Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

The Effect of Time Delays in the Stability of Load Balancing Algorithms for Parallel Computations

J. D. Birdwell, John Chiasson, C. T. Abdallah, Zhong Tang, Nivedita Alluri and Tsewei Wang

Abstract— **Deterministic dynamic nonlinear time-delay systems are developed to model load balancing in a cluster of computer nodes used for parallel computations. The model is shown to be self consistent in that the queue lengths cannot go negative and the total number of tasks in all the queues are conserved (i.e., load balancing can neither create nor lose tasks). Further, it is shown that using the proposed load balancing algorithms, the system is stable. Experimental results are presented and compared with the predicted results from the analytical model. In particular, simulations of the models are compared with an experimental implementation of the load balancing algorithm on a parallel computer network.**

Keywords— **Load balancing, Computer Networks, Time Delay Systems.**

I. Introduction

Parallel computer architectures utilize a set of computational elements (CE) to achieve performance that is not attainable on a single processor, or CE, computer. A common architecture is the cluster of otherwise independent computers communicating through a shared network. To make use of parallel computing resources, problems must be broken down into smaller units that can be solved individually by each CE while exchanging information with CEs solving other problems. For a background on mathematical treatments of load balancing, the reader is referred to [1][2][3].

The present work focuses upon the effects of delays in the exchange of information among CEs, and the constraints these effects impose on the design of a load balancing strategy. Previous results by the authors appear in [4][5]. However, new nonlinear models are developed here. Specifically, a deterministic dynamic nonlinear time-delay system is developed to model load balancing. The model is shown to be self consistent in that the queue lengths cannot go negative and the total number of tasks in all the queues and the network are conserved (i.e., load balancing can neither create nor lose tasks). Simulations of the nonlinear model are then compared with an *experimental implementation* of the load balancing algorithm on a parallel computer network.

Section 2 presents our approach to modeling the computer network and load balancing algorithms to incorporate the presence of delay in communicating between nodes and transferring tasks. Further, it is shown that the model correctly models the nonnegativity of he queue lengths and that the total number of tasks in all the queues and in transit is conserved by the load balancing algorithm. This section ends with a proof of stability of the model for the chosen load balancing algorithms (controllers). Section 3 presents simulations of the nonlinear models used for comparison with the actual experimental data. Section 4 presents *experimental data* from an implementation of the load balancing algorithm (controller). Finally, Section 5 is a summary and conclusion of the present work and a discussion of future work.

II. Models of Load Balancing Algorithms

In this section, continuous time models are developed to model load balancing among a network of computers. A basic model is described first to give the overall approach used here. This basic model is a nonlinear system with delay which is then simplified to obtain a linear time-invariant system with delay. Finally, the nonlinear model is modified so that the number of tasks a node distributes to the other nodes is based on their relative load levels.

To introduce the basic approach to load balancing, consider a computing network consisting of n computers (nodes) all of which can communicate with each other. At start up, the computers are assigned an equal number of tasks. However, when a node executes a particular task it can in turn generate more tasks so that very quickly the loads on various nodes become unequal. To balance the loads, each computer in the network sends its queue size $q_j(t)$ to all other computers in the network. A node i receives this information from node j *delayed* by a finite amount of time τ_{ij}; that is, it receives $q_j(t-\tau_{ij})$. Each node i then uses this information to compute its local estimate[1] of the average number of tasks in the queues of the n computers in the network. In this work, the simple estimator $\left(\sum_{j=1}^{n} q_j(t-\tau_{ij})\right)/n$ $(\tau_{ii}=0)$ which is based on the most recent observations is used. Node i then compares its queue size $q_i(t)$ with its estimate of the network average as

[1]It is an estimate because at any time, each node only has the delayed value of the number of tasks in the other nodes.

J. D. Birdwell, J. Chiasson, Z. Tang and Nivedita Alluri are with the ECE Dept, The University of Tennessee, Knoxville TN 37996, birdwell@utk.edu,chiasson@utk.edu,tang@hickory.engr.utk.edu

C. T. Abdallah is with the ECE Dept, University of New Mexico, Alburquerque NM 87131-1356, USA, chaouki@eece.unm.edu

T.W. Wang is with the Chem E Dept, The University of Tennessee, Knoxville TN 37996. twang@utk.edu

The work of the J.D. Birdwell, J. Chiasson and C.T. Abdallah was supported in part by the National Science Foundation. The work of J.D. Birdwell, T.W. Wang and Z. Tang 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 work of C.T. Abdallah was supported in part by the National Science Foundation through grant INT-9818312. 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.

0-7803-7924-1/03/$17.00 ©2003 IEEE

$\left(q_i(t) - \left(\sum_{j=1}^{n} q_j(t-\tau_{ij})\right)/n\right)$ and, if this is greater than zero, the node sends some of its tasks to the other nodes. If it is less than zero, no tasks are sent. Further, the tasks sent by node i are received by node j with a delay h_{ij}. The controller (load balancing algorithm) decides how often and fast to do load balancing (transfer tasks among the nodes) and how many tasks are to be sent to each node.

As just explained, each node controller (load balancing algorithm) has only *delayed* values of the queue lengths of the other nodes, and each transfer of data from one node to another is received only after a finite time delay. An important issue considered here is the effect of these delays on system performance. Specifically, the continuous time models developed here represent our effort to capture the effect of the delays in load balancing techniques and were developed so that system theoretic methods could be used to analyze them.

A. Basic Model

The basic mathematical model of a given computing node for load balancing is given by

$$\begin{aligned} \frac{dx_i(t)}{dt} &= \lambda_i - \mu_i + u_i(t) - \sum_{j=1}^{n} p_{ij}\frac{t_{p_i}}{t_{p_j}} u_j(t-h_{ij}) \\ y_i(t) &= x_i(t) - \frac{\sum_{j=1}^{n} x_j(t-\tau_{ij})}{n} \\ u_i(t) &= -K_i \text{sat}\,(y_i(t)) \\ p_{ij} &\geqslant 0, p_{jj} = 0, \sum_{i=1}^{n} p_{ij} = 1 \end{aligned} \tag{1}$$

where

$$\begin{aligned} \text{sat}\,(y) &= y \text{ if } y \geqslant 0 \\ &= 0 \text{ if } y < 0. \end{aligned}$$

In this model we have

- n is the number of nodes.
- $x_i(t)$ is the *expected waiting time* experienced by a task inserted into the queue of the i^{th} node. With $q_i(t)$ the number of *tasks* in the i^{th} node and t_{p_i} the average time needed to process a task on the i^{th} node, the expected (average) waiting time is then given by $x_i(t) = q_i(t)t_{p_i}$. Note that $x_j/t_{p_j} = q_j$ is the number of tasks in the node j queue. If these tasks were transferred to node i, then the waiting time transferred is $q_j t_{p_i} = x_j t_{p_i}/t_{p_j}$, so that the fraction t_{p_i}/t_{p_j} converts waiting time on node j to waiting time on node i.
- $\lambda_i \geq 0$ is the rate of generation of waiting time on the i^{th} node caused by the addition of tasks (rate of increase in x_i)
- $\mu_i \geq 0$ is the rate of reduction in waiting time caused by the service of tasks at the i^{th} node and is given by $\mu_i \equiv (1 \times t_{p_i})/t_{p_i} = 1$ for all i if $x_i(t) > 0$, while if $x_i(t) = 0$ then $\mu_i \triangleq 0$, that is, if there are no tasks in the queue, then the queue cannot possibly decrease.
- $u_i(t)$ is the rate of removal (transfer) of the tasks from node i at time t by the load balancing algorithm at node i. Note that $u_i(t) \leq 0$.
- $p_{ij}u_j(t)$ is the rate at which node j sends waiting time (tasks) to node i at time t where $p_{ij} \geqslant 0, \sum_{i=1}^{n} p_{ij} = 1$ and $p_{jj} = 0$. That is, the transfer from node j of expected waiting time (tasks) $\int_{t_1}^{t_2} u_j(t)dt$ in the interval of time $[t_1, t_2]$ to the other nodes is carried out with the i^{th} node receiving the fraction $p_{ij}\left(t_{p_i}/t_{p_j}\right)\int_{t_1}^{t_2} u_j(t)dt$ where the ratio t_{p_i}/t_{p_j} converts the task from waiting time on node j to waiting time on node i. As $\sum_{i=1}^{n}\left(p_{ij}\int_{t_1}^{t_2} u_j(t)dt\right) = \int_{t_1}^{t_2} u_j(t)dt$, this results in removing *all* of the waiting time $\int_{t_1}^{t_2} u_j(t)dt$ from node j.
- The quantity $-p_{ij}u_j(t-h_{ij})$ is the rate of increase (rate of transfer) of the expected waiting time (tasks) at time t from node j by (to) node i where h_{ij} ($h_{ii} = 0$) is the time delay for the task transfer from node j to node i.
- The quantities τ_{ij} ($\tau_{ii} = 0$) denote the time delay for communicating the expected waiting time x_j from node j to node i.
- The quantity $x_i^{avg} = \left(\sum_{j=1}^{n} x_j(t-\tau_{ij})\right)/n$ is the estimate[2] by the i^{th} node of the average waiting time of the network and is referred to as the *local average* (local estimate of the average).

In this model, all rates are in units of the *rate of change of expected waiting time*, or *time/time* which is dimensionless. As $u_i(t) \leq 0$, node i can only send tasks to other nodes and cannot initiate transfers from another node to itself. A delay is experienced by transmitted tasks before they are received at the other node. The control law $u_i(t) = -K_i\text{sat}(y_i(t))$ states that if the i^{th} node output $x_i(t)$ is above the local average $\left(\sum_{j=1}^{n} x_j(t-\tau_{ij})\right)/n$, then it sends data to the other nodes, while if it is less than the local average nothing is sent. The j^{th} node receives the fraction $\int_{t_1}^{t_2} p_{ji}\left(t_{p_i}/t_{p_j}\right)u_i(t)dt$ of transferred waiting time $\int_{t_1}^{t_2} u_i(t)dt$ delayed by the time h_{ij}.

Model (1) is the basic model, but one important detail remains unspecified, namely the exact form p_{ij} for each sending node i. One approach is to choose them as constant and equal

$$\begin{aligned} p_{ij} &= 1/(n-1) \text{ for } j \neq i \\ p_{ii} &= 0 \end{aligned} \tag{2}$$

where it is clear that $p_{ji} \geqslant 0, \sum_{i=1}^{n} p_{ij} = 1$.

B. Model Consistency

It is now shown that the model is consistent with actual working systems in that the queue lengths cannot go negative, and the load balancing algorithm cannot create or lose tasks; it can only move then between nodes.

[2]This is an only an estimate due to the delays.

B.1 Non Negativity of the Queue Lengths

To show the non negativity of the queue lengths, recall that the queue length of each node is given by $q_i(t) = x_i(t)/t_{p_i}$. The model is rewritten in terms of these quantities as

$$\frac{d}{dt}\left(x_i(t)/t_{p_i}\right) = \frac{\lambda_i - \mu_i}{t_{p_i}} + \frac{1}{t_{p_i}}u_i(t) - \sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t-h_{ij}). \tag{3}$$

Given that $x_i(0) > 0$ for all i, it follows from the right-hand side of (3) that $q_i(t) = x_i(t)/t_{p_i} \geq 0$ for all $t \geq 0$ and all i. To see this, suppose without loss of generality that $q_i(t) = x_i(t)/t_{p_i}$ is the first queue to go to zero, and let t_1 be the time when $x_i(t_1) = 0$. At the time t_1, $\lambda_i - \mu_i = \lambda_i \geq 0$ by the definition of μ_i and $-\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t-h_{ij}) \geq 0$ for all time by the definition of the u_j. Further, the term $u_i(t_1)$ is negative only if

$$x_i(t_1) > \left(\sum_{j=1}^{n} x_j(t_1 - \tau_{ij})\right)/n. \tag{4}$$

By supposition (up to time t_1) all the $x_j(t_1 - \tau_{ij}) > 0$ for $j \neq i$ and $x_i(t_1) = 0$ so that $u_i(t_1) = 0$ as the right side of (4) is positive at time t_1. Consequently, at time t_1 all terms on the right-hand side of (3) are non negative. Further, $x_i(t)$ cannot go negative in a neighborhood of t_1. For if it did, as the right-hand side of (4) is continuous, it follows that

$$x_i(t) < 0 < \left(\sum_{j=1}^{n} x_j(t - \tau_{ij})\right)/n \tag{5}$$

for some $t \in (t_1, t_1 + \delta)$ with $\delta > 0$. Therefore, $u_i(t) = 0$ for all $t \in [t_1, t_1 + \delta]$ and the right-hand side of (3) is non negative for all $t \in [t_1, t_1 + \delta]$ which contradicts $x_i(t) < 0$. Note that $t_1 + \delta$ can be taken to be at least as large as the time at which some x_k goes to zero, that is, $q_k(t_1 + \delta) = 0$ as the right hand side of (5) must remain positive for $t \in [t_1, t_1 + \delta]$.

If $x_i(t)$ goes positive after t_1, then the above argument is repeated at the next time a queue goes to zero. If $x_i(t)$ remains identically zero in the interval $(t_1, t_1 + \delta)$, then the argument is also similar in that at time $t_1 + \delta$, both $x_i(t_1 + \delta), x_k(t_1 + \delta)$ are then zero. As the remaining $n-2$ nodes are still positive, the right-hand side of equation (5) continues to hold with both x_i and x_k zero at time $t_1 + \delta$ and one again gets a contradiction if either x_i or x_k goes negative in an interval $(t_1 + \delta, t_2)$. Continuing in this manner, it follows that $q_i(t) = x_i(t)/t_{p_i}$ cannot go negative for all i.

B.2 Conservation of Queue Lengths

It is now shown that the total number of tasks in all the queues and the network are conserved. To do so, sum up equations (3) from $i = 1, ..., n$ to get

$$\begin{aligned}\frac{d}{dt}\left(\sum_{i=1}^{n} q_i(t)\right) &= \sum_{i=1}^{n}\left(\frac{\lambda_i - \mu_i}{t_{p_i}}\right) + \sum_{i=1}^{n} u_i(t)/t_{p_i} \\ &\quad - \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t-h_{ij})\end{aligned} \tag{6}$$

which is the rate of change of the total queue lengths on all the nodes. However, the network itself also contains tasks in transit between nodes. The dynamic model of the queue lengths in the network is given by

$$\frac{d}{dt}q_{net_i}(t) = \sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t-h_{ij}) - \sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t). \tag{7}$$

Here q_{net_i} is the number of tasks put on the network that are being sent to node i. This equation simply says that the j^{th} node is putting tasks on the network to be sent to node i at the rate $\frac{p_{ij}}{t_{p_j}}u_j(t)$ while the i^{th} node is taking these tasks from node j off the network at the rate $\frac{p_{ij}}{t_{p_j}}u_j(t-h_{ij})$. Summing (7) over all the nodes, one obtains

$$\begin{aligned}\frac{d}{dt}\left(\sum_{i=1}^{n} q_{net_i}(t)\right) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t-h_{ij}) - \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}u_j(t-h_{ij}) - \sum_{j=1}^{n}\frac{u_j(t)}{t_{p_j}}.\end{aligned} \tag{8}$$

Adding (6) and (8), one obtains the conservation of queue lengths given by

$$\frac{d}{dt}\sum_{i=1}^{n}\left(q_i(t) + q_{net_i}(t)\right) = \sum_{i=1}^{n}\left(\frac{\lambda_i - \mu_i}{t_{p_i}}\right). \tag{9}$$

In words, the total number of tasks which are in the system (i.e., in the nodes and/or in the network) can increase only by the rate of arrival of tasks $\sum_{i=1}^{n}\lambda_i/t_{p_i}$ at all the nodes, or similarly, decrease by the rate of processing of tasks $\sum_{i=1}^{n}\mu_i/t_{p_i}$ at all the nodes. The load balancing itself cannot increase or decrease the total number of tasks in all the queues.

B.3 Stability of the Model

Combining the results of the previous two subsections, one can show stability of the model. Specifically, we have

Theorem: Given the system described by (1) and (7) with $\lambda_i = 0$ for $i = 1, ..., n$ and initial conditions $x_i(0) \geq 0$, then $(q_i(t), q_{net_i}(t)) \to 0$ as $t \to \infty$.

Proof: First note that the q_{net_i} are non negative as

$$q_{net_i}(t) = -\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}\left(\int_{t-h_{ij}}^{t} u_j(\tau)d\tau\right) \geq 0. \tag{10}$$

Under the conditions of the theorem, equation (9) becomes

$$\frac{d}{dt}\sum_{i=1}^{n}\left(q_i(t) + q_{net_i}(t)\right) = -\sum_{i=1}^{n}\frac{\mu_i}{t_{p_i}}. \tag{11}$$

Let $V(t) \triangleq \sum_{i=1}^{n} (q_i(t) + q_{net_i}(t))$, and, as the $q_i(t), q_{net_i}(t)$ are non negative, $V(t) \geq 0$ and is equal to zero if and only if $q_i(t) = 0, q_{net_i}(t) = 0$. (This is a time delay system so $(q_i(t), q_{net_i}(t))$ is *not* the state of the system [6]). Further, as $\mu_i(q_i(t)) = 1$ for $q_i(t) > 0$ and $\mu_i(q_i(t)) = 0$ if only if $q_i(t) = 0$, it follows that $dV/dt = -\sum_{i=1}^{n} \mu_i(q_i(t))/t_{p_i} \leq 0$. This then implies that

$$V(t) = V(0) - \int_0^t \sum_{i=1}^{n} \frac{\mu_i(q_i(t))}{t_{p_i}} dt \geq 0 \qquad (12)$$

is monotonically decreasing. As $V(t)$ is bounded below, we have $V(t) \downarrow V_f \geq 0$, or

$$\lim_{t\to\infty} \int_0^t \sum_{i=1}^{n} \frac{\mu_i}{t_{p_i}} dt = V(0) - V_f \geq 0. \qquad (13)$$

The quantity $\mu_i(q_i(t))$ is either 1 or 0 depending on whether $q_i(t)$ is positive or zero, so $\mu_i(q_i(t))$ can be viewed as a set of pulses of unit height and varying width. The integral $\int_0^\infty \mu_i(q_i(t))dt$ is finite by (13) which implies that the widths of the unit-height pulses making up $\mu_i(q_i(t))$ must go to zero as $t \to \infty$. So, even if a $q_i(t)$ $(= x_i(t)/t_{p_i})$ continues to switch between zero and positive non zero, the time intervals for which it is non zero must go to zero as $t \to \infty$. Summarizing, the $q_i(t)$ are non negative, continuous functions, bounded by the non negative monotonically decreasing function $V(t)$, and the intervals for which the $q_i(t)$ are non zero goes to zero as $t \to \infty$. Further, as

$$\begin{aligned} u_i(t) &= -K_i \text{sat}\left(x_i(t) - \frac{\sum_{j=1}^{n} x_j(t-\tau_{ij})}{n} \right) \\ &= -K_i \text{sat}\left(t_{p_i} q_i(t) - \frac{\sum_{j=1}^{n} \left(t_{p_i}/t_{p_j}\right) q_j(t-\tau_{ij})}{n} \right) \end{aligned}$$

it follows that the time intervals for which the *bounded* functions $u_i(t)$ are non zero must go to zero as $t \to \infty$. Consequently, by equation (10), $q_{net_i}(t) \to 0$ as $t \to \infty$.

We now show that the monotonically decreasing function $V(t)$ must go to zero, that is, $\lim_{t\to\infty} V(t) = V_f = 0$. Suppose not, so that $V_f > 0$. As $q_{net_i}(t) \to 0$, choose t_1 large enough so that $0 \leq \sum_{i=1}^{n} q_{net_i}(t) < \epsilon V_f$ for $t > t_1$ where $0 < \epsilon < 1$. Then

$$V(t) = \sum_{i=1}^{n} \Big(q_i(t) + q_{net_i}(t) \Big) \geq V_f \text{ for all } t$$

and

$$\sum_{i=1}^{n} q_i(t) \geq (1 - \epsilon) V_f > 0 \text{ for } t > t_1.$$

This in turn implies that at least one of the $q_i(t) > 0$ for all $t > t_1$ and therefore $\sum_{i=1}^{n} \mu_i(q_i(t))/t_{p_i} dt \geq \min\{1/t_{p_i}\}$ for all $t > t_1$. By equation (11), we then have

$$V(t) = V(t_1) - \int_{t_1}^{t} \sum_{i=1}^{n} \frac{\mu_i(q_i(t))}{t_{p_i}} dt \leq V(t_1) - \int_{t_1}^{t} \min\{\frac{1}{t_{p_i}}\} dt \qquad (14)$$

As the right side of (14) eventually becomes negative, we have a contradiction and therefore $V_f = 0$. As it has already been shown that $q_{net_i} \to 0$ for all i, $V(t) \to 0$ then implies that the non negative functions $q_i(t) \to 0$ for all i.

III. Simulations

Experimental procedures to determine the delay values are given in [7] and summarized in [8]. These give representative values for a Fast Ethernet network with three nodes of $\tau_{ij} = \tau = 200\ \mu$sec for $i \neq j, \tau_{ii} = 0$, and $h_{ij} = 2\tau = 400\ \mu$sec for $i \neq j, h_{ii} = 0$. The initial conditions were $x_1(0) = 0.6$, $x_2(0) = 0.4$ and $x_3(0) = 0.2$. The inputs were set as $\lambda_1 = 3\mu_1, \lambda_2 = 0, \lambda_3 = 0, \mu_1 = \mu_2 = \mu_3 = 1$. The t_{p_i}'s were taken to be equal.

In this set of simulations, the model (1) is used. Figures 1 and 2 show the responses with the gains set as $K = 1000$ and $K = 5000$. To compare with the experimental results given in Figure 4, Figure 3 are the output responses with the gains set as $K_1 = 6667, K_2 = 4167, K_3 = 5000$, respectively.

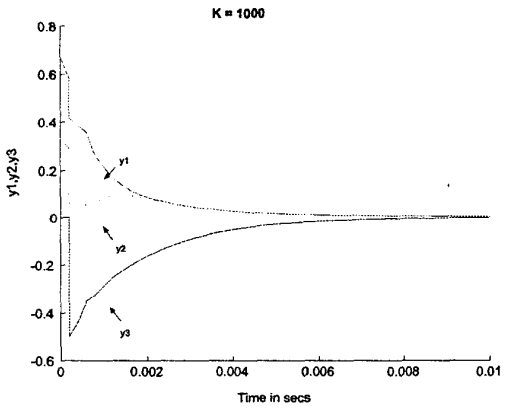

Fig. 1. Constant p_{ij} nonlinear output responses with $K = 1000$.

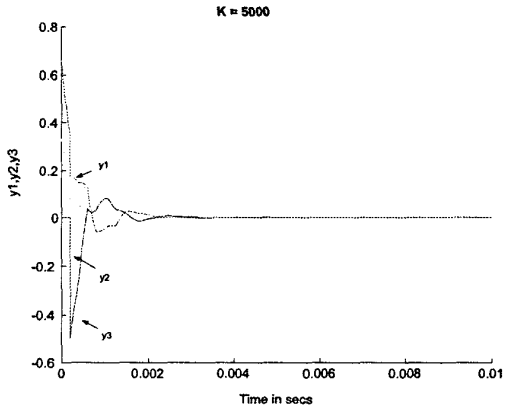

Fig. 2. Constant p_{ij} nonlinear output responses with $K = 5000$.

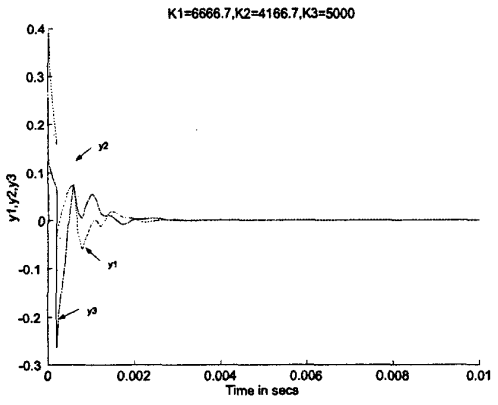

Fig. 3. Nonlinear simulation with constant p_{ij} and $K1 = 6666.7; K2 = 4166.7; K3 = 5000$

IV. Experimental Results

A parallel machine has been built to implement an experimental facility for evaluation of load balancing strategies. A root node communicates with k groups of computer networks. Each of these groups is composed of n nodes (hosts) holding identical copies of a portion of the database. (Any pair of groups correspond to different databases, which are not necessarily disjoint. A specific record, or DNA profile, is in general stored in two groups for redundancy to protect against failure of a node.) Within each node, there are either one or two processors. In the experimental facility, the dual processor machines use 1.6 GHz Athlon MP processors, and the single processor machines use 1.33 GHz Athlon processors. All run the Linux operating system. Our interest here is in the load balancing in any one group of n nodes/hosts.

The database is implemented as a set of queues with associated search engine threads, typically assigned one per node of the parallel machine. The search requests are created not only by the database clients; the search process also creates search requests as the index tree is descended by any search thread. This creates the opportunity for parallelism; search requests that await processing may be placed in any queue associated with a search engine, and the contents of these queues may be moved arbitrarily among the processing nodes of a group to achieve a balance of the load.

An important point is that the actual delays experienced by the network traffic in the parallel machine are *random*. Work has been performed to characterize the bandwidth and delay on unloaded and loaded network switches, in order to identify the delay parameters of the analytic models and is reported in [7][8]. The value $\tau = 200$ μsec used for simulations represents an average value for the delay and was found using the procedure described in [8]. The interest here is to compare the experimental data with that from the three models previously developed.

To explain the connection between the control gain K and the actual implementation, recall that the waiting time is related to the number of tasks as $x_i(t) = q_i(t)t_{p_i}$ where t_{p_i} is the average time to carry out a task. The continuous time control law is

$$u(t) = -K\text{sat}\left(y_i(t)\right)$$

where $u(t)$ is the rate of decrease of waiting time $x_i(t)$ per unit time. Consequently, the gain K represents the rate of reduction of waiting time per second in the continuous time model. Also, $y_i(t) = \left(q_i(t) - \left(\sum_{j=1}^{n} q_j(t-\tau_{ij})\right)/n\right) t_{p_i} = r_i(t)t_{p_i}$ where $r_i(t)$ is simply the number of tasks above the estimated (local) average number of tasks and, as the interest here is the case $y_i(t) > 0$, consider $u(t) = -Ky_i(t)$. With Δt the time interval between successive executions of the load balancing algorithm, the control law says that a fraction of the queue $K_z r_i(t)$ $(0 < K_z < 1)$ is removed in the time Δt so the rate of reduction of *waiting time* is $-K_z r_i(t)t_{p_i}/\Delta t = -K_z y_i(t)/\Delta t$ so that

$$u(t) = -\frac{K_z y_i(t)}{\Delta t} \Longrightarrow K = \frac{K_z}{\Delta t}. \tag{15}$$

This shows that the gain K is related to the actual implementation by how fast the load balancing can be carried out and how much (fraction) of the load is transferred. In the experimental work reported here, Δt actually varies each time the load is balanced. As a consequence, the value of Δt used in (15) is an average value for that run. The average time t_{p_i} to process a task is the same on all nodes (identical processors) and is equal 10μsec while the time it takes to ready a load for transfer is about 5μsec. The initial conditions were taken as $q_1(0) = 6000, q_2(0) = 4000, q_3(0) = 2000$ (corresponding to $x_1(0) = q_1(0)t_{pi} = 0.06, x_2(0) = 0.04, x_3(0) = 0.02$). All of the experimental responses were carried out with constant $p_{ij} = 1/2$ for $i \neq j$.

Figure 4 is a plot of the responses $r_i(t) = q_i(t) - \left(\sum_{j=1}^{n} q_j(t-\tau_{ij})\right)/n$ for $i = 1, 2, 3$ (recall that $y_i(t) = r_i(t)t_{p_i}$). The (average) value of the gains were $(K_z = 0.5)$ $K_1 = 0.5/75\mu\text{sec} = 6667, K_2 = 0.5/120\mu\text{sec} = 4167, K_3 = 0.5/100\mu\text{sec} = 5000$. This figure compares favorably with Figure 3 except for the time scale being off; that is, the experimental responses are slower. The explanation for this it that the gains here vary during the run because Δt (the time interval between successive executions of the load balancing algorithm) varies during the run. Further, this time Δt is *not* modeled in the continuous time simulations, only its average effect is represented in the gains K_i. That is, the continuous time model does not stop processing jobs (at the average rate t_{p_i}) while it is transferring tasks to do the load balancing.

Figure 5 shows the plots of the response for the (average) value of the gains given by $(K_z = 0.2)$ $K_1 = 0.2/125\mu\text{sec} = 1600, K_2 = 0.2/80\mu\text{sec} = 2500, K_3 = 0.2/70\mu\text{sec} = 2857$. The initial conditions were $q_1(0) = 6000, q_2(0) = 4000, q_3(0) = 2000$ $(x_1(0) = q_1(0)t_{pi} = 0.06, x_2(0) = 0.04, x_3(0) = 0.02)$. Figure 6 shows the plots of the response for the (average) value of the gains

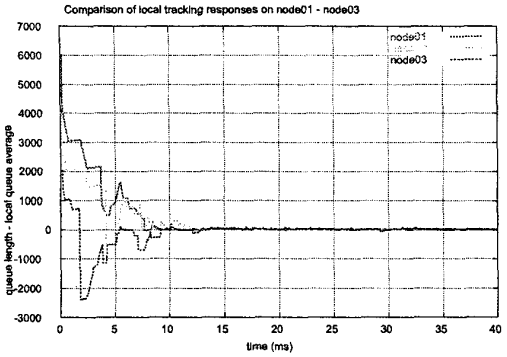

Fig. 4. Experimental response of the load balancing algorithm. The average value of the gains are $(K_z = 0.5)$ $K_1 = 6667, K_2 = 4167, K_3 = 5000$ with constant p_{ij}.

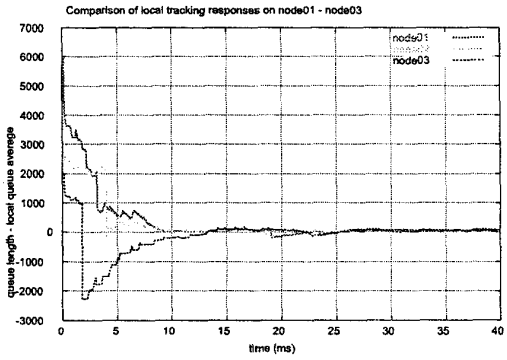

Fig. 5. Experimental response of the load balancing algorithm. The average value of the gains are $(K_z = 0.2)$ $K_1 = 1600, K_2 = 2500, K_3 = 2857$ with constant p_{ij}.

given by $(K_z = 0.3)$ $K_1 = 0.3/125\mu\text{sec} = 2400, K_2 = 0.3/110\mu\text{sec} = 7273, K_3 = 0.3/120\mu\text{sec} = 2500$.

V. Summary and Conclusions

A load balancing algorithm was modeled as a nonlinear time-delay system. The model was shown to be consistent in that the total number of tasks was conserved and the queues were always non negative. Further, the system was shown to be always stable, but the delays do create a limit on the size of the controller gains in order to ensure performance (fast enough response without oscillatory behavior). Experiments indicated a correlation of the continuous time models with the actual implementation. Future work will consider the fact that the load balancing operation involves processor time which is not being used to process tasks. There is a trade-off between using processor time/network bandwidth and the advantage of distributing the load evenly between the nodes to reduce overall processing time.

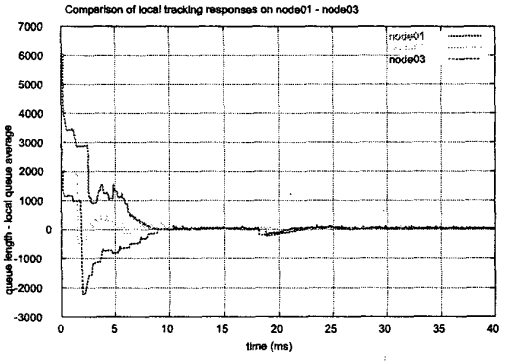

Fig. 6. Experimental response of the load balancing algorithm. The average value of the gains are $(K_z = 0.3)$ $K_1 = 2400, K_2 = 7273, K_3 = 2500$ with constant p_{ij}.

References

[1] E. Altman and H. Kameda, "Equilibria for multiclass routing in mult-agent networks," in *Proceedings of the 2001 IEEE Conference on Decision and Control*, December 2001. Orlando, FL USA.

[2] C. K. Hisao Kameda, Jie Li and Y. Zhang, *Optimal Load Balancing in Distributed Computer Systems*. Springer, 1997. Great Britain.

[3] H. Kameda, I. R. El-Zoghdy Said Fathy, and J. Li, "A performance comparison of dynanmic versus static load balancing policies in a mainframe," in *Proceedings of the 2000 IEEE Conference on Decision and Control*, pp. 1415–1420, December 2000. Sydney, Australia.

[4] J. D. Birdwell, J. Chiasson, Z. Tang, C. T. Abdallah, M. Hayat, Z. Tang, and T. Wang, "Dynamic time delay models for load balancing Part I: Deterministic models," in *CNRS-NSF Workshop: Advances in Control of Time-Delay Systems, Paris France*, January 2003. Also, to appear in an edited book by Springer-Verlag, Keqin Gu and Silviu-Iulian Niculescu, editors.

[5] C. Abdallah, J. Birdwell, J. Chiasson, V. Churpryna, Z. Tang, and T. Wang, "Load balancing instabilities due to time delays in parallel computation," in *Proceedings of the 3rd IFAC Conference on Time Delay Systems*, December 2001. Sante Fe NM.

[6] J. Hale and S. V. Lunel, *Introduction to Functional Differential Equations*. Springer-Verlag, 1993.

[7] P. Dasgupta, *Performance Evaluation of Fast Ethernet, ATM and Myrinet under PVM, MS Thesis*. University of Tennesse, 2001.

[8] P. Dasgupta, J. D. Birdwell, and T. W. Wang, "Timing and congestion studies under PVM," in *Tenth SIAM Conference on Parallel Processing for Scientific Computation*, March 2001. Portsmouth, VA.