6-1-2004

On Load balancing in distributed systems with large time delays: Theory and experiment

Chaouki T. Abdallah

J. Ghanem

S. Dhakal

M. M. Hayat

H. Jerez

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

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

Electrical & Computer Engineering Department
MSC01 1100
1 University of New Mexico
Albuquerque, NM 87131-1356
USA
{jean,sdhakal,chaouki,hayat,hjerez}@eece.unm.edu

September 27, 2003

Abstract

In a distributed computing environment with a high communication cost, limiting the number of balancing instants results in a better performance than 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 over the load-balancing gain becomes an important problem. In this paper we discuss the performance of a previously reported, control-theoretic motivated single load-balancing strategy on a distributed **physical** system and the performance is compared to our simulation predictions. Based on the concept of regeneration, we also present a mathematical model for the distributed system with two nodes where a one-shot balancing is done. We obtain a system of four difference-differential equations characterizing the mean of the overall completion time. and compare its predictive capabilities via simulation to the physical system.

1 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 large-scale distributed computing systems in which the CEs are either geographically distant from each other, or are connected via unreliable wireless channels, there is an inherent delay in the inter-node communications and transfer of loads. Moreover, in systems using a shared communication medium (such as the Internet or a wireless LAN), such delays vary according to the size of the loads to be transferred and also fluctuate according to the random condition of the communication medium that connects the CEs.

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 [5, 3, 6]. 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 by the authors and their collaborators in [8, 9, 10]. Our approach to the load-balancing problem is rooted in control-theoretic concepts, and is based on dynamical models, which has allowed us to apply powerful stability analysis techniques. A number of load-balancing strategies have been proposed, and it is almost universally observed that the types of delay described above degrade the overall performance [3, 5, 6, 7]. 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. Recently, our recent Monte-Carlo studies [8, 9] 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 [9] 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 particular, a novel concept of information states of the current queue size is introduced to handle the complexity of the queuing model. The system model is developed for the special case of two geographically distant CEs.

The paper is organized as follows: Section 2 contains a description of the load balancing policy. The experimental results, on a physical wireless 3-node network are given in section 3, while simulation results are presented in 4. The stochastic analysis of our system is detailed in section 5, while our conclusions are given in 6.

2 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 [8, 11]. 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 [12], that is, $J_i(t_1, t_2) = \sum_{k:t_1 \leq \xi_k < t_2} H_k$, where ξ_k are arrival times of job requests arriving according to a Poisson process with rate λ_i and the random sequence $H_k, k = 1, 2 \ldots$, is a sequence of integer-valued random variables describing the number of tasks associated with the kth job request.) The load balancing mechanism considered in this paper is described as follows: The ith node, at a 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 load to the other nodes according to a deterministic policy. Moreover, at the time when it is not balancing its load, it may receive loads from the neighboring nodes subject to random delays (due to the load-transfer delays). In all the examples considered in this paper, only one load-balancing execution is permitted. That is, $T_l^i = \infty$, for all $l \geq 2$ and all $i \geq 1$.

With the above description of task assignments between nodes, we can write the dynamics of the ith queue in a differential form as (in Δt time increments):

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

where $C_i(t + \Delta t)$ is a Poisson process (with rate μ_i) describing the random number of tasks completed in the interval $[t, t + \Delta t)$, $J_i(t, t + \triangle t)$ is a random number of new, external tasks arriving in the same interval, $\tau_{ij}(t)$ is the delay in transferring load from node j to node i at the same interval, and $L_{ji}(t)$ is the load transferred from node i to j in the interval $[t, t + \Delta t)$. More precisely, for any $k \neq l$, the random load L_{kl} diverted from

node l to node k has the form $L_{kl}(t) \triangleq g_{kl}(Q_l(t), Q_k(t-\eta_{lk}(t)), \ldots, Q_j(t-\eta_{lj}(t)), \ldots)$, where for any $j \neq k$, $\eta_{kj}(t) = \eta_{jk}(t)$ is the communication delay between the kth and jth nodes at time t. The function g_{kl} dictates the load-balancing policy between the kth and lth nodes. One common example is

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

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. In this example, the lth node simply compares its load to the average over all nodes and sends out a fraction p_{lk} of its excess load to the lth node. (Of course, $\sum_{l \neq k} p_{lk} = 1$.)

3 Experimental Results

We have developed various in-house testbeds to study the effects of the gain parameter k as well as the selection of the load-balancing instant. The details of one of these systems are described below.

3.1 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. 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.

In the first set of experiments, the gain parameter k was set 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 assumed 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, and 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.

3.2 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 $45 \sim 50$ s) 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 40 s) 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. 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.

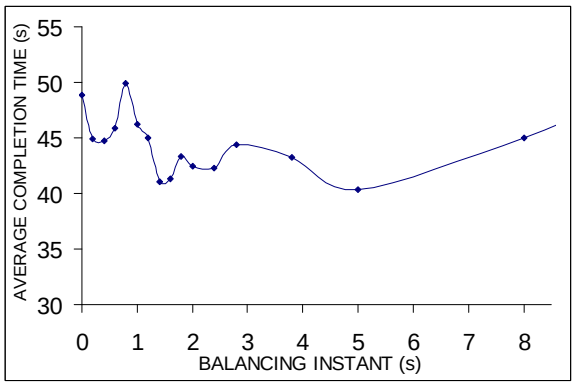

Figure 1: Average total task-completion time as a function of the load-balancing instant. The load-balancing gain parameter is set at $k = 1$. The dots represent the actual experimental values and the solid curve is a best polynomial fit. This convention is used thought out Fig. 4

.

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 another imbalance situation, which, in turn, leads to suboptimal total task completion times, as confirmed by Fig. 1.

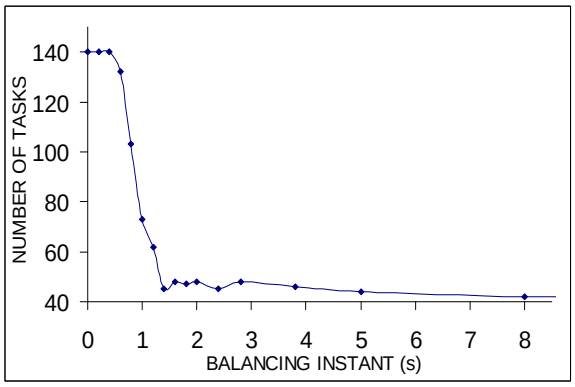

Figure 2: Average total excess load decided by the load-balancing policy to be transferred (at the load-balancing instant) as a function of the balancing instant. The load-balancing gain parameter is set at $k = 1$.

The results of the second set of experiments indeed confirm our earlier prediction (as reported in [9]) that when communication and load-transfer delays are prevalent, the load-balancing gain must be reduced to prevent "overreaction." 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 time, as confirmed earlier by Fig. 3.

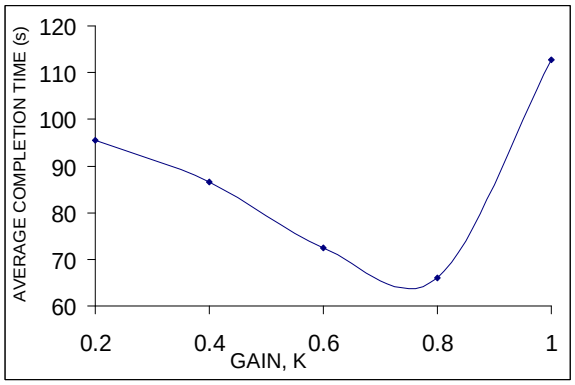

Figure 3: Average total task-completion time as a function of the balancing gain. The load-balancing instant is fixed at 1.4 s.

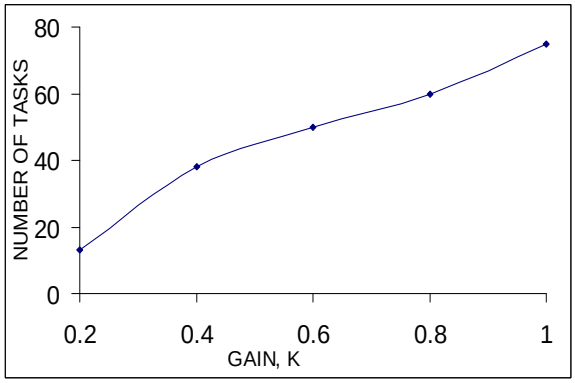

Figure 4: Average total excess load decided by the load-balancing policy to be transferred (at the load-balancing instant) as a function of the balancing gain. The load-balancing instant is fixed at 1.4 s.

4 Simulation Results

We have generated a Monte-Carlo simulation tool that allows the simulation of the queues described Section 2. We used this tool to validate the correspondence between the stochastic queuing model and the experimental setup. In particular, we have generated the simulated versions of Figures 1 through 3, which are shown below. It is observed that the general characteristics of the curves are very similar.

5 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 minimize the load-balancing gain to minimize the average completion time, we will outline a novel regenerative approach that will fit the queuing model described in Section 2. The concept of regeneration has proven to be a powerful tool in the analysis of complex stochastic systems [1, 2, 12]. The analysis presented here is not fundamentally limited to the choice of a particular balancing policy and the idea of the exposition is to show viability of the

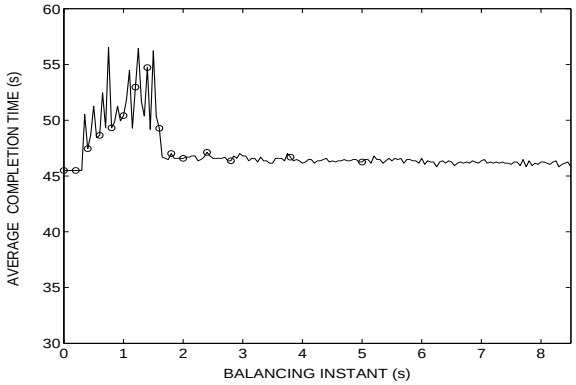

Figure 5: Simulation results for the average total task-completion time as a function of the load-balancing instant. The load-balancing gain parameter is set at $k = 1$. The dots represent the actual experimental values and the solid curve is a best polynomial fit. This convention is used thought out Fig. 4

.

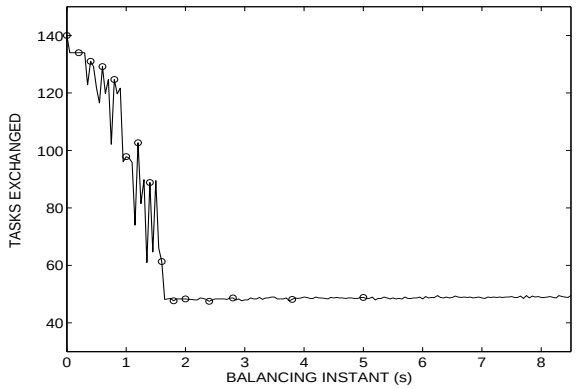

Figure 6: Simulation results for the average total excess load decided by the load-balancing policy to be transferred (at the load-balancing instant) as a function of the balancing instant. The load-balancing gain parameter is set at $k = 1$.

approach in analyzing the complex queuing model involved. 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 such that to minimize it. Here, we discuss the behavior of the zero-input response of the queues and hence do not have any task arrival at any of the nodes. The purpose of this exposition is to introduce our regenerative approach. Actual calculations to find the optimal load-balancing instant and the optimal load-balancing gain (which would minimize the average total completion time) will be reported in a future correspondence.

5.1 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, but they will 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.

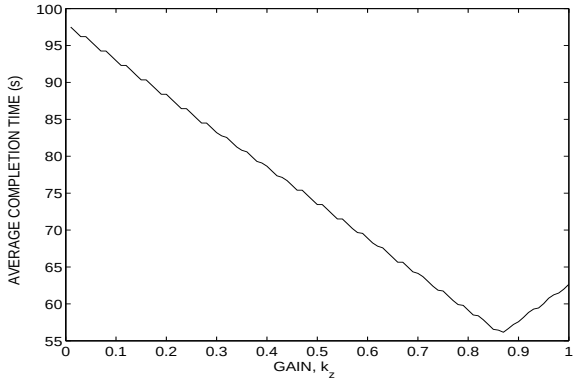

Figure 7: Simulation results for the average total task-completion time as a function of the balancing gain. The load-balancing instant is fixed at 1.4 s.

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. Therefore, we need $n(n-1)$ number bit sequences to adequately define all possible knowledge states of the whole knowledge distributed system, and hence there will be a total of $2^{n(n-1)}$ number of knowledge states. Clearly, the average overall completion time depends on the knowledge state of the system at the time of balancing. 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.

5.2 Dynamic Model Base

For 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 parameter 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) then the probability density function (pdf) of T can be characterized as $f_T(t) = \lambda e^{-\lambda t}u(t)$, where $\lambda = \lambda_{D1} + \lambda_{D2} + \lambda_{21} + \lambda_{12}$, and $u(\cdot)$ is the unit step function.

Now let $\mu_{m,n}^{k1,k2}(t_b)$ be the estimate 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 $m-1$. 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. Our objective is to find $\mu_{m,n}^{0,0}(t_b)$. 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 $\mu_{m,n}^{1,1}(t_b)$.

$$\mu_{m,n}^{0,0}(t_b) = \int_{t_b}^{\infty} f_T(s)[\mu_{m,n}^{0,0}(0) + t_b]ds +$$

$$\int_0^{t_b} f_T(s)[\mu^{0,0}_{m-1,n}(t_b-s)+s].P[T=W]ds +$$
$$\int_0^{t_b} f_T(s)[\mu^{0,0}_{m,n-1}(t_b-s)+s].P[T=X]ds +$$
$$\int_0^{t_b} f_T(s)[\mu^{0,1}_{m,n}(t_b-s)+s].P[T=Y]ds +$$
$$\int_0^{t_b} f_T(s)[\mu^{1,0}_{m,n}(t_b-s)+s].P[T=Z]ds. \quad (3)$$

In a similar way, recursive equations can be obtained for the queues corresponding to other knowledge states. For example, for $\mu^{0,1}_{m,n}(t_b)$, we have

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

The probabilities $P[T=W]$, and those alike, which appear in the above recursive equations can be evaluated directly using elementary probability. In particular,

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

These integral equations can be simplified by converting them into differential equations of standard form. For example, by differentiating each of these equations with respect to t_b, we get four differential- difference equations. For the case of $\mu^{0,0}_{m,n}(t_b)$, we have

$$\frac{\partial \mu^{0,0}_{m,n}(t_b)}{\partial t_b} = \lambda_{D1}\mu^{0,0}_{m-1,n}(t_b) + \lambda_{D2}\mu^{0,0}_{m,n-1}(t_b) +$$
$$\lambda_{21}\mu^{0,1}_{m,n}(t_b) + \lambda_{12}\mu^{1,0}_{m,n}(t_b) - \lambda\mu^{0,0}_{m,n}(t_b) + 1 \quad (6)$$

and from Eq. (4)

$$\frac{\partial \mu^{0,1}_{m,n}(t_b)}{\partial t_b} = \lambda_{D1}\mu^{0,1}_{m-1,n}(t_b) + \lambda_{D2}\mu^{0,1}_{m,n-1}(t_b) +$$
$$\lambda_{21}\mu^{0,1}_{m,n}(t_b) + \lambda_{12}\mu^{1,1}_{m,n}(t_b) - \lambda\mu^{0,1}_{m,n}(t_b) + 1 \quad (7)$$

It is intuitively clear that while solving each of these equations, we need to solve for their corresponding initial conditions,i.e.$\mu^{0,0}_{m,n}(0)$, $\mu^{0,1}_{m,n}(0)$, $\mu^{1,0}_{m,n}(0)$ and $\mu^{1,1}_{m,n}(0)$, which are determined according to the load-balancing algorithm.

In summary, we outlined a formalism that allows us to compute the average total completion time. This can be used to find the optimal balancing instant for a particular load-balancing policy. We are presently working to explicitly compute the initial condition for the balancing strategy which is Section 2.

6 Conclusions

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.

It should be observed that while load-balancing is the main thrust of this work, the methodologies developed here are applicable to various distributed computing and network control applications. We are in the process in doing so, as will be reported elsewhere. In addition, and while the experimental results reported here use a wireless network, similar experiments were conducted to illustrate the usefulness of our methodology for Internet-based, but geographically distant computing.

7 Acknowledgements

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.

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] Zhou, Utopia,"A Load Sharing Facility for Large, Heterogenous Distributed Computer Systems : Practice and Experience." TRCSRI-257, Toronto, 1992

[5] 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.

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

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

[8] 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.

[9] 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,"

[10] J. D. Bridwell, J. Chisson, 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. Also to appear in an edited book by Springer, K. Gu and S-I. Niculescu, Editors.

[11] 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.

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