Stability and Efficiency of a Random Local Load Balancing Protocol

Aris Anagnostopoulos*
Dept. of Computer Science
Brown University
Providence, RI 02912
aris@cs.brown.edu

Adam Kirsch*
Dept. of Computer Science
Brown University
Providence, RI 02912
akirsch@cs.brown.edu

Eli Upfal*
Dept. of Computer Science
Brown University
Providence, RI 02912
eli@cs.brown.edu

Abstract

We study the long term (steady state) performance of a simple, randomized, local load balancing technique. We assume a system of n processors connected by an arbitrary network topology. Jobs are placed in the processors by a deterministic or randomized adversary. The adversary knows the current and past load distribution in the network and can use this information to place the new tasks in the processors. The adversary can put a number of new jobs in each processor, in each step, as long as the (expected) total number of new jobs arriving at a given step is bounded by λn. A node can execute one job per step, and also participate in one load balancing operation in which it can move tasks to a direct neighbor in the network. In the protocol we analyze here, a node equalizes its load with a random neighbor in the graph.

We first study the stability of a system running our load balancing protocol. Clearly, if $\lambda > 1$ the system cannot be stable. We show that for any $\lambda < 1$, and any connected network topology, the system is stable.

When the system is stable, the next performance parameter of interest is the waiting time of jobs. We develop high probability bounds and bounds on the expectation of the waiting time of jobs in terms of the network topology. In particular, if the network is an expander graph the expected wait of a task is $O(\log n)$, and the waiting time of a task that enters the network at an arbitrary time is $O(\log n)$ with high probability.

We contrast these results with the work stealing load balancing protocol, where we show that, in sparse networks, the load in the system and the waiting time can be exponential in the network size.

*Supported in part by NSF grants CCR-0121154, and DMI-0121495.

1. Introduction

Efficient utilization of parallel and distributed systems can often depend on dynamic load balancing of individual tasks between processors. In the dynamic load balancing problem, we consider a system that is designed to run indefinitely. New jobs arrive during the run of the system, and existing jobs are executed by the processors and leave the system. The arrival of new jobs may not be evenly distributed between the processors. The task of the load balancing protocol is to maintain approximately uniform job load between the processors, and in particular to keep all processors working as long as there are jobs in the system waiting for execution.

We assume a simple combinatorial model of load balancing following a number of earlier studies. The computing system is represented by a connected, undirected, n-node graph. Jobs (represented by tokens) have equal execution time. The load of a node is the number of tokens in its queue. A processor can execute (or consume) one token per step, and we assume that it executes the oldest job in its queue. In each step a processor can also move a number of jobs from its queue to a queue of an adjacent node in the network. This abstraction models the case in which the execution time of a job is significantly longer than the time it takes to move a job to an adjacent node. The assumption that all jobs have equal execution time simplifies the analysis while still capturing the combinatorial complexity of the load balancing problem in networks.

Dynamic load balancing algorithms have been studied extensively in experimental settings, demonstrating significant run-time improvements obtained by relatively simple load balancing techniques [17, 18]. Rigorous, theoretical study of load balancing in the past has focused mainly on static analysis [1, 5, 6, 8, 12, 14], where a set of jobs is initially placed in the processors and the algorithm needs to distribute the jobs almost evenly between the processors in a minimum number of parallel rounds. A number of important techniques have been developed in this line of work,

and in particular our work here builds on the static analysis in [6].

Load balancing, however, is best analyzed in a dynamic setting that captures the actual application of such protocols. An important step in that direction was taken in [3], where the work stealing model was shown to be stable on the complete network. The main tool used in that work was the stability conditions for ergodic Markov chains, and consequently their stability result holds only for Markovian adversaries. In addition, a number of other works have studied dynamic load balancing under the assumptions that jobs are generated by a randomized process that is oblivious to the current state of the system [7, 9, 10, 16]. Finally, [2, 11] proved stability results for load balancing on a general network assuming the deterministic adversarial model. The computational model there is different, assuming that only one job can traverse an edge per step.

2. Model and Main Results

In this work we address both the stability and the efficiency (waiting time) of the load balancing task. We present a simple local randomized protocol and analyze its performance on a general n-node network, and under several adversarial models for the arrival of jobs into the system. Since we do not use Markov chain techniques in the analysis, our results are not restricted to Markovian adversaries. That is, the process that injects new jobs into the system can use information about all previous steps. For concreteness, we first present our results here for two simple but powerful adversaries: (1) a deterministic adversary that can add up to λn new jobs to the network at any step; and (2) a randomized adversary that places n generators in the processors in an arbitrary fashion, and each generator adds one new job to the processor in which it is placed independently with probability at most $\lambda < 1$.

We first show that under these adversary input models the system is stable, that is, the expected total load in the system is bounded with respect to time. The following theorem, proven in section 5.3, relates the load in the system to the network topology, and establishes the stability of the system. We assume a network $G = (V, E)$ that has n nodes, maximum degree at most d, and whose Laplacian[1] has smallest nontrivial eigenvalue Λ. For convenience, we define the quantity $\gamma = \Lambda/16d$.

Theorem 2.1. *The system is stable and as time tends to infinity the expected total load in the system is $O(\gamma^{-1} n \ln n)$. If the system starts with no load, the limit is not required.*

[1]Let A denote the adjacency matrix of a graph G, and let $D = (d_{ij})$, where d_{ij} is the degree of node i if $i = j$, and is 0 otherwise. The Laplacian of G is the matrix $L = D - A$. The eigenvalues of L are $0 = \Lambda_1 \leq \Lambda_2 \leq \cdots \leq \Lambda_n$, and $\Lambda_2 = \Omega(n^{-2})$, if G is connected

Next we address the efficiency of the load balancing protocol, an important performance parameter that was not addressed in most previous work. Again we relate the waiting time of jobs to the topology of the network.

Theorem 2.2. *Let $W(t)$ be the wait of a job that arrived at time t. For any constant $c > 1$ there is a constant $\kappa = \kappa(c)$, such that*

1. $\lim_{t\to\infty} \mathbf{Pr}(W(t) \leq \kappa\gamma^{-1} \ln n) \geq 1 - n^{-c}$.
2. $\lim_{t\to\infty} \mathbf{E}[W(t)] = O(\gamma^{-1} \ln n)$.

The above bounds hold without the limits if the system starts with no load.

In particular, for bounded degree regular expanders $\gamma = \Theta(1)$, and the diameter of the graph is $O(\log n)$, so for these graphs the above result is optimal.

The analysis technique developed here is fairly general and can be applied to a variety of other adversarial input models. In Section 6 we discuss its application to the deterministic and stochastic adversaries studied in [4].

We contrast these results with the work stealing load balancing protocol where we show that in sparse networks the load in the system and the waiting time can be exponential in the network size.

3. Protocol

If we were just interested in a stability result we could use the protocol studied in [6]: in each step nodes are matched randomly with direct neighbors in the network, and if node v is matched with node u, they equalize their load subject to integer rounding. The details of the algorithm are given below.

1. **Matching Phase:**
   - **For** each node i
     - Node i inserts each incident edge (i, j) into a set S with probability $\frac{1}{8\max(d_i, d_j)}$, where d_i is the degree of node i.
     - Node i removes edge (i, j) from S if some edge (i, k) or (j, k) is in S, with $k \neq i, j$.
   - Let the matching M consist of the remaining edges in S.
2. **Transfer Phase:**
   - **If** $(i, j) \in M$
     - i and j equalize their loads so that, say, i gets load $\lceil \ell_t(i) + \ell_t(j)/2 \rceil$ and j gets load $\lfloor \ell_t(i) + \ell_t(j)/2 \rfloor$, where $\ell_t(i)$ is the load of processor i in the beginning of the step.

To bound the waiting time of jobs in the system we need to augment the above protocol with a distributed load version of the *first-in-first-out* queueing discipline. It is not

enough to require that a node consumes the oldest job in its queue; we also need to consider the jobs' ages in the load balancing procedure. In particular, when u and v are matched they should not only equalize their total load, but also equalize (up to rounding) the load that they have above any given age. A simple method to maintain this property follows.

1. **Matching Phase:**
   - **For** each node i
     - Node i inserts each incident edge (i,j) into a set S with probability $\frac{1}{8\max(d_i,d_j)}$, where d_i is the degree of node i.
     - Node i removes edge (i,j) from S if some edge (i,k) or (j,k) is in S, with $k \neq i,j$.
   - Let the matching M consist of the remaining edges in S.
2. **Transfer Phase:**
   - **If** $(i,j) \in M$
     - Let $J_1^i, J_2^i, \ldots, J_{\ell_t(i)}^i$ (where $\ell_t(i)$ is the load of processor i in the beginning of the step), and $J_1^j, J_2^j, \ldots, J_{\ell_t(j)}^j$ be the jobs in the queues of nodes i and j respectively, where J_1^* is the oldest job in node $*$ and $J_{\ell_t(*)}^*$ is the newest.
     - Node i sends jobs $J_2^i, J_4^i, \ldots, J_{2\lfloor \ell_t(i)/2 \rfloor}^i$ to node j. Similarly, node j sends jobs $J_2^j, J_4^j, \ldots, J_{2\lfloor \ell_t(i)/2 \rfloor}^j$ to node i.
     - Node i merges jobs $J_1^i, J_3^i, J_5^i, \ldots$ with $J_2^j, J_4^j, J_6^j, \ldots$ in its queue, so that finally, if a job J is older than a job J', then job J is in the queue before job J'.
     - Similarly, node j merges jobs $J_1^j, J_3^j, J_5^j, \ldots$ with $J_2^i, J_4^i, J_6^i, \ldots$ in its queue, so that finally, if a job J is older than a job J', then job J is in the queue before job J'.

4. Analysis of the Static Case

We first analyze our load balancing protocol in a static setting in which initial load is placed on the n processors, and load is moved between processors until the loads in all the processors are approximately equal. No new load is added to or removed from the system through the execution of the protocol.

Our analysis of the static case is based on coupling the execution of our protocol with the nonintegral protocol studied in [6]. The only difference between the two protocols is that in the nonintegral protocol the load is equally distributed between the two matched processors with no rounding. We consider two copies of the network starting with the same initial distribution, one using our protocol and the other using the nonintegral protocol. The two processes are coupled so that they use the same matching at every time step.

Fix any initial distribution of K tokens to the nodes in V, and let $\bar{\ell} = K/n$. For each time step $t \geq 0$ and $u \in V$, let $\ell_t(v)$ be the number of tokens at v at the end of step t of the original, integral protocol, and let $\ell'_t(u)$ be the number of tokens at u at the end of step t in the nonintegral copy of the protocol. Also, for each time step $t \geq 0$, let $\Phi'_t = \sum_{v \in V} (\ell'_t(v) - \bar{\ell})^2$. Note that if the total load in the system is K, then for any $t \geq 0$, $\Phi'_t \leq K^2$. It is also easy to see that Φ'_t is nonincreasing with time.

Recall that $\gamma = \Lambda/16d$. The performance of the nonintegral protocol was analyzed in terms of γ in [6]. The relevant result is the following lemma:

Lemma 4.1. *For any $t \geq 0$,*

$$\mathbf{E}\left[\frac{\Phi'_t - \Phi'_{t+1}}{\Phi'_t}\right] \geq \gamma \quad \textit{or} \quad \mathbf{E}\left[\frac{\Phi'_{t+1}}{\Phi'_t}\right] \leq 1 - \gamma,$$

where the probability space considered is defined by the matchings chosen in each step.

Adapting the technique in [6] we can prove the following static load balancing result for the nonintegral copy:

Lemma 4.2. *If $t \geq \gamma^{-1}(2\ln K + c\ln n)$, then the probability that there is some $v \in V$ with $|\ell'_t(v) - \bar{\ell}| > 1$ is at most $1/n^c$.*

Proof.

$$\Phi'_t = \frac{\Phi'_t}{\Phi'_{t-1}} \frac{\Phi'_{t-1}}{\Phi'_{t-2}} \cdots \frac{\Phi'_1}{\Phi'_0} \Phi'_0.$$

And

$$\mathbf{E}[\Phi'_t] = \mathbf{E}\left[\frac{\Phi'_t}{\Phi'_{t-1}} \frac{\Phi'_{t-1}}{\Phi'_{t-2}} \cdots \frac{\Phi'_1}{\Phi'_0}\right] \Phi'_0.$$

But the random variable $\frac{\Phi'_j}{\Phi'_{j-1}}$ depends only on the random matching in the j-th iteration, and therefore is independent of any other $\frac{\Phi'_i}{\Phi'_{i-1}}$ for $i \neq j$. Applying Lemma 4.1 we have,

$$\begin{aligned}
\mathbf{E}[\Phi'_t] &= \mathbf{E}\left[\frac{\Phi'_t}{\Phi'_{t-1}}\right] \mathbf{E}\left[\frac{\Phi'_{t-1}}{\Phi'_{t-2}}\right] \cdots \mathbf{E}\left[\frac{\Phi'_1}{\Phi'_0}\right] \Phi'_0 \\
&\leq (1-\gamma)^t \Phi'_0 \\
&\leq \Phi'_0 e^{-\gamma t} \\
&\leq e^{-c\ln n},
\end{aligned}$$

where in the last inequality we used the facts that $\Phi'_0 \leq K^2$ and $t \geq \gamma^{-1}(2\ln K + c\ln n)$. Applying Markov's inequality yields $\mathbf{Pr}(\Phi'_t > 1) < n^{-c}$. □

We will now tie the performance of our protocol to the performance of the nonintegral copy.

Lemma 4.3. *For any $t \geq 0$ and any $v \in V$, $|\ell_t(v) - \ell'_t(v)| \leq t/2$, regardless of the chosen matchings and the rounding decisions in the original protocol.*

Proof. By induction on $t \geq 0$. If $t = 0$, then $\ell_t(v) = \ell'_t(v)$ for every $v \in V$, because no randomness has been introduced into the process yet. For the induction step, suppose that the lemma holds for time t. Choose any $v \in V$. If v is not matched at time $t+1$, then $\ell_{t+1}(v) = \ell_t(v)$ and $\ell'_{t+1}(v) = \ell'_t(v)$, so the lemma holds by the induction hypothesis. Otherwise, v is matched to some vertex u at time $t+1$. In this case, for any rounding choice, $(\ell_t(u) + \ell_t(v) - 1)/2 \leq \ell_{t+1}(v) \leq (\ell_t(u) + \ell_t(v) + 1)/2$. Also, $\ell'_{t+1}(v) = (\ell'_t(u) + \ell'_t(v))/2$. These observations give

$$\begin{aligned}
|\ell_{t+1}(v) - \ell'_{t+1}(v)| &\leq \frac{1}{2} + \left| \frac{\ell_t(u) + \ell_t(v)}{2} - \frac{\ell'_t(u) + \ell'_t(v)}{2} \right| \\
&\leq \frac{1}{2} + \frac{1}{2}|\ell_t(u) - \ell'_t(u)]| \\
&\qquad + \frac{1}{2}|\ell_t(v) - \ell'_t(v)| \\
&\leq \frac{1}{2} \cdot \frac{t}{2} + \frac{1}{2} \cdot \frac{t}{2} + \frac{1}{2} \\
&= \frac{t+1}{2},
\end{aligned}$$

where the first inequality follows from previous observations, the second inequality follows from the triangle inequality, and the third inequality follows from the induction hypothesis. □

Putting Lemmata 4.2 and 4.3 together yields the following theorem.

Theorem 4.1. *Let $\hat{t} = \gamma^{-1}(2 \ln K + c \ln n)$. Then, for any $t \geq \hat{t}$, the probability that there is some $v \in V$ with $|\ell_t(v) - \bar{\ell}| > 1 + \frac{\hat{t}}{2}$ is at most $1/n^c$.*

Proof. We first prove the theorem for the case $t = \hat{t}$. Regardless of the matchings generated in the first $\hat{t}$ steps, $|\ell_{\hat{t}}(v) - \ell'_{\hat{t}}(v)| \leq \hat{t}/2$ for every $v \in V$, by Lemma 4.3. With probability at least $1 - 1/n^c$, $|\ell'_{\hat{t}}(v) - \bar{\ell}| \leq 1$ for all $v \in V$, by Lemma 4.2. Adding these inequalities proves the theorem for the case $t = \hat{t}$. To extend the result for the case $t > \hat{t}$ notice that the sequence $\{\max_{v \in V} |\ell_t(v) - \bar{\ell}|\}_{t \geq \hat{t}}$ is nonincreasing. □

5. Analysis of the Dynamic Case

We first prove that for any constant $\lambda < 1$ and any connected network topology, the load distribution protocol is stable. We then use the structure developed here to analyze the waiting time of individual jobs.

We first give the values of some constants that we use throughout the proof, for easy reference. $\epsilon = 1 - \lambda$, $c > 3$, $c_3 = c + 8$, $c_1 = \frac{2\lambda(2c_3+1)}{(1+\epsilon)^{-1}-\lambda}$, $c_4 = c_1 + c_3 + 1$, $c_2 = c_3 + c_4$, $c_6 = (1+\epsilon)\lambda c_2$, $c_5 = c_1 - c_6$, $\theta = c_5/(c_2 + c_5)$.

To simplify the analysis we partition time into *epochs*, where each epoch has $c_2\gamma^{-1} \ln n$ steps. We say that the system is in a *good* state at the end of an epoch if the total load in the system is less than $c_1\gamma^{-1}n \ln n$, otherwise the system is in a *bad* state.

We will define a two state renewal process that alternates between G and B states. The crux of the proof is to show that the distribution of the length of a B segment in the renewal process stochastically dominates the distribution of the number of successive epochs in which the original system is in a bad state, conditioned on its past. On the other hand, the distribution of the number of successive epochs in which the original system is in a good state, conditioned on its past, stochastically dominates the length of time that the renewal process is in the G state. Once this relation is established, the analysis of the renewal process implies the stability and waiting-time results for the load balancing protocol.

5.1. Analysis of One Epoch

We say that an epoch is *successful* if one of the following conditions hold:

- The total system load is less than $c_1\gamma^{-1}n \ln n$ immediately after the epoch finishes.
- The total load of the system decreases by at least $c_5\gamma^{-1}n \ln n$ during the epoch.

Lemma 5.1. *The probability that an epoch is successful, conditioned on all events in the past, is at least $1 - 2n^{-c}$.*

Proof. Assume that the epoch starts at time τ. Let $L(t)$ be the total load in the system at time t, so $L(\tau)$ is the load at the start of the epoch. For the purpose of the analysis we split the length of the epoch into two parts. In the first $c_3\gamma^{-1} \ln n$ steps we focus on the distribution of load between the processors, and in the remaining $c_4\gamma^{-1} \ln n$ steps we focus on the consumption of load by the processors (although load is consumed throughout the execution by processors that have load).

To analyze the distribution of load between the processors, we couple the actual execution of the protocol in the first $c_3\gamma^{-1} \ln n$ steps with an execution of the protocol in a static setting that starts with a total load of exactly $c_1\gamma^{-1}n \ln n$. We refer to the actual execution of the protocol as the *dynamic copy* and the static execution as the *static copy*.

To formulate the coupling we first color all the tokens in the dynamic copy at time τ by red and blue. A subset of the $M = \min\{L(\tau), c_1\gamma^{-1}n \ln n\}$ oldest (ties broken arbitrarily) tokens in the system at time τ is colored red, and the rest are colored blue.

We now place $c_1\gamma^{-1}n\ln n$ tokens in the static copy so that each node in the static copy starts the process with at least as many tokens as the number of red tokens in the corresponding node of the dynamic copy.

New tokens that arrive through the execution of the dynamic copy are colored blue. Red tokens have higher priority to be consumed by the node, since they are at the beginning of the queues. The executions of the two copies are coupled so that they use the same matching in each step. Note that the load balancing protocol guarantees that if u is matched with v then, while executing the load balancing in the dynamic copy, u and v equalize (up to 1) their number of red tokens. Since the red tokens are at least as old as the blue ones, we can assume that they are at the heads of the queues (so in particular they will be consumed before the blue ones). Finally, we also assume that the rounding decisions in both executions are the same, that is, if one node receives one more red token and there is a rounding decision in the static copy, the same node receives the extra token; we are allowed to do this because the results of Section 4 hold for an arbitrary rounding procedure.

Lemma 5.2. *At any time $\tau \le t \le \tau + c_3\gamma^{-1}\ln n$ the number of red tokens in each node of the dynamic copy is bounded by the number of tokens in the corresponding node in the static copy.*

Proof. We prove the claim by induction on t. The claim is true for $t = \tau$ by the construction of the static copy. Assume that the claim holds after the execution of step $t-1$, and consider the load of node v after the execution of step t. If v was not part of a matching in step t then the load of the static copy did not change. The number of red tokens of the dynamic copy either did not change, or was reduced by 1 if a red token was consumed. In both cases, using the induction hypothesis, the number of red tokens in the dynamic copy after the execution of step t is bounded by the number of tokens in the static copy of v.

Assume now that node v was matched with node u during step t. Let $r_t(u)$ and $r_t(v)$ be the number of red tokens in the dynamic copies of u and v after step t, and let $\ell_t(u)$ and $\ell_t(v)$ be the number of tokens in the static copies of u and v. After the execution of step t, using the induction hypothesis (and the fact that red tokens might be consumed but no tokens are consumed in the static execution), as well as the fact that we perform the same rounding decisions for both copies, we show

$$\begin{aligned} r_t(u) &\le \left\lceil \frac{1}{2}(r_{t-1}(u) + r_{t-1}(v)) \right\rceil \\ &\le \left\lceil \frac{1}{2}(\ell_{t-1}(u) + \ell_{t-1}(v)) \right\rceil = \ell_t(u), \end{aligned}$$

and

$$\begin{aligned} r_t(v) &\le \left\lfloor \frac{1}{2}(r_{t-1}(u) + r_{t-1}(v)) \right\rfloor \\ &\le \left\lfloor \frac{1}{2}(\ell_{t-1}(u) + \ell_{t-1}(v)) \right\rfloor = \ell_t(v). \end{aligned}$$

□

Applying Theorem 4.1 to the execution of the static copy, we see that at time $\tau + c_3\gamma^{-1}\ln n$ (the reader can verify that the condition of Theorem 4.1 holds for $t = c_3\gamma^{-1}\ln n$ and $K = c_1\gamma^{-1}n\ln n$, for sufficiently large n), with probability at least $1 - n^{-c}$ no node of the static copy has more than $c_3\gamma^{-1}\ln n + c_1\gamma^{-1}\ln n + 1 \le c_4\gamma^{-1}\ln n$ tokens. Thus, with the same probability, no node has more than $c_4\gamma^{-1}\ln n$ red tokens in the dynamic copy. If the red tokens do not move in the next $c_4\gamma^{-1}\ln n$ steps then they are clearly all consumed during that time interval, since red tokens are consumed first. If at some step during this interval a red token moves, then, after that step, the node with the maximum number of red tokens will either have the same number of red tokens, or even fewer. So if a red token would have been consumed if it hadn't moved, it will be consumed now, as well. Thus, we have shown that with probability $1 - n^{-c}$ at least $M = \min\{L(\tau), c_1\gamma^{-1}n\ln n\}$ tokens are consumed in this epoch.

The expected number of new tokens arriving during the epoch is at most $\lambda c_2\gamma^{-1}n\ln n$, and by applying a Chernoff bound we deduce that with probability at least $1 - n^{-c}$ this number is bounded by $c_6\gamma^{-1}n\ln n = (1+\epsilon)\lambda c_2\gamma^{-1}n\ln n$.

We now distinguish between two cases.

1. If $L(\tau) \le c_1\gamma^{-1}n\ln n$, then with probability at least $1 - n^{-c}$, the initial $L(\tau)$ tokens are consumed during this epoch, and with probability at least $1 - n^{-c}$ no more than $c_6\gamma^{-1}n\ln n < c_1\gamma^{-1}n\ln n$ tokens join the system during this epoch. Thus, with probability at least $1 - 2n^{-c}$ the epoch is successful since it ends with less than $c_1\gamma^{-1}n\ln n$ total load.

2. If $L(\tau) > c_1\gamma^{-1}n\ln n$, then with probability at least $1 - 2n^{-c}$ at least $c_1\gamma^{-1}n\ln n$ tokens are consumed during this epoch and no more than $c_6\gamma^{-1}n\ln n$ new token arrive during this epoch. In this case, the epoch is successful, since the total load decreases by at least $c_1\gamma^{-1}n\ln n - c_6\gamma^{-1}n\ln n = c_5\gamma^{-1}n\ln n$.

Note that the probability space considered here is defined by the distribution of the matchings performed during this epoch, and by the arrival of new load (and not by the adversary's placement of the generators). So, in this probability space, the event that the epoch is successful is independent of any event in the past. □

5.2. Basic Renewal Theory

In this section we provide some basic background for renewal processes, that is needed for the analysis of the process. For more information refer for example to [15, Chapter 3].

Let $\{X_n, n = 1, 2, \dots\}$ be a sequence of nonnegative independent random variables with a common distribution function F. Let $S_0 = 0$, and for $n \geq 1$, $S_n = \sum_{i=1}^{n} X_i$; finally define $N(t) = \sup\{n : S_n \leq t\}$. The process $\{N(t), t \geq 0\}$ is a *Renewal Process*. We say that a renewal occurs at time t, if $S_n = t$ for some n. Since the interarrival times X_i are independent and identically distributed, it follows that after each renewal the process starts over again.

We say that a nonnegative random variable X (or the corresponding distribution function) is *lattice* (or periodic) if there exists a $d \geq 0$ such that $\sum_{n=0}^{\infty} \mathbf{Pr}(X = nd) = 1$. We are interested in renewal processes that are nonlattice.

Let $Z(t) = t - S_{N(t)}$ be the *age* at time t, that is, the time that has elapsed from the last renewal before t until t. We can compute the distribution of $Z(t)$ as $t \to \infty$ by Lemma 5.3 (see [15, Corollary 3.13]).

Lemma 5.3. *If F is not lattice, then*

$$\lim_{t\to\infty} \mathbf{Pr}(Z(t) \leq z) = \frac{1}{\mathbf{E}[X_1]} \int_0^x [1 - F(y)]\mathrm{d}y.$$

An *Alternating Renewal Process* can be in one of two states, *on* or *off*. Initially, it is on and it remains on for a time X_1; it then goes off and remains off for a time Y_1; it then goes on for a time X_2, then off for a time Y_2, and so on. We assume, moreover, that the X_i's have the same distribution, that the Y_i's have the same distribution, and that all the X_i's and the Y_i's are independent of each other. Let $P(t)$ be the probability that the system is on at time t. Then, Lemma 5.4 (repeating [15, Proposition 3.11]) gives the probability that at some time, in the limit, the system is found in state on.

Lemma 5.4. *If $\mathbf{E}[X_1 + Y_1] < \infty$ and if $X_1 + Y_1$ is nonlattice, then*

$$\lim_{t\to\infty} P(t) = \frac{\mathbf{E}[X_1]}{\mathbf{E}[X_1] + \mathbf{E}[Y_1]}.$$

5.3. Stability

We now define the two state renewal process that dominates the execution of our protocol. The process alternates between states G and B. Let $T(G)$ be the length of a G segment and $T(B)$ be the length of a B segment. To apply Lemmata 5.3 and 5.4 we define these distributions on continuous time. For any $x \geq 1$, let

- $\mathbf{Pr}(T(G) \geq x) = (1 - 2n^{-c})^{(x-1)}$ and
- $\mathbf{Pr}(T(B) \geq x) = n^{-\theta(c-1)(x-1)}$.

Note that these functions are monotonically decreasing in x,

$$\begin{aligned}\mathbf{E}[T(G)] &= 1 + \int_1^{\infty} (1 - 2n^{-c})^{(x-1)}\mathrm{d}x \\ &\geq \sum_{i=1}^{\infty} (1 - 2n^{-c})^{(i-1)} = \frac{1}{2}n^c,\end{aligned}$$

and

$$\begin{aligned}\mathbf{E}[T(B)] &= 1 + \int_1^{\infty} n^{-\theta(c-1)(x-1)}\mathrm{d}x \\ &\leq 1 + \sum_{i=1}^{\infty} n^{-\theta(c-1)(i-1)} = 1 + \frac{1}{1 - n^{-\theta(c-1)}}.\end{aligned}$$

Now consider the sequence of epochs in the execution of our protocol. Recall that at the end of an epoch the system is in a *good* state if the total load in the system is bounded by $c_1\gamma^{-1}n \ln n$, otherwise the system is in a *bad* state.

Lemma 5.5. *Let $T(good)$ be the distribution of the number of epochs in a given segment in which the system is in a good state, and $T(bad)$ be the number of epochs in a given segment in which the system is in a bad state. Conditioned on all events before the start of the segment, for any $i \geq 1$,*

1. $\mathbf{Pr}(T(good) \geq i) \geq \mathbf{Pr}(T(G) \geq i)$,
2. $\mathbf{Pr}(T(bad) \geq i) \leq \mathbf{Pr}(T(B) \geq i)$.

Proof. If the total load in the system at the start of an epoch is less than $c_1\gamma^{-1}n \ln n$, and the epoch was successful, then the total load in the system at the end of that epoch is also less than $c_1\gamma^{-1}n \ln n$. Therefore, once the system is in a good state it stays in a good state until at least the first unsuccessful epoch. The probability that an epoch is successful, conditioned on the past, is at least $1 - 2n^{-c}$, by Lemma 5.1. Thus,

$$\mathbf{Pr}(T(good) \geq i) \geq (1 - 2n^{-c})^{(i-1)} = \mathbf{Pr}(T(G) \geq i).$$

To bound the distribution of $T(bad)$ we observe that if a segment of i epochs begins when the system is in a good state, and each of those i epochs finishes when the system is in a bad state, then at least θi of those i epochs must have been unsuccessful. Otherwise, the total change in the system load over the course of those i epochs is at most $\theta i c_2 n \ln n - i(1 - \theta)c_5 n \ln n \leq 0$, which implies that the last epoch finishes with the system being in a good state, yielding a contradiction.

The probability of at least θi unsuccessful epochs in a sequence of i epochs, for n large enough, is bounded by

$$\begin{aligned}\binom{i}{\theta(i-1)}\left(\frac{2}{n^c}\right)^{\theta(i-1)} &\le \left(\frac{ei}{\theta(i-1)}\right)^{\theta(i-1)}\left(\frac{2}{n^c}\right)^{\theta(i-1)}\\ &= \left(n^{-c}\cdot\frac{2e}{\theta}\cdot\frac{i}{i-1}\right)^{\theta(i-1)}\\ &\le n^{-\theta(i-1)(c-1)}.\end{aligned}$$

Thus, for sufficiently large n,

$$\mathbf{Pr}(T(bad)\ge i)\le n^{-\theta(c-1)(i-1)} = \mathbf{Pr}(T(B)\ge i).$$

□

We now use the stochastic dominance proven in Lemma 5.5 to analyze the performance of our protocol using renewal theory. Let $\mathcal{E}(t)$ be the event "the system is in a bad state at time t." Then Lemmata 5.4 and 5.5 along with the fact that $T(B)$ and $T(G)$ have continuous distribution functions and therefore are not lattice, yield

Lemma 5.6.

$$\begin{aligned}\lim_{t\to\infty}\mathbf{Pr}(\mathcal{E}(t)) &\le \frac{\mathbf{E}[T(B)]}{\mathbf{E}[T(B)]+\mathbf{E}[T(G)]} \le \frac{\mathbf{E}[T(B)]}{\mathbf{E}[T(G)]}\\ &\le 3n^{-c},\end{aligned}$$

for large enough n.

We now turn to bounding the expected load in the system at time t, thus establishing the fact that the system is stable.

Let $L(t)$ denote the total load in the system at time t.

$$\begin{aligned}\mathbf{E}[L(t)] &= \mathbf{E}[L(t)|\overline{\mathcal{E}(t)}]\mathbf{Pr}(\overline{\mathcal{E}(t)}) + \mathbf{E}[L(t)|\mathcal{E}(t)]\mathbf{Pr}(\mathcal{E}(t))\\ &\le c_1\gamma^{-1}n\ln n + \mathbf{E}[L(t)|\mathcal{E}(t)]\mathbf{Pr}(\mathcal{E}(t)),\end{aligned} \tag{1}$$

where in the inequality we use the fact that in a good state the load of the system is bounded by $c_1\gamma^{-1}n\ln n$.

To compute

$$\mathbf{E}[L(t)|\mathcal{E}(t)] = \sum_{x=1}^{\infty}\mathbf{Pr}(L(t)\ge x\mid\mathcal{E}(t))$$

we need a bound on the probability $\mathbf{Pr}(L(t)\ge x\mid\mathcal{E}(t))$. Observe that if $L(t)\ge ic_2\gamma^{-1}n\ln n$ then the system must have been in a bad state for at least the last $i-1$ epochs. Otherwise, by the beginning of the current epoch, the load would have been at most $(i-2)c_2\gamma^{-1}n\ln n$, so at time t the load would be at most $(i-1)c_2\gamma^{-1}n\ln n$. Thus, we need a bound on $Z(t)$, the time from the start of the bad segment until t. Let $Z'(t)$ be the corresponding random variable in the renewal process, that is, the time from the start of a B segment until t, conditioned on t being in a B segment. With the same argument as in Lemma 5.5, we have

$$\mathbf{Pr}(Z(t)\ge i-1\mid\mathcal{E}(t))\le\mathbf{Pr}(Z'(t)\ge i-1\mid\mathcal{E}(t)).$$

Using that fact, and applying Lemma 5.3, and using again the fact that $T(B)$ and $T(G)$ have continuous distribution functions and therefore are nonlattice, we compute

$$\begin{aligned}&\lim_{t\to\infty}\mathbf{Pr}(L(t)\ge ic_2\gamma^{-1}n\ln n\mid\mathcal{E}(t))\\ &\qquad\le\lim_{t\to\infty}\mathbf{Pr}(Z(t)\ge i-1\mid\mathcal{E}(t))\\ &\qquad\le\lim_{t\to\infty}\mathbf{Pr}(Z'(t)\ge i-1\mid\mathcal{E}(t))\\ &\qquad=\frac{1}{\mathbf{E}[T(B)]}\int_{i-1}^{\infty}\mathbf{Pr}(T(B)\ge x)\mathrm{d}x\\ &\qquad\le n^{-\theta(c-1)i}\end{aligned}$$

where in the last inequality we used the fact that (trivially) $\mathbf{E}[T(B)]\ge 1$. Therefore, Relation (1) and Lemma 5.6 give, for sufficiently large n,

$$\begin{aligned}\lim_{t\to\infty}\mathbf{E}[L(t)] &\le c_1\gamma^{-1}n\ln n\\ &\quad+3n^{-c}c_2\gamma^{-1}n\ln n\sum_{i=1}^{\infty}n^{-\theta i(c-1)}\\ &\le c_1\gamma^{-1}n\ln n+3n^{-c}c_2\gamma^{-1}n\ln n\\ &\le (c_1+1)\gamma^{-1}n\ln n.\end{aligned}$$

Thus, we prove:

Theorem 5.1. *The system is stable and as time tends to infinity the expected total load in the system is $O(\gamma^{-1}n\ln n)$.*

Note that, given that the system started with no load, or even just in a good state, the above bound holds for any t, not only in the limit. To see this, assume that at some point before time t the system had no more than $c_1\gamma^{-1}n\ln n$ load. Then the system can have load $c_1\gamma^{-1}n\ln n+ic_2\gamma^{-1}n\ln n$ at time t only if the system was in a bad state in the last i epochs up to time t. The probability of that event is bounded by $n^{-\theta i(c-1)}$. Thus,

$$\begin{aligned}\mathbf{E}[L(t)] &\le c_1\gamma^{-1}n\ln n+c_2\gamma^{-1}n\ln n\sum_{i\ge1}n^{-\theta i(c-1)}\\ &= O(\gamma^{-1}n\ln n).\end{aligned}$$

5.4. Waiting Time

Having established that the system is stable, the next important performance parameter is the waiting time of a job from when it enters the system until it is executed. For a given task that entered the system at time t, let $W(t)$ be the number of steps until the task is executed.

The proof of Lemma 5.1 shows that if an epoch starts when the system is in a good state, (i.e., with less than $c_1\gamma^{-1}n\ln n$ load), and the execution of that epoch is successful, then all the load that was in the system before the epoch started is consumed during the epoch. Thus, if the system is in a good state at time t, and both the current and the next epoch are successful, then $W(t) \leq 2c_2\gamma^{-1}\ln n$. By summing the failure probabilities we prove that

$$\lim_{t\to\infty} \mathbf{Pr}(W(t) \leq 2c_2\gamma^{-1}\ln n) \geq 1 - 7n^{-c}, \qquad (2)$$

for large enough n, and the limit can be removed if the system starts in a good state.

Next we turn to computing $\mathbf{E}[W(t)]$. The first problem we have to address is that there may be unsuccessful epochs during a good segment. The probability that t is in a good segment but the next epoch is unsuccessful is bounded by $2n^{-c}$. To simplify the computation, we assume that if an unsuccessful epoch occurs during a good segment, then the system switches to a bad state, and add $2n^{-c}$ to the probability that the system is in a bad state at time t. To bound the waiting time when a job arrives during a bad state, we note that the waiting time of a job is always bounded by the total load of the system at the time it arrives. If the system is in a bad state at time t, and the system switched to a bad state i epochs back, then the load of the system at time t is bounded by $c_1\gamma^{-1}n\ln n + ic_2\gamma^{-1}n\ln n$.

If we let $\mathcal{F}(t)$ denote the event "the system is in a good state at time t, and both the current and the next epochs are successful," then by Lemma 5.5 and Relation (2), we have that

$$\begin{aligned}
\lim_{t\to\infty}\mathbf{E}[W(t)] &\leq \lim_{t\to\infty}\Big(2c_2\gamma^{-1}\ln n\cdot\mathbf{Pr}(\mathcal{F}(t)) \\
&\qquad + \mathbf{E}[L(t)\mid\overline{\mathcal{F}(t)}]\cdot\mathbf{Pr}(\overline{\mathcal{F}(t)})\Big) \\
&\leq 2c_2\gamma^{-1}\ln n + \Bigg(c_1\gamma^{-1}n\ln n \\
&\qquad + c_2\gamma^{-1}n\ln n\sum_{i=1}^{\infty} n^{-\theta i(c-1)}\Bigg)\cdot 7n^{-c} \\
&= O(\gamma^{-1}\ln n).
\end{aligned}$$

Where again the limit can be removed if the system starts in a good state. We have proven the following theorem:

Theorem 5.2. *Let $W(t)$ be the wait of a job that arrived at time t.*

1. $\lim_{t\to\infty}\mathbf{Pr}(W(t) \leq 2c_2\gamma^{-1}\ln n) \geq 1 - 7n^{-c}$.
2. $\lim_{t\to\infty}\mathbf{E}[W(t)] = O(\gamma^{-1}\ln n)$.

The above bounds hold without the limits if the system starts in a good state.

6. Other Adversarial Models

We now discuss application of our proof technique to more general input adversarial models. In particular, we consider the type of adversaries proposed by Borodin et al. in [4]. Following that work we define a $(w,\lambda n)$ input adversary as a process that inserts jobs in the system subject to the condition that for every sequence of w consecutive time steps, the total inserted load is at most λnw. This allows the adversary to insert more jobs at some time steps, as long as the total load in windows of size w is bounded. An extension is a $(w,\lambda n)$ stochastic adversary, whose input load is a random variable, with the property that the expected injected load during any sequence of w consecutive time steps is bounded by λnw, and additionally, for some $p > 2$ the p-th moment of the new load is bounded (see [4] for detailed discussion).

We have shown that our system is stable for a window of size $w = 1$. In the next theorem we generalize it to any window size.

Theorem 6.1. *Consider any window size w and some $(w,\lambda n)$ deterministic or stochastic adversary. Then our process is stable for that adversary, that is, there exists some finite Ξ such that for any time t the expected load is bounded: $\mathbf{E}[L(Y_t)] < \Xi$.*

Proof. We will prove the stability by showing that the expected load of the system decreases if it is above some threshold depending on w. We make use of Theorem 6.2 which appeared in [4, Lemma 2], and follows from [13, Theorem 1].

Theorem 6.2. *Let $X_1, X_2, \ldots$ be a sequence of nonnegative random variables satisfying the following properties:*

1. *There exist positive constants α and β such that for all $x_1,\ldots,x_i$ with $x_i > \beta$,*
$$\mathbf{E}[X_{i+1} - X_i | X_1 = x_1,\ldots,X_i = x_i] \leq -\alpha$$
2. *There exists a positive constant ξ and a $p > 2$ such that for all $x_1,\ldots,x_i$*
$$\mathbf{E}[|X_{i+1} - X_i|^p | X_1 = x_1,\ldots,X_i = x_i] \leq \xi.$$

Then there exists $\Xi = \Xi(X_0,\alpha,\beta,\xi)$ and t_0 such that for all $t \geq t_0$, $\mathbf{E}[X_t] \leq \Xi$.

Note that all the constants in the above theorem can be functions of the system parameters n and γ.

We now prove the result for a window of size $w = c_2\gamma^{-1}\ln n$; for longer w we consider longer epochs—the details will appear in the full paper. So we consider a window of size $w = c_2\gamma^{-1}\ln n$ and we verify the conditions

of Theorem 6.2. The second condition is satisfied since $|X_{i+1} - X_i|$ is bounded by the new load during the epoch, plus at most $n \cdot w$(the maximum consumed load during the epoch), and the model assumes that for some $p > 2$ the new load has a bounded p-th moment. For the first condition, we will argue similarly to the proof of Lemma 5.1. There, by using the static-case result, we showed that if the initial load is greater than $c_1\gamma^{-1}n\ln n$, then with probability at least $1-n^{-c}$ the number of consumed jobs during the epoch is at least $c_1\gamma^{-1}n\ln n$, *independently of the newly-arrived jobs*. On the other hand, for any stochastic adversary, the expected number of new jobs is $\lambda c_2\gamma^{-1}n\ln n$.

Hence, if $L_t = \ell_t > c_1\gamma^{-1}n\ln n$, we let $\beta = c_1\gamma^{-1}n\ln n$, and $\mathcal{S}$ denote the event "the number of consumed jobs during the epoch from t until $t + c_2\gamma^{-1}\ln n$ is at least $c_1\gamma^{-1}n\ln n$," and we compute

$$\begin{aligned}
&\mathbf{E}[L_{t+c_2\gamma^{-1}\ln n} - L_t | L_1 = \ell_1, \ldots, L_t = \ell_t] \\
&\quad = \mathbf{E}[L_{t+c_2\gamma^{-1}\ln n} - L_t | L_1 = \ell_1, \ldots, L_t = \ell_t, \mathcal{S}] \cdot \mathbf{Pr}(\mathcal{S}) \\
&\qquad + \mathbf{E}[L_{t+c_2\gamma^{-1}\ln n} - L_t | L_1 = \ell_1, \ldots, L_t = \ell_t, \overline{\mathcal{S}}] \cdot \mathbf{Pr}(\overline{\mathcal{S}}) \\
&\quad \leq -c_1\gamma^{-1}n\ln n + \lambda c_2\gamma^{-1}n\ln n + c_2\gamma^{-1}n\ln n \cdot n^{-c} \\
&\quad < -(c_1 - \lambda c_2)\gamma^{-1}n\ln n + 1
\end{aligned}$$

By letting $X_i = L_{ic_2\gamma^{-1}n\ln n}$, we can conclude that there exists some Ξ large enough so that for any t sufficiently large $\mathbf{E}[L(Y_t)] \leq \Xi$. □

Similarly to Subsection 5.4 we can use this technique to bound the waiting time as well. In the case of a stochastic adversary the time will depend on the moments of the distribution. The details are left for the full paper.

7. Work Stealing on Sparse Networks

In the *work stealing* load balancing protocol, nodes initiate load balancing steps only when their queues are empty. Work stealing was shown in [3] to be stable on the complete network for any constant $\lambda < 1$. For comparison to our result, we show here that when applied to sparse networks, the work stealing protocol is either unstable or stable but with expected load exponential in the network size.

Consider first a deterministic adversary that can add in each step up to λn tokens to the n-node network. Let v be a node in the network with constant degree d. In each step, the adversary puts $\lambda n - d$ tokens in v and one token in each of its neighbors. Since the queue of the d neighbors of v is never empty, v never participates in a load balancing step, and so its load increases (unboundedly) by $\lambda n - d - 1$ in each step.

Consider now the randomized adversarial input process studied in [3]. Here the adversary can place n generators among the nodes of the network in each step, and each generator adds one new token to the queue of its location independently with probability λ, all the generators being independent. Assume again a node v with constant degree d, and assume that in each step the adversary places exactly $m = \lfloor n/(d+1) \rfloor$ generators at v and at least m generators at each of its neighbors. Let $\ell_v(t)$ denote the load at v at time t.

Define $p = (d+1+m/(1-\lambda))e^{-\lambda m}$, and assume that n is large, so that $p < 1$ (we use this fact later). Fix some time step $t \geq 1$. If the load at v does not increase during this time step, then either the generators at some neighbor of v did not produce any load (in which case v gave away half of its load), or the generators at v produced at most one unit of load (so the load of v either stays the same or decreases by one). Thus,

$$\begin{aligned}
\mathbf{Pr}(\ell_v(t) - \ell_v(t-1) < 1) &\leq d(1-\lambda)^m + (1-\lambda)^m \\
&\quad + \lambda m(1-\lambda)^{m-1} \\
&\leq p,
\end{aligned} \tag{3}$$

conditioned on all past events.

We now construct a random sequence $\{\ell'_v(t)\}_{t\geq 0}$ as follows. $\ell'_v(0) = 0$ and for $t > 0$,

$$\ell'_v(t) = \begin{cases} \ell'_v(t-1)/2 & \text{with probability } p, \\ \ell'_v(t-1) + 1 & \text{with probability } 1-p, \end{cases}$$

with all the random choices being independent.

Using the recursion

$$\mathbf{E}[\ell'_v(t)] = \frac{p}{2}\mathbf{E}[\ell'_v(t-1)] + (1-p)(\mathbf{E}[\ell'_v(t-1)] + 1),$$

we compute

$$\mathbf{E}[\ell'_v(t)] = \frac{2(1-p)}{p}\left[1 - \left(1 - \frac{p}{2}\right)^t\right],$$

and

$$\begin{aligned}
\mathbf{E}[L(t)] \geq \mathbf{E}[\ell_v(t)] &\geq \mathbf{E}[\ell'_v(t)] \\
&= \frac{2(1-p)}{p}\left[1 - \left(1 - \frac{p}{2}\right)^t\right],
\end{aligned}$$

To see why the second inequality holds, notice that by Relation 3 the load of node v decreases with probability at most p. Moreover, if the load of node v decreases, the new load will be at least half of the old load, hence each load $\ell_v(t)$ stochastically dominates the corresponding $\ell'_v(t)$. The equality follows from induction on t and properties of conditional expectation.

Using the fact that $p < 1$, we get

$$\begin{aligned}\lim_{t\to\infty} \mathbf{E}[L(t)] &\geq \frac{2(1-p)}{p} \\ &= 2\left(\frac{1}{p} - 1\right) \\ &= \frac{2(1-\lambda)e^{\lambda\left\lfloor \frac{n}{d+1}\right\rfloor}}{(d+1)(1-\lambda) + \lambda\left\lfloor \frac{n}{d+1}\right\rfloor} - 2.\end{aligned}$$

So in the limit the expected total load in the system is at least exponential in the network size, n.

References

[1] W. Aiello, B. Awerbuch, B. M. Maggs, and S. Rao. Approximate load balancing on dynamic and asynchronous networks. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC '93)*, pages 632–641, May 1993.

[2] E. Anshelevich, D. Kempe, and J. Kleinberg. Stability of load balancing in dynamic adversarial systems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 399–406, New York, May 2002.

[3] P. Berenbrink, T. Friedetzky, and L. A. Goldberg. The natural work-stealing algorithm is stable. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pages 178–187, Oct. 2001.

[4] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *Journal of the ACM*, 48(1):13–38, Jan. 2001.

[5] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. *SIAM Journal on Computing*, 29(1):29–64, Feb. 2000.

[6] B. Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *Journal of Computer and System Sciences*, 53(3):357–370, Dec. 1996.

[7] R. Luling and B. Monien. A dynamic distributed load balancing algorithm with provable good performance. In *Proceedings of the 5th ACM Symposium on Parallel Algorithms and Architectures (SPAA '93)*, pages 164–172, July 1993.

[8] F. Meyer auf der Heide, B. Oesterdiekhoff, and R. Wanka. Strongly adaptive token distribution. In *Proceedings of the 20th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 398–409, July 1993.

[9] M. Mitzenmacher. Load balancing and density dependent jump Markov processes. In *Proceedings of 37th IEEE Conference on Foundations of Computer Science (FOCS '96)*, pages 213–222, Oct. 1996.

[10] M. Mitzenmacher and D. S. R. Center. Analyses of load stealing models based on differential equations. In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '98)*, pages 212–221, June 1998.

[11] S. Muthukrishnan and R. Rajaraman. An adversarial model for distributed dynamic load balancing. In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '98)*, pages 47–54, June 1998.

[12] D. Peleg and E. Upfal. The token distribution problem. *SIAM Journal on Computing*, 18(2):229–243, Apr. 1989.

[13] R. Pemantle and J. S. Rosenthal. Moment conditions for a sequence with negative drift to be unifromly bounded in L^r. *Stochastic Processes and their Applications*, 82:143–155, 1999.

[14] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of markov chains and the analysis of iterative load balancing schemes. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science (FOCS '98)*, pages 694–705, Nov. 1998.

[15] S. M. Ross. *Applied Probability Models with Optimization Applications*. Holden-Day, 1970.

[16] L. Rudolph, M. Slivkin-Allalouf, and E. Upfal. A simple load balancing scheme for task allocation in parallel machines. In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '91)*, pages 237–245, July 1991.

[17] C. Xu and F. Lau. Iterative dynamic load balancing in multicomputers. *Journal of the Operational Research Society*, 45(7), July 1994.

[18] W. Zhu, C. Steketee, and B. Muilwijk. Load balancing and workstation autonomy on Amoeba. *Australian Computer Science Communications*, 17(1):588–597, 1995.