Dynamic Time Delay Models for Load Balancing Part II: A Stochastic Analysis of the Effect of Delay Uncertainty

Majeed M. Hayat[1], Sagar Dhakal[1], Chaouki T. Abdallah[1], J. Douglas Birdwell[2], and John Chiasson[2]

[1] ECE Dept, University of New Mexico, Albuquerque NM 87131-1356, USA
{hayat,dhakal,chaouki}@eece.unm.edu
[2] ECE Dept, University of Tennessee, Knoxville TN 37996, USA
{chiasson,birdwell}@utk.edu

Summary. In large-scale distributed computing systems, in which the computational elements are physically or virtually distant from each other, there are communication-related delays that can significantly alter the expected performance of load-balancing policies that do not account for such delays. This is a particularly significant problem in systems for which the individual units are connected by means of a shared broadband communication medium (e.g., the Internet, ATM, wireless LAN or wireless Internet). In such cases, the delays, in addition to being large, fluctuate randomly, making their one-time accurate prediction impossible. In this work, the stochastic dynamics of a load-balancing algorithm in a cluster of computer nodes are modeled and used to predict the effects of the random time delays on the algorithm's performance. A discrete-time stochastic dynamical-equation model is presented describing the evolution of the random queue size of each node. Monte Carlo simulation is also used to demonstrate the extent of the role played by the magnitude and uncertainty of the various time-delay elements in altering the performance of load balancing. This study reveals that the presence of delay (deterministic or random) can lead to a significant degradation in the performance of a load-balancing policy. One way to remedy such a problem is to weaken the load-balancing mechanism so that the load-transfer between nodes is down-scaled (or discouraged) appropriately.

1 Introduction

Effective load balancing of a cluster of computational elements (CEs) in a distributed computing system relies on accurate knowledge of the state of the individual CEs. This knowledge is used to judiciously assign incoming computational tasks to appropriate CEs, according to some load-balancing policy [1, 2]. In large-scale distributed computing systems, in which the CEs are physically or virtually distant from each other, there are a number of inherent time-delay factors that can seriously alter the expected performance of the load-balancing policies that do not account for such delays. One manifestation of such time delay is attributable to the computational limitations of the individual CEs. A more significant manifestation of such delay arises from the communication limitations between the CEs. These include delays in transferring loads between CEs and delays in the communication between them. Moreover, these delay elements not only fluctuate within each CE, as the amounts of the loads to be transferred vary, but also fluctuate as a result of the uncertainty in the condition of the communication medium that connects the units. There has been an extensive research in the development of the appropriate dynamic load balancing policies. The policies have been proposed for categories such as local versus global, static versus dynamic, and centralized versus distributed scheduling [3, 4, 5]. Some of the existing approaches consider constant performance of the network while others consider deterministic communication and transfer delay. Here, we propose and investigate a dynamic load balancing scheme for distributed systems which incorporates the stochastic nature of the delay in both communication and load transfer.

To adequately model load balancing problems, several features of the parallel computation environment should be captured including: (1) the workload awaiting processing at each CE (i.e., queue size); (2) the relative performances of the CEs; (3) the computational requirements of each workload component; (4) the delays and bandwidth constraints of CEs and network components involved in the exchange of workloads, and (5) the delays imposed by CEs and the network on the exchange of measurements and information. The authors have previously developed a deterministic model, based on dynamic rate equations, describing the load-balancing dynamics and characterizing conditions for its stability [2, 6, 7, 8]. While thie deterministic model is appropriate when dealing with a dedicated communication medium, it may become inadequate for cases when a shared communication medium is used whereby the delays encountered are stochastic. In this paper, we will focus on the effect of stochastic delay on the performance of load balancing. The effect of delay is expected to be a key factor as searching large databases moves toward distributed architectures with potentially geographically distant units.

This article is organized as follows. In Section 2 we identify the stochastic elements of the load-balancing problem at hand and describe its time dynamics. In Section 3 we present a discrete-time queuing model describing the evolution of the random queue size of each node in the presence of delay for a typical load balancing algorithm. In Section 4 we present the results of Monte-Carlo simulations which demonstrate the extent of the role played by the uncertainty of the various time-delay elements in altering the performance of load balancing from that predicted by deterministic models, which assume fixed delays. Finally, the conclusions are given in Section 5.

2 Description of the Stochastic Dynamics

The load balancing problem in the presence of delay can be generically described as follows. Consider n nodes in a network of geographically-distributed CEs. Computational tasks arrive at each node randomly and tasks are completed according to an exponential service-time model. In a typical load-balancing algorithm, each node routinely checks its queue size against other nodes and decides whether or not to allocate a portion of its load to less busy nodes according to a predefined policy. Now due to the physical (or virtual) distance between nodes in large-scale distributed computing systems, communication and load transfer activity among them cannot be assumed instantaneous. Thus, the information that a particular node has about other nodes at any time is dated and may not accurately represent the current state of the other nodes. For the same reason, a load sent to a recipient node arrives at a delayed instant. In the mean time, however, the load state of the recipient node may have considerably changed from what was known to the transmitting node at the time of load transfer. Furthermore, what makes matters more complex is that these delays are random. For example, the communication delay is random since the state of the shared communication network is unpredictable, depending on the level of traffic, congestion, and quality of service (QoS) attributes of the network. Clearly, the characteristics of the delay depend on the network configuration and architecture, the type of communication medium and protocol, and on the overall load of the system.

Other factors that contribute to the stochastic nature of the distributed-computing problem include: 1) randomness and possible burst-like nature of the arrival of new job requests at each node from external sources (i.e., from users); 2) randomness of the load-transfer process itself, as it depends on some deterministic law that may use a sliding-window history of all other nodes (which are also random); and 3) randomness in the task completion process at each node. In the next section, we lay out a queuing model that characterizes the dynamics of the load-balancing problem described so far.

3 A Discrete-time Queuing Model with Delays

Consider n nodes (CEs), and let $Q_i(t)$ denote the number of tasks awaiting processing at the ith node at time t. Suppose that the ith node completes tasks at a rate μ_i, and new job requests are assigned to it from external sources (i.e., from external users) at a rate λ_i. Note that these incoming tasks come from sources external to the network of nodes and do not included the jobs transferred to a node from other nodes as a result of load balancing. Let the counting process $J_i(t_1, t_2)$ denote the number of such external tasks arriving at node i in the interval $(t_1, t_2]$. To capture any possible burst-like characteristics in the external-task arrivals (as each job request may involve a large number of computational tasks), we will assume that the process $J_i(\cdot, \cdot)$ is a compound

Poisson process [9]. That is, $J_i(t_1,t_2) = \sum_{k:t_1<\tau_k\le t_2} H_k$, where τ_k are the arrival times of job requests (which arrive according to a Poisson process with rate λ_i) and H_k $(k = 1, 2\ldots)$ is an integer-valued random variable describing the number of tasks associated with the kth job request. We next address the load transfer between nodes which will allow us to describe the dynamics of the evolution of the queues.

For the ith node and at its specific load-balancing instants T^i_ℓ, $\ell = 1, 2, \ldots$, the node looks at its own load $Q_i(T^i_\ell)$ and the loads of other nodes at randomly delayed instants (due to communication delays), and decides whether it should allocate some of its load to other nodes, according to a deterministic (or randomized, if so desired) load-balancing policy. Moreover, at times when it is not balancing its load, it may receive loads from other nodes that were transmitted at a randomly delayed instant, governed by the characteristics of the load-transfer delay. With the above description of task assignments between nodes, and with our earlier description of task completion and external-task arrivals, we can write the dynamics of the ith queue in differential form as

$$Q_i(t+\mathit{\Delta} t) = Q_i(t) - C_i(t,t+\mathit{\Delta} t) - \sum_{j\neq i} L_{ji}(t) + \sum_{j\neq i} L_{ij}(t-\tau_{ij}) + J_i(t,t+\mathit{\Delta} t), \quad (1)$$

where

- $C_i(t,t+\mathit{\Delta} t)$ is a Poisson process with rate μ_i describing the random number of tasks completed in the interval $(t, t+\mathit{\Delta} t]$
- τ_{ij} is the delay in transferring the load arriving to node i sent by node j, and finally
- $L_{ij}(t)$ is the load transferred from node j to node i at the time t. Note that $L_{ij}(t)$ is zero except at the load-transfer instants T^j_ℓ, $\ell = 1, 2, \ldots$, for the jth node.

Now for any $k \neq \ell$, the random load $L_{k\ell}(t)$ diverted from node ℓ to node k at a pre-specified load-transfer time t is governed by the mutual load-balancing policy a-priory agreed upon between the two nodes. This policy utilizes knowledge of the state of the ℓth (transmitting) node and the delayed knowledge of the recipient kth node as well as the dated states of all the other nodes. More precisely, we assume $L_{k\ell}(t) = g_{k\ell}(Q_\ell(t), Q_k(t-\eta_{\ell k}), \ldots, Q_j(t-\eta_{\ell j}), \ldots)$, where for any $j \neq k$, $\eta_{kj} = \eta_{jk}$ is the communication delay between the kth and jth nodes. The function $g_{k\ell}$ governs the load-balancing policy between the kth and ℓth nodes. One common example is

$$\begin{aligned} &g_{k\ell}(Q_\ell(t), Q_k(t-\eta_{\ell k}), \ldots, Q_j(t-\eta_{\ell j}), \ldots) \\ &\quad = K_k p_{k\ell} \cdot \Big(Q_\ell(t) - n^{-1} \sum_{j=1}^{n} Q_j(t-\eta_{\ell j}) \Big) \\ &\qquad \cdot u\Big(Q_\ell(t) - n^{-1} \sum_{j=1}^{n} Q_j(t-\eta_{\ell j}) \Big), \end{aligned} \quad (2)$$

where $u(\cdot)$ is the unit step function with the obvious convention $\eta_{ii}(t) = 0$, and K_k is a parameter that controls the "strength" or "gain" of load balancing at the kth (load distributing) node. We will refer to it henceforth as the gain coefficient. In this example, the ℓth node simply compares its load to the average over all nodes and sends out a fraction $p_{k\ell}$ of its excess load, $Q_\ell(t) - n^{-1}\sum_{j=1}^{n} Q_j(t-\eta_{\ell j})$, to the ℓth node. (Of course we require that $\sum_{k\neq\ell} p_{k\ell} = 1$.) This form of policy has been previously adopted and implemented by the authors for a cluster of CEs [1, 2]. Finally, the fractions $p_{\ell k}$ can be defined in a variety of ways. In this work they are defined as follows:

$$p_{k\ell} \triangleq \frac{1}{n-2}\left\{1 - \frac{Q_k(t-\eta_{\ell k})}{\sum_{i\neq\ell} Q_i(t-\eta_{\ell i})}\right\}. \tag{3}$$

In this definition, a node sends a larger fraction of its excess load to a node with a small load relative to all other candidate recipient nodes.

4 Simulation Results

We have developed a custom-made Monte-Carlo simulation software according to our queuing model. We utilized actual data from load-balancing experiments (conducted at the University of Tennessee) pertaining to the number of tasks awaiting processing, average communication delay, average load-transfer delay, and actual load-balancing instants [2]. In the actual experiment, the communication and load-transfer delays were minimal (due the fact that the PCs were all in a local proximity and benefited from a dedicated fast Ethernet). Thus, to better reflect cases when the nodes are geographically distant we synthesized larger delays in communication and load transfer in our simulations.

4.1 Effect of Delay

Three CEs ($n = 3$) were used in the simulations and a standard load-balancing policy [as described by (2)] was implemented. The PCs were assumed to have equal computing power (the average task completion time was 10 μs per task), but the initial load was distributed unevenly among the three nodes as 7000, 4500, and 500 tasks, with no additional external arrival of tasks (viz., $J_1(t_1,t_2) = 7000, J_2(t_1,t_2) = 4500, J_3(t_1,t_2) = 500$ only if $t_1 = 0, 0 < t_2$ and they are zero otherwise). Figure 1 corresponds to the case where no communication nor load-transfer delays are assumed. This case approximates the actual experiment [1], where all the computers were within the proximity of each other benefiting from a dedicated fast Ethernet. Note that the system is balanced at approximately 15 ms and remains balanced thereafter until all tasks are executed in approximately 39 ms.

We next considered the presence of deterministic communication delay of 8 ms and a load transfer-delay of 16 ms. The behavior is seen in Fig. 2, where

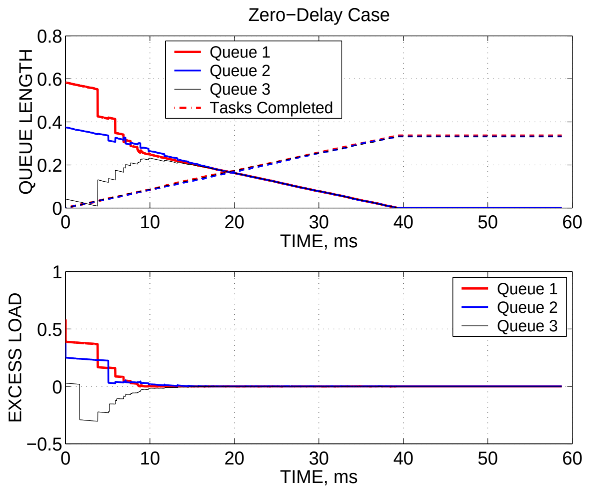

Fig. 1. Top: Queue size in the ideal case when delays are nonexistent. The queues are normalized by the total number of submitted tasks (12000 in this case). The dashed curves represent the tasks completed cumulatively in time by each node. **Bottom:** Excess queue length for each node computed as the difference between each nodes normalized queue size and the normalized queue size of the overall system. Note that the three nodes are balanced at approximately 15 ms and that all tasks are completed in approximately 39 ms.

it is observed that the delay prevents load balancing to occur. For example, nodes 1 and 2 each eventually executes approximately 40% of the total tasks, whereas node 3 executes only 20% of the total tasks submitted to the system (as seen from the dashed curves in the top figure in Fig. 2). The conclusion drawn here is that the presence of delay in communication and load transfer seriously disturbs the performance of the load balancing policy, as each node utilizes *dated* information about the state of the other nodes as it decides what fraction of its load must be transferred to each of the other nodes.

To see the effect of the delay randomness on the load balancing performance, two representative realizations of the performance were generated and are shown in Figs. 3 and 4. The average delays were taken as in the deterministic case (i.e., 8 ms for the communication delay and 16 ms for the load-transfer delay). For simplicity and due to lack of availability of detailed information on the statistics of the delays, the delays were assumed to be uniformly-distributed in the range extending from 0 to twice their mean values. For the example considered, it turns out that the performance is sensitive to the realizations of the delays in the early phase of the load-balancing procedure. For example, it is seen from the simulation results that a deterministic (fixed) delay can lead to a more severe performance degradation than the case when the delays are assumed random (with the same mean as the determinis-

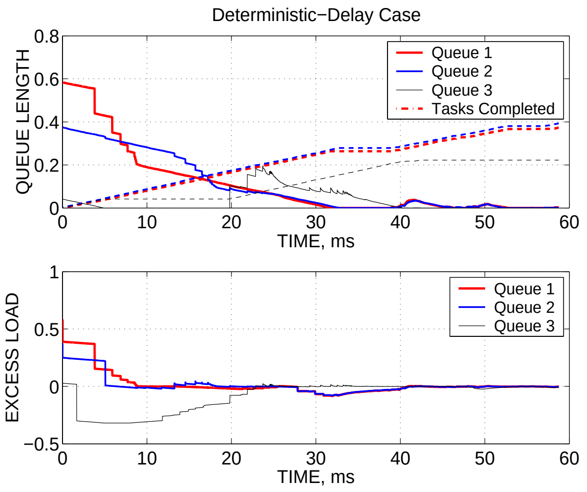

Fig. 2. Similar to Fig. 1 but with a deterministic communication and load-transfer delays of 8 ms and 16 ms, respectively. In contrast to the zero-delay case, the three nodes are balanced at approximately 60 ms and all tasks are completed shortly afterwards. Also note that nodes 2 and 3 each execute approximately 40% of the total tasks, where node 3 executes only 20% of the total tasks submitted to the system.

tic case). To see the average effect of the random delay, we calculated the mean queue size and the normalized variance (normalized by the mean square) over 100 realizations of the queue sample functions, each with a different set of randomly generated delays. The results are shown in Figs. 5 and 6. It is seen from the mean behavior that the randomness in the delay actually leads, on average, to balancing characteristics (as far the excess-load is concerned) that are superior to the case when the delays are deterministic! However, there is a high level of uncertainty in the queue size, and hence in the load balancing. It is seen from Fig. 5 (dashed curves) that the average total number of tasks completed by each node continues to increase well beyond 60 ms, which is inferred from the positive slope of the dashed curves. This indicates that in comparison to the deterministic-delay case, the system requires (1) almost twice as long as the zero-delay case to complete all the tasks and (2) a longer time to complete all the tasks than the deterministic-delay case.

4.2 Interplay Between Delay and the Gain Coefficient K

We now consider the effect of varying the gain coefficient K on the performance of load balancing (assume that $K_1 = K_2 = K_3 \equiv K$). Figures 7 and 8 show the performance under two cases corresponding to a large and small gain coefficient, $K = 0.8$ and $K = 0.2$, respectively. It is seen that when

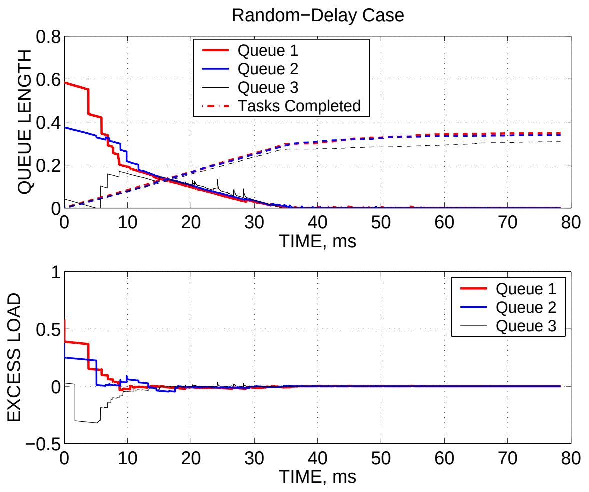

Fig. 3. In this example, the communication and load-transfer delays are assumed random with average values of 8 ms and 16 ms, respectively. Note that the performance is somewhat superior to the deterministic-delay case shown in Fig. 2.

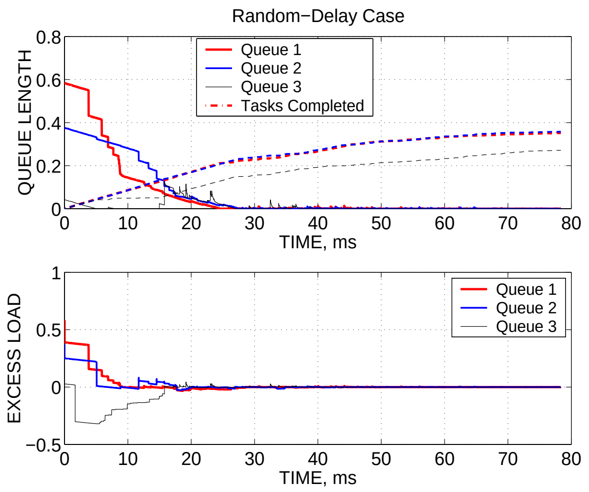

Fig. 4. Another realization of the case described in Fig. 3 showing the variability in the performance from one realization to another. Load-balancing characteristics here are inferior to those in Fig. 3.

$K = 0.8$, the queue lengths fluctuate more than the case when $K = 0.2$, resulting in a longer overall time to total task completion. This example shows that a "weak" load-balancing policy can outperform a "strong" policy in the

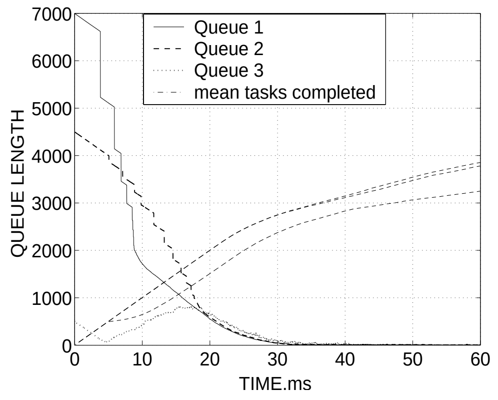

Fig. 5. The empirical average queue length using 100 realizations of the queues for each node (solid curves). The dashed curves are the empirical average of the number of tasks performed by each node cumulatively in time normalized by the total number of tasks submitted to the system. Only 87% of the total tasks are completed within 60 ms.

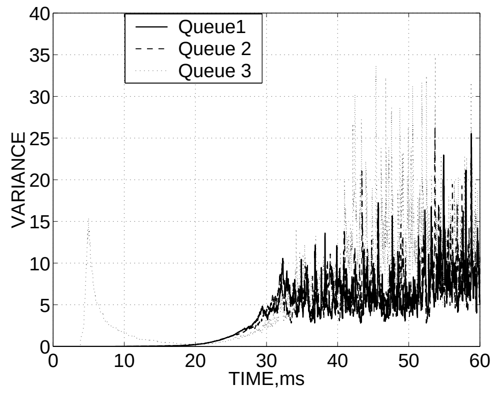

Fig. 6. The empirical variance of the queue length normalized by the mean-square values. Observe the high-degree of uncertainty in the lowest queue as well as the variability at large times, which is indicative of the fact that nodes continue to exchange tasks back and forth, perhaps unnecessarily.

presence of random delay. We will revisit this interesting observation in more detail in the next section.

4.3 Load Dependent Delay

Clearly, the nature of the transfer delay depends on the amount of load to be transferred; a sizable load will entail, on average, a longer transfer delay

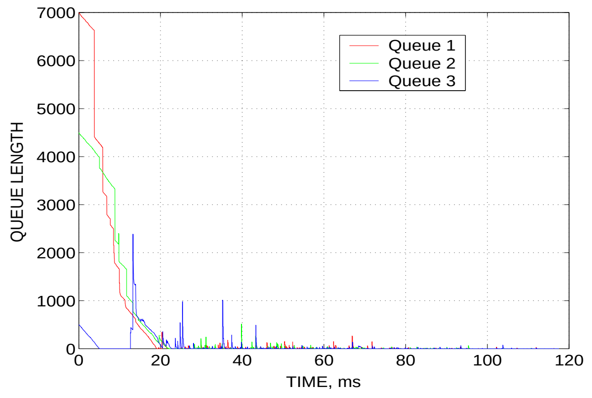

Fig. 7. Queue size versus time when the gain coefficient is $K = 0.8$, corresponding to a "strong" load-balancing policy. Notice the abundance of fluctuations in the tail of the queue in comparison to Fig. 8.

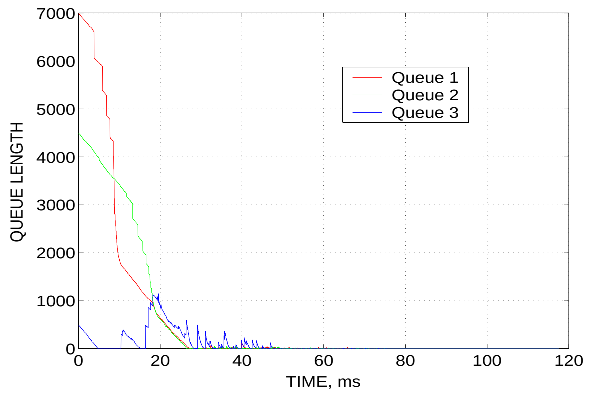

Fig. 8. Same as Fig. 7 but with $K = 0.2$, corresponding to "weak" load-balancing policy.

than a small load. As a consequence, the load balancing policy is directly affected by the load-dependent nature of transfer delay. For example, if there is a high degree of load imbalance present at any time, it might seem tempting to redistribute big packets of data up front so as to rid the imbalance quickly. However, the tradeoff here is that the sizable load takes much longer to reach the destination node, and hence, the overall computation time will inevitably increase. Thus, we would expect the gain coefficient K to play an important role in cases when transfer delay is load dependent. Since the balancing is done frequently, it is intuitively obvious that we would be better off if we were to select K conservatively. To address this issue quantitatively, we will need to develop a model for the load-dependent transfer delay. This is done next.

We propose to capture the load-dependent nature of the random transfer delay τ_{ij} by requiring that its average value, θ_{ij}, assumes the following form

$$\theta_{ij} = d_{\min} - \frac{1 + \exp([L_{ij}(t)d\beta]^{-1})}{1 - \exp([L_{ij}(t)d\beta]^{-1})}, \tag{4}$$

where $d_{\min}$ is the minimum possible transfer delay (its value is estimated as 9 ms in this paper), d is a constant (equal to 0.082618), and β is a parameter which characterizes the transfer delay (selected as 0.04955). Moreover, we will assume that conditional on the size of the load to be transferred, the random delay τ_{ij} is uniformly-distributed in the interval $[0, 2\theta_{ij}]$. This model assumes that up to some threshold, the delay is constant (independent of the load size) that is dependent on the capacity of the communication medium. Beyond this threshold, however, the average delay is expected to increases monotonically with the load size. The parameters d and b are selected so that the above model is consistent with the overall average delay for all the actual transfers that occurred in the simulation. We omit the details.

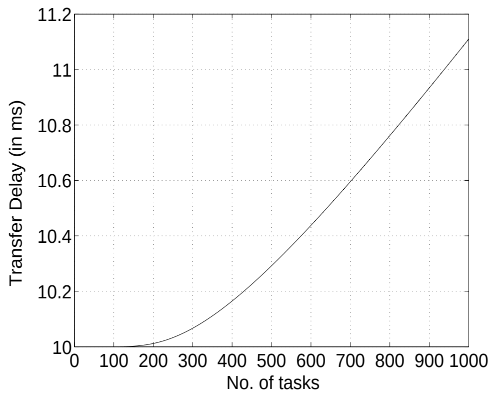

Fig. 9. The load-dependent transfer delay as a function of the load size according to the model shown in (4).

The load-dependent transfer delay versus the load is shown in the Fig. 9. The transfer delay for the loads sent from node 1 to node 3 (top) and from node 2 to node 3 (bottom) over the period of execution time is shown in Fig. 10. With the average communication delay being equal to 8 ms (as before) and the transfer delay made load dependent, according to the model described in (4), one realization of the load-balancing performance for $K = 0.5$ was generated and it is shown in Fig. 11. As expected, the performance deteriorates beyond the case corresponding to a fixed transfer delay. For example, we see from the

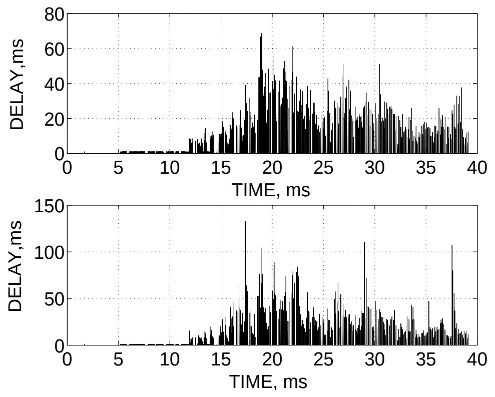

Fig. 10. Scatter-plot of the transfer-delay showing its fluctuations for a particular realization of the queues.

figure that a load sent by node 1 at approximately 5 ms arrives at node 3 approximately 50 ms later, thereby bringing more fluctuation to the tail of the queues. The average effect (over 50 realizations) of this delay model for

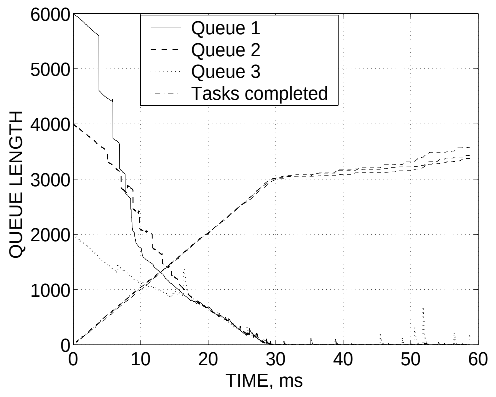

Fig. 11. Queue size and cumulative (in time) number of tasks completed (by each node) as a function of time. These graphs show that queues become more uncertain when the load-transfer delays are load dependent. ($K = 0.5$ in this example.)

two different gain coefficients ($K = 0.1$ and $K = 0.9$) can be seen in Figs. 12 and 13. When $K = 0.9$, the queue is fluctuating even beyond $t = 80$ ms while when $K = 0.1$, all the tasks are completed at approximately 60 ms. The optimal value of K for this delay model was found to be equal to $K = 0.06$ and the overall completion time in this case was 54.85 ms. The variation of the overall completion time with respect to the gain coefficient is shown in Table 1 below.

Table 1. Dependence of the load-balancing performance on the gain coefficient K.

Gain (K)	Task completion time (ms)	Time to execute 95% of tasks (ms)
0.01	62.53	41.80
0.02	61.44	42.86
0.03	59.68	42.59
0.04	57.27	41.98
0.05	56.79	41.35
0.06	54.85	41.99
0.07	56.04	42.49
0.08	59.68	41.56
0.09	62.53	41.81
0.1	61.10	42.18
0.2	65	43.38
0.3	63.40	46.2
0.4	78.313	53.33
0.5	> 80	55.21

It is clearly seen in Fig. 13 that the required time for completing all tasks (in the system) is significantly larger than the time required to execute 95% percent of the assigned tasks. This difference increases with higher values of K. This is due to the fact that even when all the queues are *almost* depleted of tasks, they continue to execute the balancing policy. As a result, small amounts of tasks (e.g., one or two) are sent from one node to other nodes and vice versa. This unnecessary task-swapping significantly increases the transfer delay, therefore increasing the overall computational time. This phenomenon is clearly depicted in Fig. 13 where the minute fluctuations are evident near the tail of the queues.

5 Summary and Conclusions

Whenever there are tangible communication limitations between nodes in a distributed system, possibly with geographically-distant computational elements, we must take a totally new look at the problem of load balancing. In such cases, the presence of non-negligible random delays in inter-node communication and load transfer can significantly alter the expected performance of existing load-balancing strategies. The load-balancing problem must be viewed as a stochastic system, whose performance must be evaluated statistically. More importantly, the policy itself must be developed with appropriate statistical performance criteria in mind. Thus, if we design a load-balancing

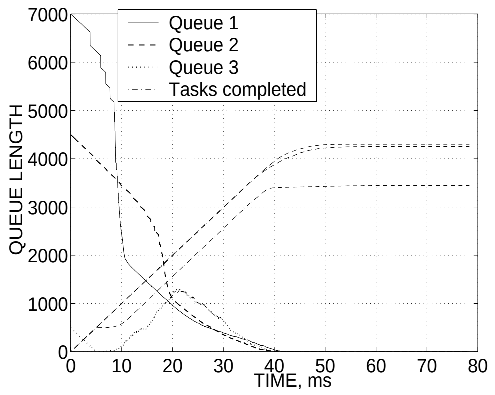

Fig. 12. Queue size versus time for the case where the transfer delay is load dependent. The gain coefficient K is 0.1. Note that the total execution time is approximately 60 ms.

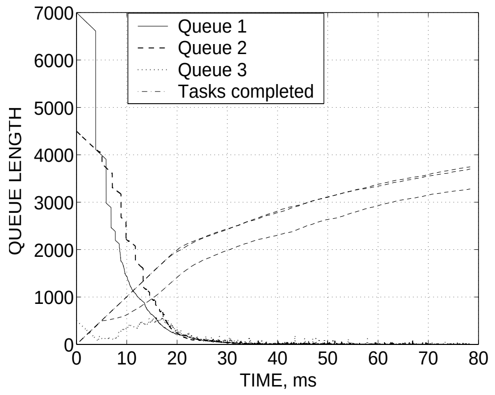

Fig. 13. Same as Fig. 12 but with $K = 0.9$. The queues fluctuate even at 80 ms.

policy under the no-delay or fixed-delay assumptions, the policy will not perform as expected in a real situation when the delays are non-zero or random. A load-balancing policy must be designed with the stochastic nature of the delay in mind.

Monte-Carlo simulation indicates that the presence of delay (deterministic or random) can lead to a significant degradation in the performance of a load-balancing policy. Moreover, when the delay is stochastic, this degradation is worsened, leading to extended cycles of unnecessary exchange of tasks (or loads), back and forth between nodes, leading to extended overall delays and prolonged task-completion times. One way to remedy such a problem is to weaken the load-balancing mechanism so that the load-transfer between nodes is down-scaled (or discouraged) appropriately. This action makes the load-balancing policy in the presence of random delay “less reactionary” to changes in the load distribution within the system. This, in turn, reduces the sensitivity of the load-balancing process to inaccuracies in the state-of-knowledge of each node about the load distribution in the remainder of the system caused by communication limitations.

6 Acknowledgements

This work was supported by the National Science Foundation (NSF) under Information Technology Research (ITR) grant No. ANI-0312611. Additional support was also provided by NSF under grant No. INT-9818312. The work of J.D. Birdwell was supported by U.S. Department of Justice, Federal Bureau of Investigation under contract J-FBI-98-083. Drs. Birdwell and Chiasson were also partially supported by a Challenge Grant Award from the Center for Information Technology Research at the University of Tennessee. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Government.

References

1. J. D. Birdwell, T. W. Wang, R. D. Horn, P. Yadav, and D. J. Icove, “Method of indexed storage and retrieval of multidimensional information,” U. S. Patent Application 09/671,304 (2000)
2. C. T. Abdallah, N. Alluri, J. D. Birdwell, J. Chiasson, V. Chupryna, Z. Tang, and T. Wang “A linear time delay model for studying load balancing instabilities in parallel computations,” *The International Journal of System Science*, to appear (2003)
3. T.L. Casavant and J.G. Kuhl, “A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems,” *IEEE Trans. Software Eng.*, vol. 14, pp. 141–154 (Feb. 1988)
4. Zhiling Lan, Valerie E. Taylor, Greg Bryan, “Dynamic Load Balancing for Adaptive Mesh Refinement Application,”*In Proc. Of ICPP’2001*, Valencia, Spain (2001)
5. G. Cybenko, “Dynamic load balancing for distributed memory multiprocessors,” *IEEE Transactions on Parallel and Distributed computing*, vol. 7, pp. 279–301 (1989)
6. J. D. Birdwell, J. Chiasson, Z. Tang, T. Wang, C. T. Abdallah, and M. M. Hayat, “Dynamic time delay models for load balancing Part I: Deterministic models,” *CNRS-NSF Workshop: Advances in Control of Time-Delay Systems*, Paris, France (Jan. 2003)
7. C. T. Abdallah, J.D. Birdwell, J. Chiasson, V. Churpryna, Z. Tang, and T.W. Wang “Load balancing instabilities due to Time delays in farallel computation,” *Proceedings of the 3rd IFAC Conference on Time Delay Systems*, Santa Fe, NM (Dec. 2001)

8. M. M. Hayat, C. T. Abdallah, J. D. Birdwell and J. Chiasson, "Dynamic time delay models for load balancing, Part II: A stochastic analysis of the effect of delay uncertainty," *CNRS-NSF Workshop: Advances in Control of Time-Delay Systems*, Paris, France (Jan. 2003)
9. D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes* (Springer-Verlag, New York, NY 1988)