Load Balancing in Distributed Systems with Large Time Delays: Theory and Experiment

J. Ghanem S. Dhakal M. M. Hayat H. Jérez C. T. Abdallah J. Chiasson*

Department of Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM 87131-1356 USA
{jean,dhakal,hayat,hjerez,chaouki}@ece.unm.edu

*Department of Electrical and Computer Engineering
University of Tennessee
Knoxville, TN 37996-2100 USA
chiasson@utk.edu

***Abstract*— In distributed-computing environments with high communication delays and cost, limiting the number of balancing instants in scheduling algorithms results in an improved performance compared to the case where load balancing is executed continuously. Therefore, finding the optimal number of balancing instants and optimizing the performance over the inter-balancing time and load-balancing gain becomes an important problem. In this paper, the performance of a previously reported single load-balancing strategy on a distributed physical system is studied. The performance is also compared to simulations generated by an in-house wireless distributed system. Based on the concept of regeneration in stochastic processes, a mathematical model for the queuing systems representing a distributed system with two nodes is developed. In particular, a system of four difference-differential equations is obtained characterizing the mean of the overall completion time.**

I. Introduction

The effectiveness of load balancing in a cluster of distributed computational elements (CEs) relies heavily on the accurate knowledge of the state of the individual CEs as well as the delays involved in transferring loads from one CE to another. For example, the shared knowledge of the load state of the system is used by individual CEs to judiciously assign an appropriate fraction of the incoming loads to less busy CEs, according to some load-balancing policy. However, in systems that use a shared communication medium (such as the Internet or a wireless LAN), there is an inherent delay in the inter-node communications and transfer of loads. Moreover, such delays vary according to the size of the loads to be transferred and also fluctuate due to the random condition of the communication medium that connects the CEs.

The load balancing policies have previously been proposed for categories such as local versus global, static versus dynamic, and centralized versus distributed scheduling [4], [3], [5]. Some of the existing approaches consider constant performance of the network while others consider deterministic communication and transfer delay. Recently, the limitations and the overheads involved with the continuous implementation of the balancing policy have been discussed in [7], [8]. It is almost universally observed that the types of delay described above degrade the overall performance [3], [4], [5], [6]. In particular, communication delays may lead to unnecessary transfer of loads (as each CE has only dated information about the state of other CEs), and large load-transfer delays may lead to a situation where much time is wasted on transferring loads while certain CE's may be idle during the transfer. In [6], it is assumed that the communication channels have fixed delay times and the load balancing is completed within a finite interval. But in reality, the delays are random and hence such balancing policies are not applicable to the delay-limited distributed systems. Recently, our Monte-Carlo studies [7], [8] was used to show that indeed there is an interplay between the stochastic delay (e.g., its mean and its dependence on the load) and the strength and frequency of balancing. Moreover, it has been shown in [8] that limiting the number of load balancing instants while optimizing the strength of the load balancing and the actual load-balancing instants is a feasible solution to the problem of load balancing in a delay-limited environment. It was also observed that by fixing the number of load balancing instants (in an effort to avoid the unnecessary exchange of loads between CEs), the sensitivity of the balancing policy to the balancing time (relative to the time when load arrives at the system) is significantly increased. This is primarily due to the observation that it is more advantageous to balance at a delayed time simply because the CEs will have more time to exchange their respective load states, thereby allowing for "better-informed" load balancing.

In this paper, we primarily focus on the implementation of the single load balancing policy on a real distributed system and evaluate its performance with respect to the balancing instant and the gain parameter. We show that

the choice of the balancing strategy is an optimization problem with respect to the choice of the gain parameter. The results of this implementation are compared to the results obtained from the custom-made Monte Carlo simulation software which we developed for our preliminary work. Finally, an approach for modeling the distributed system dynamics, as a queuing system, is introduced using the concept of regeneration in stochastic processes. We define the events (called regeneration events) whose occurrence will regenerate the queues with similar statistical properties and dynamics. More specifically, on every arrival or departure of a task at any of the nodes, the system of queues is said to be regenerated because they preserve the same stochastic dynamics of the queues prior to this arrival/departure event but will start from different set of initial conditions. The initial conditions consist of the initial load of queues as well as their knowledge state of each other. To this end, a novel concept of "information states" of the queues is introduced to handle the complexity of the initial conditions. The system model, which is developed here for the special case of two CEs, also maintains the gist of the multi-CE problem and conveys the underlying principles of our analytical solution while keeping the algebra at a minimum. The approach, nonetheless, can be extended to the multi-CE case in a straightforward fashion.

II. Description of the load balancing policy

We begin by briefly describing the queuing model that characterizes the stochastic dynamics of the load balancing problem described [7], [9]. Suppose that we have a cluster of n nodes. Let $Q_i(t)$ denote the number of tasks awaiting processing at the ith node at time t. Assume that the ith node completes tasks according to a Poisson process and at a constant rate μ_i. Let the counting process $J_i(t_1, t_2)$ denote the number of external tasks (requests) arriving at node i in the interval $[t_1, t_2)$. (For example, $J_i(t_1, t_2)$ can be a compound Poisson process with a constant rate λ_i [10].) The load transfer between nodes, denoted by $L_{ji}(t)$, is addressed next. The ith node, at the lth specific load-balancing instant t_l^i, looks at its own load $Q_i(t_l^i)$ and the loads of other nodes at randomly delayed instants (due to communication delays), and decides whether it should allocate a fraction K of its excess load to the other nodes according to a deterministic policy. Moreover, at the time when it is not balancing, it may receive loads from the neighboring nodes subject to random delays (due to the load-transfer delays). In all the examples considered later in this paper, only one load-balancing execution is permitted. That is, $t_l^i = \infty$, for all $l \geq 2$ and all $i \geq 1$.

We now write the dynamics of the ith queue in a differential form as (in Δt time increments) as follows:

$$\begin{aligned} Q_i(t+\Delta t) &= Q_i(t) - C_i(t, t+\Delta t) \\ &- \sum_{j\neq i}\sum_{l} L_{ji}(t) I_{\{t_l^i = t\}} \\ &+ \sum_{j\neq i}\sum_{k} L_{ij}(t-\tau_{ij,k}) I_{\{t_k^j = t-\tau_{ij,k}\}} \\ &+ J_i(t, t+\Delta t), \end{aligned} \tag{1}$$

where I_A is an indicator function for the event A, $C_i(t, t+\Delta t)$ is a Poisson process (with rate μ_i) describing the random number of tasks completed in the interval $[t, t+\Delta t)$, and $\tau_{ij,k}$ is the delay in transferring load $L_{ij}(t-\tau_{ij,k})$ from node j to node i at the kth load balancing instant of node j. More precisely, for any $k \neq l$, the random load $L_{kl}(t)$ diverted from node l to node k has the form $L_{kl}(t) \triangleq g_{kl}(Q_l(t), Q_k(t-\eta_{lk}), \ldots, Q_j(t-\eta_{lj}), \ldots)$, where for any $j \neq k$, $\eta_{kj} = \eta_{jk}$ is the communication delay between the kth and jth nodes. The function g_{kl} dictates the load-balancing policy between the kth and lth nodes. In this paper, the following equation is used.

$$\begin{aligned} &g_{kl}(Q_l(t), Q_k(t-\eta_{lk}), \ldots, Q_j(t-\eta_{lj}), \ldots) \\ &= Kp_{kl} \cdot \left(Q_l(t) - n^{-1}\sum_{j=1}^{n} Q_j(t-\eta_{lj})u(t-\eta_{lj})\right) \cdot \\ &\qquad u\left(Q_l - n^{-1}\sum_{j=1}^{n} Q_j(t-\eta_{lj})u(t-\eta_{lj})\right), \end{aligned} \tag{2}$$

where $u(\cdot)$ is the unit step function with the obvious convention $\eta_{ii} = 0$, and K is a parameter that controls the "strength" or "gain" of load balancing at the kth (load distributing) node. In this example, the lth node simply compares its load to the average over all load and sends out a fraction p_{kl} of its excess load to the kth node. (Of course, $\sum_{k\neq l} p_{kl} = 1$). Finally, the fractions p_{kl} are defined as:

$$p_{kl} = \begin{cases} \frac{1}{n-2}\left(1 - \frac{Q_k(t-\eta_{lk})}{\sum_{i\neq l} Q_i(t-\eta_{li})}\right), & k \neq l, \eta_{lk}(t) \leq t \\ \frac{1}{n-1}, & \text{otherwise} \end{cases} \tag{3}$$

where $n \geq 3$. 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. For the special case when $n = 2$, $p_{kl} = 1$, where $k \neq l$.

III. Experimental Results

We have developed an in-house wireless testbeds to study the effects of the gain parameter K as well as the selection of the load-balancing instant. We would like to highlight the importance of these experiments since, to our knowledge, no previous work has been done in the optimization with respect to the load-balancing instant t_b especially over a wireless network. The details of the system are described below.

A. Description of the experiments

The experiments were conducted over a wireless network using an 802.11b access point. The testing was completed on three computers: a 1.6 GHz Pentium IV processor machine (node 1) and two 1 GHz Transmeta Processor machines (nodes 2 & 3). To increase communication delays between the nodes (so as to bring the test-bed to setting that resembles a realistic setting of a busy network), the access point was kept busy by third party machines which continuously downloaded files. We consider the case where all nodes execute the load balancing at a common balancing time t_b. The application used to illustrate the load balancing process was matrix multiplication, where one task has been defined as the multiplication of one row by a static matrix duplicated on all nodes (3 nodes in our experiment). The size of the elements in each row was generated randomly from a specified range which made the execution time of a task variable. On average, the completion time of a task was 525 ms on node 1, and 650 ms on the other two nodes. As for the communication part of the program, UDP was used to exchange queue size information among the nodes and TCP was used to transfer the data or tasks from one machine to another. The load balancing policy used in the experiments, is governed by (3) where the node decides to use either one according to its knowledge states.

The aim of the first experiment is to optimize the overall completion time with respect to the balancing instant t_b by setting the gain value K to 1. Each node was assigned a certain number of tasks according to the following distribution: Node 1 was assigned 60 tasks, node 2 was assigned 30 tasks, and node 3 was assigned 120 tasks. The information exchange delay (viz., communication delay) was on average 850 ms. Several experiments were conducted for each case of the load-balancing instant and the average was calculated using five independent realizations for each selected value of the load-balancing instant. In the second set of experiments, the load balancing instant was fixed at 1.4 s in order to find the optimal gain that minimizes the overall completion time. The initial distribution of tasks was as follows: 60 tasks were assigned to node 1, 150 tasks were assigned to node 2, and 10 tasks were assigned to node 3. The information exchange delay was 322 ms and the data transfer delay per task was 485 ms.

B. Discussion of results

The results of the first set of experiments show that if the load balancing is performed blindly, as in the onset of receiving the initial load, the performance is poorest. This is demonstrated by the relatively large average completion time (namely 45s$\sim$50s) when the balancing instant is prior to the time when all the communication between the CEs have arrived (namely when t_b is approximately below 1s), as shown in Fig. 1. Note that the completion time drops significantly (down to 40s) as t_b begins to approximately exceed the time when all inter-CE communications have arrived (e.g., $t_b > 1.5$s). In this scenario of t_b, the load balancing is done in an informative fashion, that is, the nodes have knowledge of the initial load of every CE. Thus, it is not surprising that the load balancing is more effective than the case the load balancing is performed on the onset of the initial load arrival for which the CEs have not yet received the state of the other CEs. One explanation for the sudden rise in the completion time between the 0.5s and 1s balancing instances, is that the knowledge states in the system are "hybrid" (some nodes are aware of the queue sizes of the others and some aren't). This fact dictates the use at the same time of the 2 policy equations (3) by the system, resulting in an uneven distribution of the load across the nodes. Finally, we observe that as t_b increases farther beyond the time all the inter-CE communications arrive (e.g., $t_b > 5$s), then the average completion time begins to increase. This occurs precisely because any delay in executing the load balancing beyond the arrival of the inter-CE communications time would enhance the possibility that some CEs will run out of tasks in the period before any transferred load arrives to it.

Next we examine the size of the loads transferred as a function of the instant at which the load balancing is executed, as shown in Fig. 2. This behavior will show that the dependence of the size of the total load transferred on the "knowledge state" of the CEs. It is clear from the figure that for load-balancing instants up to approximately the time when all CEs have accurate knowledge of each other's load states, the average size of the load assigned for transfer is unduly large. Clearly, this seemingly "uninformed" load balancing leads to the waste of bandwidth on the interconnected network.

The results of the second set of experiments indeed confirm our earlier prediction (as reported in [8]) that when communication and load-transfer delays are prevalent, the load-balancing gain must be reduced to prevent "overreaction" (sending unnecessary excess load). This behavior is shown in Fig. 3, which demonstrates that the optimal performance is achieved not at the maximal gain ($K = 1$) but when K is approximately 0.8. This is a significant result as it is contrary to what we would expect in a situation when the delay is insignificant (as in a fast Ethernet case), where $K = 1$ yields the optimal performance. Figure 4 shows the dependence of the total load to be transferred as a function of the gain. A large gain (near unity) results in a large load to be transferred, which, in turn, leads to a large load-transfer delay. Thus, large gains increase the likelihood of a node (that may not have been overloaded initially) to complete all its load and remain idle until the transferred load arrives. This would clearly increase the total average task completion

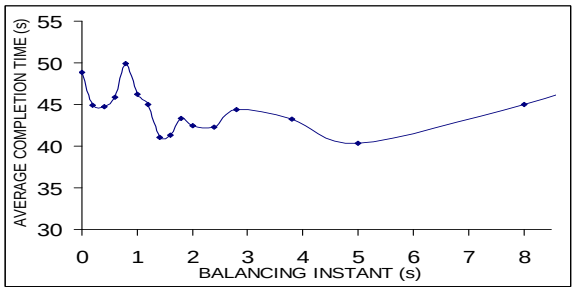

Fig. 1. Average total task-completion time as a function of the load-balancing instant. The load-balancing gain parameter is set at $K = 1$. The dots represent the actual experimental values and the solid curve is a best polynomial fit. This convention is used thought out Fig. 4

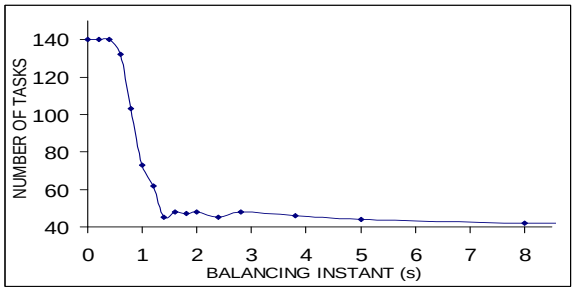

Fig. 2. Average total excess load decided by the load-balancing policy to be transferred (at the load-balancing instant) as a function of the balancing instant. The load-balancing gain parameter is set at $K = 1$.

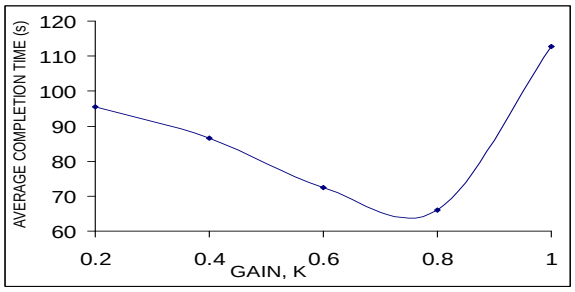

Fig. 3. Average total task-completion time as a function of the balancing gain. The load-balancing instant is fixed at 1.4 s.

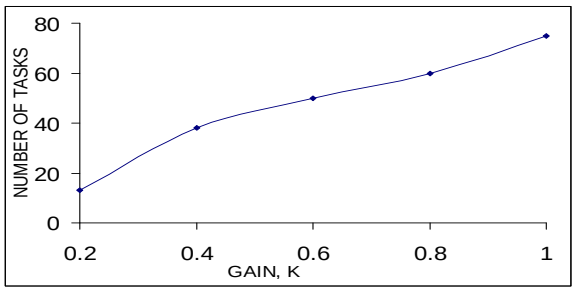

Fig. 4. Average total excess load decided by the load-balancing policy to be transferred (at the load-balancing instant) as a function of the balancing gain. The load-balancing instant is fixed at 1.4 s.

time, as confirmed earlier by Fig. 3.

IV. Simulation Results

We have generated a Monte-Carlo simulation tool that allows the simulation of the queues described in Section II [Eqns. (1) to (3)]. The network parameters (i.e. Communication delays η_{kj} and load transfer delays τ_{ij}) and the task execution time in the simulation were set to the respective average values obtained from the experiments described in Section III-A. We used this simulation tool to validate the correspondence between the stochastic queuing model and the experimental setup. In particular, we have generated the simulated versions of Figs. 1–3, which are shown below.

It is observed that the general characteristics of the curves are very similar, but they are not exactly identical, due to the un-predictive behavior and complexity of the wireless environment. Nerveless, the result of the first simulation, shown in Fig. 5, were consistent with the experimental result where we can clearly identify the sudden rise in the completion time around the balancing instant corresponding to the communication delay (850 ms). The reason was described in the experimental section. As for the excess transferred load plotted in Fig. 6, the simulation resulted in the same curve and transition shape obtained from the experiment.

The curve characteristics of the second simulation, shown in Fig. 7, are analogous to the ones obtained in the experiment. Indeed, the gain values found are almost the same: 0.8 from the experiment and 0.87 from the simulation. As indicated before, the small difference is due to the unstable delay values and other factors present in the wireless environment which has been approximated both by the model and the simulator.

V. Stochastic Analysis of the Queuing Model: A Regeneration Approach

Motivated by the fact that we are dealing with an optimization problem, in which we wish to optimize the load-balancing gain and the balancing instants to minimize the average completion time, we will outline a novel regenerative approach that will fit the queuing model described in Section II. The concept of regeneration has proven to be a powerful tool in the analysis of complex stochastic systems [1], [2], [10]. Consider n nodes in a network of geographically-distributed CEs with some random initial workload. We are interested to know the average overall completion time if only one-time balancing is allowed and hence decide when to balance and what is the appropriate balancing gain K such that the average completion time is minimized. Here, we discuss the behavior of the zero-input response

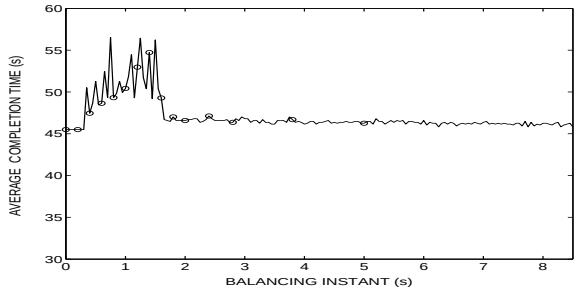

Fig. 5. Simulation results for the average total task-completion time as a function of the load-balancing instant. The load-balancing gain parameter is set at $K = 1$. The dots represent the actual experimental values.

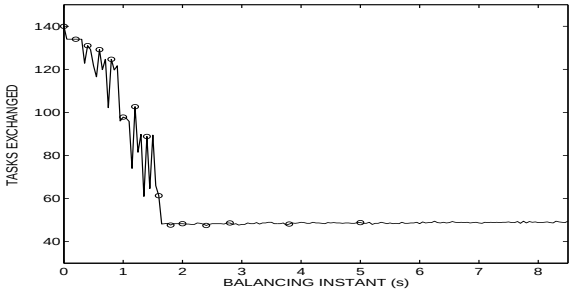

Fig. 6. Simulation results for the average total excess load decided by the load-balancing policy to be transferred (at the load-balancing instant) as a function of the balancing instant. The load-balancing gain parameter is set at $K = 1$.

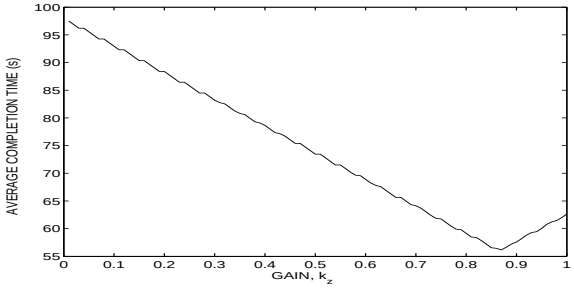

Fig. 7. Simulation results for the average total task-completion time as a function of the balancing gain. The load-balancing instant is fixed at 1.4 s.

of the queues and hence do not have any task arrival at any of the nodes.

A. Rationale

The idea of our approach is to define an *initial event*, defined as the completion of a task by any node or the arrival of a communication by any node, and analyzing the queues that emerge immediately after the occurrence of the initial event. We assume that initially all queues have zero knowledge about the state of the other queues. The point here is that immediately after the occurrence of the initial event, we will have a set of new queues whose stochastic dynamics are identical to the original queues. However, there will be a different set of initial conditions (i.e., different initial load distribution if the initial event is a task completion) or different knowledge state (if the initial event happens to be a communication arrival rather than a task completion). Thus, in addition to having an initial load state, we introduce the novel concept of knowledge states to be defined next.

In a system of n nodes, any node will receive $n-1$ number of communications, one from each of the other nodes. Depending upon the choice of the balancing instant, a node may receive all of those communication or may receive none by the time balancing is done. We assign a vector of size $n-1$ to each of the nodes and initially set all its elements to 0 (corresponding to the null knowledge state). If a communication arrives from any of the node, the bit position corresponding to that particular node is set to 1. There will be a total of $2^{n(n-1)}$ number of knowledge states. In the case when two nodes are present, the knowledge states are: 1) state $(0,0)$, corresponding to the case when the nodes do not know about each others initial load; 2) state $(1,1)$, when both nodes know about each other's initial load states; 3) $(1,0)$, corresponding to the case when only node 1 knows about node 2; and 4) state $(0,1)$, which is the opposite of the $(1,0)$ case.

B. Dynamic Model Base

To simplify the description, we consider the case where only two nodes are present. We will assume that each node has an exponential service time with parameters λ_{D1} and λ_{D2}, respectively. Let m and n be the initial number of tasks present at nodes 1 and 2, respectively. The communication delays from node 1 to node 2 and from node 2 to node 1 are also assumed to follow an exponential distribution with rates λ_{21} and λ_{12}, respectively. Let W, X, Y and Z be the waiting times for the departure of the first task at node 1, departure of the first task at node 2, the arrival of the communication sent from node 1 to node 2 and the arrival of the communication sent from 2 to 1, respectively. Let T=min(W,X,Y,Z). Now the probability density function (pdf) of T can be characterized as $f_T(t) = \lambda e^{-\lambda t}u(t)$, where $\lambda = \lambda_{D1} + \lambda_{D2} + \lambda_{21} + \lambda_{12}$.

Let $\mu_{m,n}^{k1,k2}(t_b)$ be the expected value of the overall completion time given that the balancing is executed at

time t_b, where nodes 1 and 2 are assumed to have m and n tasks at time $t = 0$, and the system knowledge state is $(k1, k2)$ at time $t = 0$. Suppose that the initial event happens to be the departure of a task at node 1 at time $t = \tau$, $0 \leq \tau \leq t_b$. At this instant, the system dynamics remains the same except that node 1 will now have $m-1$ tasks. Thus, the queue has re-emerged (with a different initial load, nonetheless) and the average of the overall completion time is now $\tau + \mu_{m-1,n}^{k1,k2}(t_b - \tau)$. The effect of other possibilities for the initial event are taken into account similarly. However, to calculate this we need to define the completion time for all cases, i.e., the system initially being in any of the four knowledge states. Therefore, based on this discussion we characterize the average of the completion times for all four cases below, namely, $\mu_{m,n}^{0,0}(t_b)$, $\mu_{m,n}^{0,1}(t_b)$, $\mu_{m,n}^{1,0}(t_b)$ and $\mu_{m,n}^{1,1}(t_b)$.

$$\mu_{m,n}^{0,0}(t_b) = \int_{t_b}^{\infty} f_T(s)[\mu_{m,n}^{0,0}(0) + t_b]ds + \int_0^{t_b} f_T(s)[\mu_{m-1,n}^{0,0}(t_b - s) + s]\mathsf{P}\{T = W\}ds + \int_0^{t_b} f_T(s)[\mu_{m,n-1}^{0,0}(t_b - s) + s]\mathsf{P}\{T = X\}ds + \int_0^{t_b} f_T(s)[\mu_{m,n}^{0,1}(t_b - s) + s]\mathsf{P}\{T = Y\}ds + \int_0^{t_b} f_T(s)[\mu_{m,n}^{1,0}(t_b - s) + s]\mathsf{P}\{T = Z\}ds. \quad (4)$$

In a similar way, recursive equations can be obtained for the queues corresponding to other knowledge states. This would yield (not shown here for space constraints) similar equations for $\mu_{m,n}^{0,1}(t_b)$, etc. Moreover, simple calculations give

$$\mathsf{P}\{T = W\} = \frac{\lambda_{D1}}{\lambda}, \quad \mathsf{P}\{T = X\} = \frac{\lambda_{D2}}{\lambda}, \quad \mathsf{P}\{T = Y\} = \frac{\lambda_{21}}{\lambda}, \quad \mathsf{P}\{T = Z\} = \frac{\lambda_{12}}{\lambda}. \quad (5)$$

These integral equations can be further simplified by converting them into differential equations (we will not show the details here). Finally, we arrive at a set of four coupled difference-differential equations which completely defines the queuing dynamics of our distributed system. It is intuitively clear that while solving each of these equations, we need to solve for their corresponding initial conditions, i.e. $\mu_{m,n}^{0,0}(0)$, $\mu_{m,n}^{0,1}(0)$, $\mu_{m,n}^{1,0}(0)$ and $\mu_{m,n}^{1,1}(0)$, which are determined according to the load-balancing algorithm as defined in Section II. In summary, we outlined a formalism that allows us to compute the average total completion time.

VI. Conclusions and Acknowledgements

We have performed experiments and simulations to investigate the performance of a load balancing policy that involves redistributing the load of the nodes only once after a large load arrives at the distributed system. Our experimental results (using a wireless LAN) and simulations both indicate that in distributed systems where communication and load-transfer delays are tangible, it is best to execute the load balancing after each node receives communications from other nodes regarding their load states. In particular, our results indicate that the loss of time in waiting for the inter-node communications to arrive is overcompensated by the informed nature of the load balancing. Moreover, the optimal load-balancing gain turns out to be less than unity, contrary to systems that do not exhibit significant latency. In delay infested systems, a moderate balancing gain has the benefit of reduced load-transfer delays, as the fraction of the load to be transferred is reduced. This in turn, will result in a reduced likelihood of certain nodes becoming idle as soon as they are depleted of their initial load.

This work is supported by the National Science Foundation under Information Technology Research (ITR) grant No. ANI-0312611. Additional support was received from the National Science Foundation through grant No. INT-9818312.

VII. References

[1] C. Knessly and C. Tiery,"Two Tandem queues with general renewal input I: Diffusion approximation and integral representation,"*SIAM J. Appl. Math.*,vol.59, pp1917-1959, 1999.

[2] F. Bacelli and P. Bremaud, "Elements of Queuing Theory: Palm-Martingale Calculus and Stochastic Recurrence", New York: Springer-Verlag, 1994.

[3] Z. Lan, V. E. Taylor, and G. Bryan, "Dynamic load balancing for adaptive mesh refinement application," in *Proc. ICPP'2001*, Valencia, Spain, 2001.

[4] 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.

[5] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *IEEE Trans. Parallel and Distributed Computing*, vol. 7, pp. 279–301, Oct. 1989

[6] C-C. Hui and S. T. Chanson,"Hydrodynamic load balancing," *IEEE Trans. Parallel and Distributed Systems*, vol. 10, Issue 11, Nov. 1999.

[7] M. M. Hayat, S. Dhakal, C. T. Abdallah " Dynamic time delay models for load balancing. Part II: Stochastic analysis of the effect of delay uncertainty, *CNRS-NSF Workshop: Advances in Control of Time-Delay Systems*, Paris France, January 2003. Also to appear in an edited book by Springer, Keqin Gu and Silviu-Iulian Niculescu, Editors.

[8] S. Dhakal, B.S. Paskaleva, M.M. Hayat, E. Schamiloglu, C.T. Abdallah "Dynamical Discrete-Time Load Balancing in Distributed Systems in the Presence of Time Delays," *IEEE CDC 2003*, Maui, Hawaii.

[9] 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.

[10] D. J. Daley and D. Vere-Jones, *An introduction to the theory of point processes*. Springer-Verlag, 1988.