even if the local 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, and ($R2$) 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 system state, which calls for the on-line estimation of the composite arrival rate.

Bayesian estimation is used for the on-line computation of the composite task arrival rate on a node. We consider the case of Poisson external task arrivals.[6] We further *approximate* the composite task arrival process to be Poisson (in spite of $R1$ and $R2$). This approximation rests on a general result of renewal theory which states that the superposition of increasingly many component processes (i.e., a reasonably large number of nodes) yields (in the limit) a Poisson process. We also ran simulations, collected task interarrival times on-line under the proposed LS algorithm, and used the Kolmogorov–Smirnov test to verify whether or not the Poisson approximation is valid. The simulation results show that for a light to medium loaded system of size ≥ 12, this approximation holds. More on this will be discussed in Section V.

Bayesian estimation works as follows [20]. Each node:

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

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

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

To make the above 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_S and any values of the observed interarrival 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 [20] 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

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

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

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

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

[6] We will later in the simulation consider the case of hyperexponential task interarrival times which represents a system potentially with bursty task arrivals, and investigate to what extent the proposed algorithm remains effective.

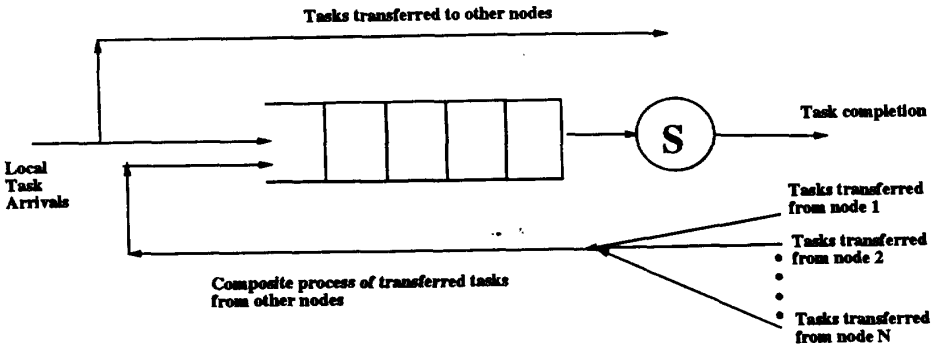


Fig. 7. A generic queueing model for each node.

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 latest N_S interarrival-time samples 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 proposed LS algorithm are $\{\hat{p}_i(j)\}$, and $\{p_i(j)\}$. The estimation techniques used to determine $\{\hat{p}_i(j)\}$ and $\{p_i(j)\}$ are virtually the same; 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 ($1 \le j \le L_{\max}$), where $\sum_{j=1}^{L_{\max}} \hat{p}_i(j) = 1$. Suppose N_S independent experiment outcomes are available. Let $\mathbf{Y} = (Y_1, \cdots, Y_{L_{\max}})$, 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 $\mathbf{p} = (p_i(1), p_i(2), \cdots, p_i(L_{\max}))$, i.e., see (5.3) at the bottom of the page. The conjugate family of distributions for the parameter $\mathbf{p}_i$ with a multinomial likelihood function is the Dirichlet distribution with parametric vector $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_{L_{\max}})$, i.e., see (5.4) at the bottom of the page, where $\alpha_0 = \sum_{i=1}^{L_{\max}} \alpha_i$. Specifically, each node assumes the noninformative distribution as the prior distribution of $\mathbf{p}$, e.g., the prior distribution of $\mathbf{p}$ is the Dirichlet distribution with $\alpha_j = 1, 1 \le j \le L_{\max}$. 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 $\mathbf{p}$ is updated as

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

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

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

Thus, 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 performance of the proposed LS algorithm is evaluated according to the following sequence:

- Validation of the Poisson approximation of the composite task arrival process which was made to facilitate the on-line estimation of task arrival rates.
- Discussion of the parameters considered/varied in performance evaluation of LS algorithms.
- Performance evaluation. We first discuss the performance metrics used and their significance. Second, we analyze the effect of considering **G2** on the performance of LS algorithms. Then, we comparatively evaluate 1) no load sharing, 2) the focused addressing algorithm [11], [12], 3) the proposed LS algorithm, and 4) perfect information LS. Finally, we study the impact of statistical fluctuation in task arrivals (by use of hyperexponential external task arrivals in the simulation) on the performance of the proposed algorithm.

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

$$\mathcal{D}(\mathbf{p} \mid \alpha) = \begin{cases} \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_{L_{\max}})} p_i(1)^{\alpha_1 - 1} \cdots p_i(L_{\max})^{\alpha_{L_{\max}} - 1}, & \mathbf{p} \in \mathbf{R}^{L_{\max}}, p_i(j) > 0, \text{ for } 1 \le j \\ & \le L_{\max}, \text{ and } \sum_{j=1}^{L_{\max}} p_i(j) = 1, \\ 0, & \text{otherwise,} \end{cases} \tag{5.4}$$

A. On the Poission Assumption of Composite Task Arrivals

The on-line estimation of the composite task arrival rate is done under the assumption that the composite task arrival process can be approximated to be Poisson.[7] This assumption 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, when the task transfer-out ratio is small, so is the "disturbance" to the (originally) Poisson 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 $\lambda = \frac{n}{\sum_{j=1}^{n} t_j}$, the Kolmogorov–Smirnov goodness-of-fit test is used to determine if $t_1, \cdots, t_n$ represent a random sample from an exponential distribution.

For completeness, we summarize below the steps of the Kolmogorov–Smirnov test used and discuss the data obtained from event-driven simulations. The interested readers are referred to [22] for a detailed account of the Kolmogorov–Smirnov test. We first run 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 $F_k(t)$ that is defined as the portion of the observed samples that 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

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

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., $F(t) \triangleq plim_{k\to\infty} F_k(t) = F_\lambda(t)$, where *plim* denotes "probabilistic limit";

H_1: H_0 is not true;

where $F_\lambda(t) = 1 - \exp(-\lambda t)$ is the hypothesized exponential distribution. The test statistic D for the Kolmogorov–Smirnov 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, there are large differences between $F(t)$ and $F_\lambda(t)$, and the null hypothesis is rejected. To judge whether or not D is large enough to justify rejecting H_0, we compare D with the critical value D^* [22] of the Kolmogorov–Smirnov test. For example, as the sample size $k > 40$, D^* can be calculated as $\frac{1.36}{\sqrt{k}}$ ($= 0.136$ in our case) at the significance level $\alpha = 0.05$, where α is the probability that H_0 is falsely rejected. If $D > D^*$, we reject H_0; otherwise, we accept H_0 at the significance level α.

TABLE I
VALIDATION OF THE POISSON ASSUPTION: IF $D < D^* = 0.136$, THEN THE ASSUMPTION IS VALID FOR THE SIGNIFICANCE LEVEL 0.05

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

It turns out that in the case of Poisson external task arrivals, we have for all combinations of task attributes[8], $D < D^* = 0.136$ (i.e., H_0 is accepted) 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 $\overline{\lambda_{ext}} = \frac{1}{N}\sum_{i=1}^{N} \lambda_i^{ext} \le 0.8$. See Table I for a typical numerical example. Since both of the above conditions are satisfied for the proposed LS algorithm, the approximation of exponential interarrival times is acceptable at the significance level $\alpha = 0.05$ 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 three groups.

1) *System parameters*, such as the number, N, of nodes in the system, the degree of system heterogeneity, and the communication delay which consists of task-transfer and medium-queueing delays. The former delay depends on the capacity of the communication network and the size of the transferred task, while the latter dynamically changes with the system load.
2) *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. For all results presented below, we use $\{e_1, e_2, \cdots, e_k\}_{\{q_{e_1}, q_{e_2}, \cdots, q_{e_k}\}}$ to denote the task set in which an external task requires execution time e_i with probability q_{e_i}, $\forall i$. If $q_{e_i} = q \ \forall e_i$, then $\{q_{e_1}, q_{e_2}, \cdots, q_{e_k}\}$ is condensed to q. Similarly, $\{\ell_1, \ell_2, \cdots, \ell_n\}_{\{\hat{q}_{\ell_1}, \hat{q}_{\ell_2}, \cdots, \hat{q}_{\ell_n}\}}$ is used to describe the laxity distribution of external tasks.

[7] The same assumption was also used in [8], [21] without any justification.

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

3) *Design parameters of the proposed LS algorithm*, such as the number, K, and values of thresholds, $TH_1, \cdots, TH_{K-1}$, and the posterior probability update interval T_p.

A 16-node ($N = 16$) regular[9] system is used in our simulations. The size of buddy set is chosen to be 12, because increasing it beyond 10 was shown to be ineffective [8]. The numerical experiments on the degree of system heterogeneity were conducted by dividing nodes into K_n groups; the nodes in each group g, $1 \leq g \leq K_n$, have an external task arrival rate λ_g^{ext} such that $\frac{\lambda_{g+1}^{\text{ext}}}{\lambda_g^{\text{ext}}} = r$ and $\frac{1}{N}\sum_{i=1}^{N} \lambda_i^{ext} = \overline{\lambda_{\text{ext}}}$. The performance of the proposed LS algorithm was simulated while varying K_n from 2 to 6, and r from 2 to 4. Region-change broadcasts 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-transfer delay is varied from 10% to 50% of the execution time of each task being transferred. The broadcast-message-transmission delay is assumed to be negligible. 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 candidate node. The physical transfer of a task may thus require tens of communication packets, while a region-change broadcast would in all likelihood need at most one packet. The medium-queueing delay which is experienced by *both* broadcast messages and transferred tasks and which dynamically changes with system load and traffic is modeled as a linear function of the number of tasks/messages queued for the particular medium.

The simulation was carried out for both exponential and hyperexponential task arrivals while varying the average external task arrival rate per node, $\overline{\lambda_{\text{ext}}}$, from 0.2 to 0.9, the ratio of $\frac{e_{j+1}}{e_j}$ $(1 \leq j \leq k-1)$ from 2 to 10, and the ratio of of $\frac{\ell_{j+1}}{\ell_j}$ $(1 \leq j \leq n-1)$ from 2 to 6. The case with hyperexponential interarrival times represents a system potentially with bursty task arrivals, and is used to study the impact of statistical fluctuation in task arrivals on the LS performance. The squared coefficient of variation of hyperexponential arrivals (CV^2) is varied from 1 to 64. For convenience, all time-related parameters are reported/expressed in units of average task execution time.

The design parameters, K, TH_k's, and T_p, may affect the accuracy of posterior distributions which were used for **G1** and **G2**. 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 [1] that determines the design parameters. For each task set and each system configuration, we fix all but one design parameter of interest at a time, and obtain performance figures as a function of this parameter from which its optimal value can then be determined. Next, we vary another parameter of interest while keeping the first parameter fixed at its "optimal" value and the rest of the parameters fixed at their originally chosen values. This process was repeated until all parameters are varied. Since a different order of examining parameters may lead to different parameter values, we choose the one which yields the smallest P_{dyn}.

Although the set of parameters obtained through the above method may not be globally optimal, our simulations have shown them to yield good results, as compared to other existing LS algorithms. Moreover, the proposed LS algorithm was shown to be robust to the variation of these parameters, an important advantage coming from 1) the use of Bayesian analysis and 2) the use of threshold values as reference points for broadcasting messages, rather than for transferring tasks.

We shall present only those results that 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 external task execution and laxity distributions and a given system configuration is valid over a wide range of combinations of external task execution time and laxity distributions.

C. Performance Evaluation

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

- The probability of dynamic failure, P_{dyn}: Since failure to complete a real-time task before its deadline could cause a disaster in real-time systems, this measure is the key performance metric for the evaluation of real-time systems.
- Maximum system utilization, $\overline{\lambda_{\text{ext}}}$: the system utilization is defined as the ratio of the external exponential task arrival rate ($\overline{\lambda_{\text{ext}}}$) to the system service rate, under the assumption that both task laxity and execution time are exponentially distributed. The service rate is normalized to 1 in our numerical results, and thus the system utilization simply becomes $\overline{\lambda_{\text{ext}}}$. Since P_{dyn} increases with system load, there exists an upper bound or $\overline{\lambda_{\text{ext}}}$, termed as *maximum system utilization* $\overline{\lambda_{\text{max}}}$, below which $P_{\text{dyn}} \leq \epsilon$ can be guaranteed for some prespecified $\epsilon > 0$. This is the highest frequency of task activations allowed for a specified P_{dyn}.
- The task transfer-out ratio, γ: is defined as the ratio of transferred tasks to composite (both local and transferred) tasks. This is a measure of traffic overhead due to task transfers.
- The frequency of task collision, f_{tc}: is defined as the fraction of transferred tasks that are not guaranteed (under the MLFS scheduling policy) on remote nodes after their transfer. This is a measure of the capability of the LS algorithms in reducing the possibility of task re-transfers.

Effects of **G2** *on the Performance of LS Algorithms:* We now analyze the performance improvement achievable by considering **G2**, and compare the performance of the proposed LS policy with others using trace-driven simulations. For each combination of task set and system configuration, the simulation ran until it reached a 95% level of confidence in the results for a maximum error of 1) 2% of the specified

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

TABLE II
f_{tc} OF THE PROPOSED APPROACH WITH AND WITHOUT **G2** FOR A TASK SET $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, AND $L = \{1, 2, 3\}_{1/3}$

$\overline{\lambda_{ext}}$	$(K_n, r) = (2,2)$		$(K_n, r) = (4,3)$		$(K_n, r) = (8,2)$	
	with **G2**	without **G2**	with **G2**	without **G2**	with **G2**	without **G2**
0.2	0.018	0.021	0.019	0.027	0.021	0.034
0.4	0.047	0.056	0.050	0.072	0.062	0.087
0.6	0.074	0.118	0.078	0.132	0.097	0.146
0.8	0.176	0.224	0.189	0.247	0.201	0.269

The task transfer delay is assumed to be 10% of task execution time.

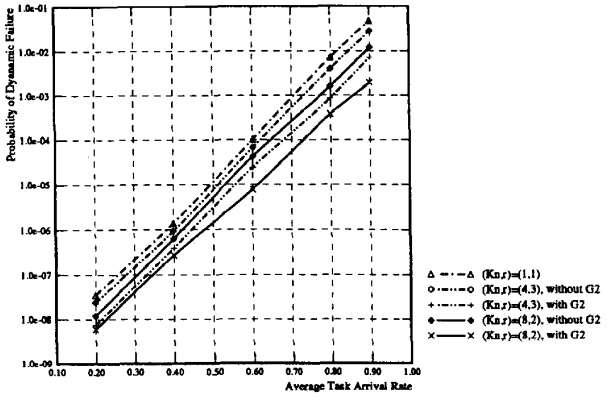


Fig. 8. $P_{\rm dyn}$ of the proposed LS approach with and without **G2** for a task set with $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, and $L = \{1, 2, 3\}_{1/3}$. The task transfer delay is assumed to be 10% of task execution time.

probability if $P_{\rm dyn}$ is the measure of interest, and 2) 5% of the ratio or frequency if the task transfer-out ratio or the frequency of task collision is the measure. 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 to be calculated is normally-distributed with unknown mean and variance.

Fig. 8 plots the performance of the proposed LS policy *with* and *without* consideration of **G2** for different degrees of system heterogeneity. When $\overline{\lambda_{\rm ext}} \geq 0.4$ and as the degree of system heterogeneity increases, one can make a substantial performance gain with **G2**. In other words, inclusion of **G2** in a LS algorithm avoids the possibility of transferring tasks to those nodes which tend to become overloaded or receive tighter-laxity tasks. This, in turn, reduces the possibility of task collisions and task re-transfers (and thus $P_{\rm dyn}$). See Table II for numerical examples of $f_{\rm tc}$ for the proposed algorithm.

Taking **G2** into account is not restricted to the proposed LS algorithm; it can also be incorporated into other existing LS algorithms. For example, a parallel state-probing approach can be modified to reduce $P_{\rm dyn}$ as follows. Each node collects and estimates λ_i, $\{p_i(j), 0 \leq j \leq E_{\max}\}$, and $\{\hat{p}_i(k), 1 \leq k \leq L_{\max}\}$ on-line as discussed in Section V. A node with an unguaranteed task probes a predetermined number[10] of nodes in parallel. A probed node i sends the probing node, in addition to its *CET*, the estimated λ_i, $p_i(j)$'s, and $\hat{p}_i(k)$'s. After receiving this information, the probing node considers **G2** and chooses the most capable receiver. Fig. 9 depicts

[10]This has been set to 5 in our simulations based on the finding in [5].

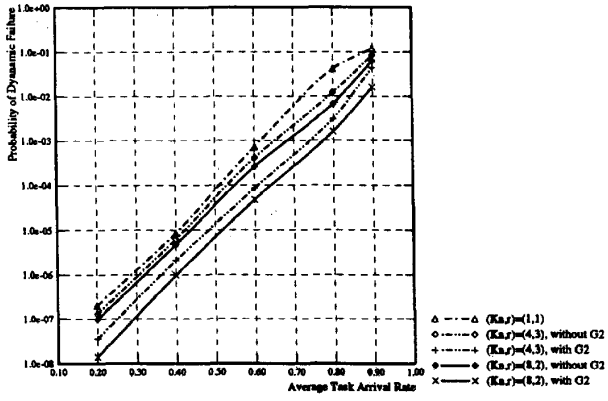


Fig. 9. $P_{\rm dyn}$ of the parallel state probing with and without **G2** for a task set with $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, and $L = \{1, 2, 3\}_{1/3}$. The task transfer delay is assumed to be 10% of task execution time.

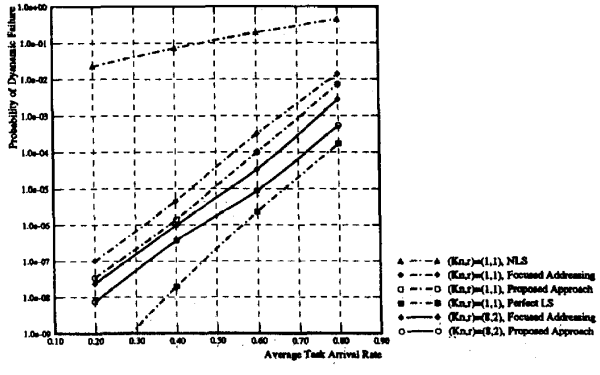


Fig. 10. Performance comparison w.r.t. $P_{\rm dyn}$ among different LS approaches for a 16-node system with a task set $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{1/3}$. The task transfer delay is assumed to be 10% of task execution time.

the performance of the parallel state-probing approach with and without consideration of **G2**. Again, the performance improvement made by **G2** becomes substantial as the degree of system heterogeneity increases.

Performance Comparison Among DifferentLS Algorithms: The proposed LS algorithm is comparatively evaluated against a simplified version of the focused addressing approach in [11], [12]. It differs slightly from that of [11], [12] in the way a node chooses the focused node. The authors of [11], [12] used the percentage of free time during the next window (which is a design parameter) and many other parameters—that need to be estimated/tuned—to determine the focused node or the node to which the task must be transferred again. By contrast, we use the observed *CET* of other nodes to determine the receiver of each unguaranteed task.

We also compare the proposed LS algorithm with two baseline schemes. The first baseline scheme assumes no load sharing, while the second is an ideal scheme where each node has complete information on all the other nodes without any overhead/delay in collecting it.

Figs. 10–12 show the performance curves of different LS algorithms for different task attributes. Three task sets are con-

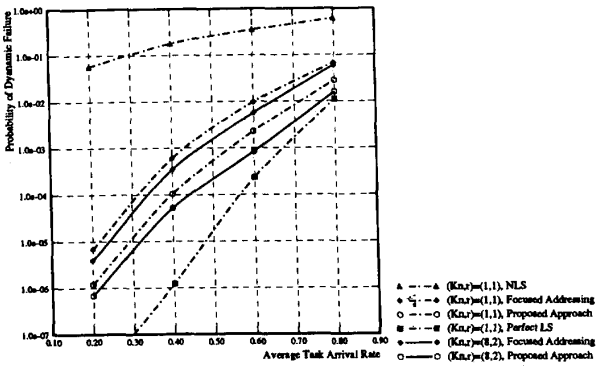


Fig. 11. Performance comparison (w.r.t. $P_{\rm dyn}$) among different LS approaches for a 16-node system with a task set $ET = \{0.027, 0.27, 2.7\}_{1/3}$, $L = \{1, 2, 3\}_{1/3}$. The task transfer delay is assumed to be 10% of task execution time.

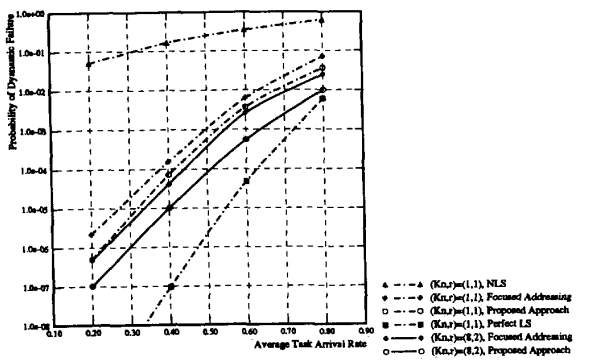


Fig. 12. Performance comparison (w.r.t. $P_{\rm dyn}$) of different LS approaches for a 16–node system with a task set $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1\}_1$. The task transfer delay is assumed to be 10% of task execution time.

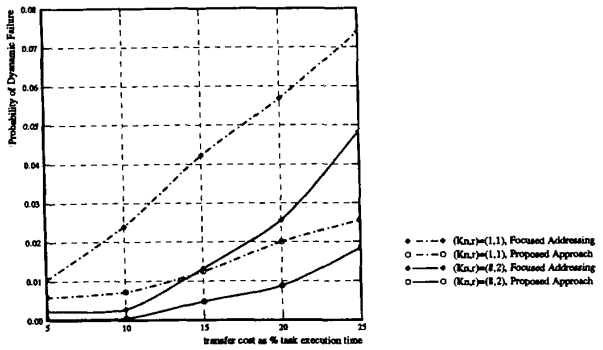


Fig. 13. Effect of task transfer delay on $P_{\rm dyn}$ for the proposed approach and the focused addressing approach in a 16-node system with $\overline{\lambda_{\rm ext}} = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{1/3}$.

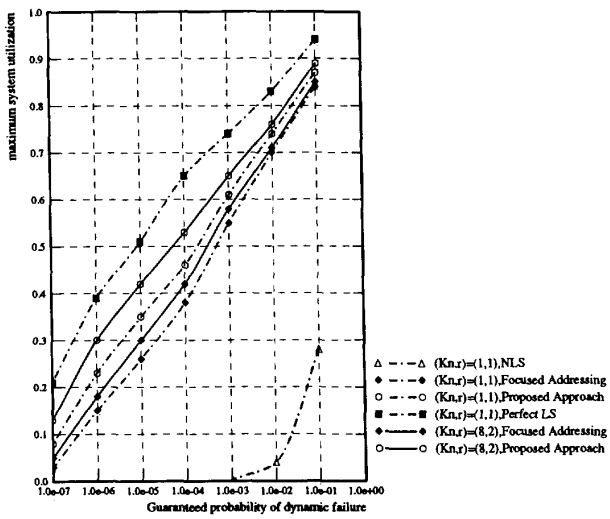


Fig. 14. $\overline{\lambda_{\rm ext}}$ vs. ϵ. External task arrivals are Poisson. Both task execution time and laxity are exponentially distributed.

sidered: (I) $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{1/3}$; (II) $ET = \{0.027, 0.27, 2.7\}_{1/3}$, $L = \{1, 2, 3\}_{1/3}$; and (III) $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1\}_1$. The average external task arrival rate $\overline{\lambda_{\rm ext}}$ is varied from 0.2 to 0.9. Fig. 13 shows the effect of task-transfer delay on the performance of different LS algorithms. For clarity, only the performance curves corresponding to $(K_n, r) = (1, 1)$ are shown for no LS and perfect LS. From these curves, one can observe that the proposed LS algorithm outperforms the focused addressing approach in minimizing $P_{\rm dyn}$, especially when 1) the distribution of external task laxity is tight, 2) the spectrum of task execution time is wide, 3) the degree of system heterogeneity is large, and 4) the task-transfer delay is significant. The superiority of the proposed LS policy under condition 1) comes from the fact that an unguaranteed task with tight laxity cannot tolerate the possibility of being transferred to an incapable node or a node which will become incapable in near future. (Note that the proposed LS algorithm deliberately eliminates such a possibility.) Under condition 2), a node easily becomes incapable with the arrival of even a single task which has a tight laxity and requires a large execution time. This makes the consideration of future task arrivals crucial in locating the receiver of each unguaranteed task. The performance improvement under 3) and 4) results from the consideration of **G2** and **G1**, respectively.

Fig. 14 shows the plot of maximum system utilization $\overline{\lambda_{\rm ext}}$ versus ϵ. This relates the worse-case achievable $P_{\rm dyn}$ to the frequency of task activations. One important result from these curves is that with the clever use/interpretation of state information/statistical samples, we do not have to sacrifice $\overline{\lambda_{\rm ext}}$ for lower $P_{\rm dyn}$, which is in contrast to the common notion of trading system utilization for real-time performance. Moreover, the proposed algorithm outperforms the focused addressing algorithm, and the performance superiority becomes more visible as the degree of system heterogeneity increases.

Table III gives numerical results of task transfer-out ratio γ for different LS algorithms. From this table, we observe that γ for the proposed LS approach is smaller than that for the focused addressing. This indicates the ability of the proposed LS algorithm in avoiding task transfers to 1) incapable nodes as a result of using out-of-date state information, and 2)

TABLE III
PERFORMANCE COMPARISON (W.R.T. TASK TRANSFER-OUT RATIO) OF DIFFERENT LS APPROACHES FOR A 16-NODE SYSTEM $\overline{\lambda_{\text{ext}}} = 0.8$

LS algorithm	Focused Addressing		Proposed Algorithm		Perfect LS	
(K_n, r)	(1,1)	(8,2)	(1,1)	(8,2)	(1,1)	(8,2)
Task set I	0.241	0.278	0.206	0.237	0.184	0.213
Task set II	0.398	0.416	0.304	0.311	0.276	0.294
Task set III	0.355	0.392	0.321	0.347	0.286	0.312

The task transfer delay is assumed to be 10% of task execution time.

capable nodes that may easily become incapable as a result of future task arrivals. The performance improvement becomes more pronounced as the tightness of task laxity distribution and/or the degree of system heterogeneity increases.

Effect of Statistical Fluctuation in Task Arrivals on the Proposed Scheme: One issue in using a Bayesian estimation model is to what extent the proposed LS scheme remains effective when the attributes of tasks arrived at a node randomly fluctuate. We study this effect on the estimation of composite task arrival rates by simulating different task sets with hyperexponential external task interarrival times. This represents a system potentially with bursty task arrivals, and the degree of statistical fluctuation over a short period is modeled well by varying the coefficient of variation (CV) of the hyperexponential external task interarrival times.[11] Fig. 15 shows the simulation results under heavy system load ($\overline{\lambda_{\text{ext}}} = 0.8$, where the LS performance is most sensitive to the variation of CV) with the window of the sample size $N_S = 30$. From Fig. 15, we draw the following conclusions: as the variance of task interarrival times (CV) becomes greater, the sample-mean based estimate gets worse. This is because the variability effect due to task burstiness cannot be totally smoothed out. This accounts, in part, for the performance degradation of the proposed algorithm. Another reason for performance degradation is due to the capacity limit of the distributed system; that is, the system inherently cannot guarantee simultaneously bursty time-constrained task arrivals. The proposed scheme remains effective (despite its gradual degradation) up to CV = 7.42 (or $\text{CV}^2 = 55$) beyond which it reduces essentially to the scheme without the use of Bayesian estimation. This suggests that within a wide range of statistical fluctuation in task arrival patterns, parameter estimates based on the Bayesian technique suffice to serve as an index of the tendency of future task arrivals on a node.

VII. CONCLUSION

We proposed a new LS algorithm for real-time applications by considering the effects of future task arrivals on locating the best receiver for each unguaranteed task. The proposed LS algorithm minimizes not only the probability of transferring an unguaranteed task $\mathcal{T}$ to an incapable node, but also the probability of the chosen remote node failing to guarantee $\mathcal{T}$ because of the node's future arrivals of tighter-laxity tasks. Consideration of future task arrivals significantly improves the performance of LS (in minimizing P_{dyn}) when system workloads are unevenly distributed among nodes.

All parameters needed in the LS decision process—observation/estimation of other nodes' states, composite task arrival rates of other nodes, and task execution time and task laxity distributions of other nodes—are collected/estimated on-line using time-stamped region-change broadcasts and Bayesian estimation theory. This makes the proposed algorithm 1) less sensitive to communication delays and 2) adaptive to dynamically varying workloads with little computational overhead.

The Poisson approximation of composite task arrivals—which has been used without justification in other LS algorithms (e.g., [8], [21])—has been carefully checked by the Kolmogorov–Smirnov goodness-of-fit test. Our simulation results have indicated that this assumption holds for a system with a reasonably large ($\geq$12) number of nodes and/or with a small ($\leq$0.25) average task transfer-out ratio. The negative impact of statistical fluctuation in task arrivals on the proposed approach with use of Bayesian estimation is also shown to be acceptably low over a wide range of bursty task arrivals (e.g., up to $\text{CV}^2 = 55$, where CV is the coefficient of variation of external task interarrival times used in the simulation).

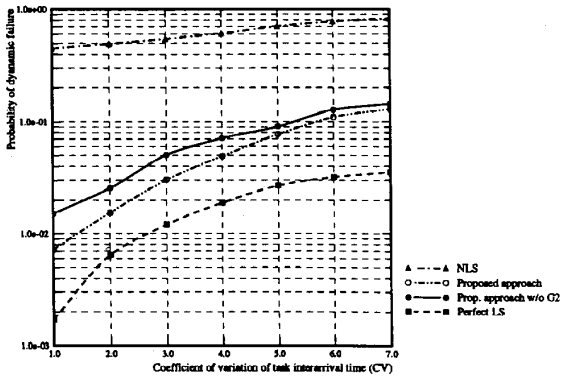


Fig. 15. P_{dyn} vs. coefficient of variation (CV) of external task interarrival times for a 16-node homogeneous ($(K_n, r) = (1,1)$) system with $\overline{\lambda_{\text{ext}}} = 0.8$, $ET = \{0.4, 0.8, 1.2, 1.6\}_{0.25}$, $L = \{1, 2, 3\}_{1/3}$.

APPENDIX
LIST OF SYMBOLS

$\{p_i(j), j = 1, \cdots, E_{\max}\}$: the distribution of composite[12] task execution time on node i, where $E_{\max}$ is the maximum task execution time. This distribution will be estimated on-line by each node i.

$\{\hat{p}_i(j), j = 1, \cdots, L_{\max}\}$: the distribution of composite task laxity on node i, where $L_{\max}$ is the maximum laxity. This distribution will also be estimated on-line by each node i.

$CET_i(\ell)$: the cumulative execution time (*CET*) on node i contributed by tasks with laxity $\leq \ell$ under the MLFS discipline.

$O_i(\ell)$: the observation about $CET_i(\ell)$ made by some node $j \neq i$.

[11] Let T_l be the task interarrival time. By Chebyshev's inequality, $P(\mid T_l - E(T_l) \mid \geq nE(T_l)) \leq \frac{\text{CV}^2}{n^2}$; i.e., the smaller CV^2, the less likely T_l will deviate from its mean.

[12] both external and transferred

$p_{C_i}(\cdot \mid O_i(\ell))$: the posterior distribution of $CET_i(\ell)$ given the observation $O_i(\ell)$. This posterior distribution is constructed by each node $j \neq i$ with the state samples collected via time-stamped region-change broadcasts.

$\overline{V}_{i,\ell}$: the event that future tighter-laxity task arrivals at node i do not invalidate the existing guarantee of a task with laxity ℓ.

$G_{i,\ell}$: the event that a task with laxity ℓ can be guaranteed by node i even in the presence of future tighter-laxity task arrivals.

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

$g_1(\lambda_i) = \text{const}$: the noninformative distribution which serves as the prior distribution in Bayesian estimation.

$f(t \mid \lambda_i) = \lambda_i e^{-\lambda_i t}$: 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 α and β.

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

$Y = (Y_1, \cdots, Y_{L_{\max}})$: 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.

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

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

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

$F_k(t)$: 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_{ks}: 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.

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

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

K_n: the number of heterogeneous groups in the system.

r_n: the ratio of external task arrival rates between two "adjacent" groups, i.e., $\frac{\lambda_{g+1}^{\text{ext}}}{\lambda_g^{\text{ext}}} = r_n$.

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

REFERENCES

[1] K. G. Shin and C.-J. Hou, "Design and evaluation of effective load sharing in distributed real-time systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 7, pp. 704–719, July 1994.

[2] C. M. Krishna and K. G. Shin, "Performance measures for multiprocessor controllers," *Performance '83*, A. K. Agrawala and S. K. Tripathi, Eds. Amsterdam: North-Holland, 1983, pp. 229–250.

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

[4] 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.

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

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

[7] ——, "An application of Bayesian decision theory to decentralized control of job scheduling," *IEEE Trans. Comput.*, vol. C-34, no. 2, pp. 117–130, Feb. 1985.

[8] 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.

[9] ——, "Load sharing in hypercube multicomputers for real-time applications," presented at the *4th Conf. Hypercube, Concurrent Comput., and Applicat.*, 1989.

[10] J. F. Kurose and R. Chipalkatti, "Load sharing in soft real-time distributed computer systems," *IEEE Trans. Comput.*, vol. 36, no. 8, pp. 993–999, Aug. 1987.

[11] 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.

[12] 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. 1110–1141, Aug. 1989.

[13] T. P. Yum and M. Schwartz, "The join-biased-queue rule and its application to routing in computer communication networks," *IEEE Trans. Commun.*, vol. COM-29, no. 4, pp. 505–511, Apr. 1981.

[14] T. P. Yum and H.-C. Lin, "Adaptive load balancing for parallel queues with traffic constraints," *IEEE Trans. Commun.*, vol. COM-32, no. 12, pp. 1339–1342, Dec. 1984.

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

[16] S. Pulidas, D. Towsley, and J. A. Stankovic, "Embedding gradient estimators in load balancing algorithms," in *IEEE Proc. 8th Int. Conf. Distrib. Comput. Syst.*, 1988, pp. 482–490.

[17] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York: John Wiley, 1975.

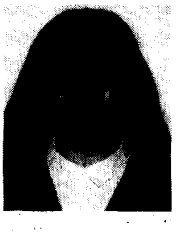
[18] D. Gross and C. Harris, *Fundamentals of Queueing Theory*, second ed. New York: John Wiley, 1985.

[19] S. M. Ross, *Stochastic Processes*. New York: John Wiley, 1983.

[20] M. H. DeGroot, *"Optimal Statistical Decisions*. New York: Mcgraw-Hill, 1970.

[21] 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.

[22] M. H. DeGroot, *"Probability and Statistics*, second ed. Reading, MA: Addison-Wesley, 1986.



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, real-time communications, queueing systems, estimation and decision theory, and performance modeling/evaluation. She is the recipient of Woman in Science Initiative Award, Chancellor's Faculty Enhancement Fund, from the University of Wisconsin-Madison, 1993–1995. 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) for photograph and biography, please see p. 1025 in this issue of this TRANSACTIONS.