An Adversarial Model for Distributed Dynamic Load Balancing

S. Muthukrishnan[*] *Rajmohan Rajaraman*[†]

May 17, 2002

Abstract

We study the problem of balancing the load on processors in an arbitrary network. If jobs arrive or depart during the process of load balancing, we have the *dynamic* load balancing problem; otherwise, we have the *static* load balancing problem. While static load balancing on a number of different networks has been well-studied, analytical work on dynamic load balancing is limited. The difficulty lies in modeling the arrivals and departures of jobs in a clean manner.

In this paper, we introduce a new framework for dynamic load balancing in which the job traffic is modeled by an adversary. Our main result is that a simple local load balancing algorithm maintains the load of the network within a stable level against a powerful adversary. Our result holds for different models of traffic patterns and processor communication.

1 Introduction

An important problem in a distributed system is to balance the total workload among the various processors of the system. Such load balancing problems arise in a number of parallel and distributed applications including job scheduling in operating systems (e.g., see [31]), packet routing (e.g., see [25]), and parallel finite element methods (e.g., see [11]); other applications can be found in [29]. Load balancing problems can be classified into two categories: *static* and *dynamic*.

In static load balancing, the total workload is available at the start of the computation, and no new load is added to the system. The main objective is to distribute the total load amongst the processors before initiating the computation such that the assignment of tasks to processors is balanced. Parallel computations such as large-scale partial diferential equations and finite element methods rely on static load balancing. In these applications, the given computation can be divided into a large number of small computational tasks that are distributed among the processors (for example, see [11, 31]). Another important application of static load balancing arises in certain packet routing problems, where the initial distribution of packets may be irregular. Routing is performed by first redistributing the packets among the processors in a balanced manner, and then invoking standard routing techniques such as permutation routing or k-k-routing [25]. Much of the extensive literature on distributed load balancing, in fact, study static load balancing.

[*] AT&T Labs – Research, Florham Park, NJ. Email: `muthu@research.att.com`. Part of this work was done when the author was at Bell Labs.

[†] College of Computer Science, Northeastern University, Boston MA 02115. Email: `rraj@ccs.neu.edu`. Part of this work was done while the author was at DIMACS Center, Rutgers University, Piscataway, NJ 08854-8818. DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999 and partially supported by the New Jersey Commission on Science and Technology. Part of this work was done when the author was at the University of Texas at Austin, with support from the National Science Foundation under Grant No. CCR–9504145.

In dynamic load balancing, the load is dynamic, that is, the total workload may vary with time[1]. Dynamic load balancing is required in a wide variety of applications, including operating systems [12, 20], combinatorial optimization problems [19], and adaptive mesh partitioning [16, 31]. The results and techniques of static load balancing are applicable for certain problems in which the computation can be divided into alternating phases of balancing and processing. For many applications, however, it is desired to have a continuous process that manages the distribution of load among nodes. For example, while scheduling jobs in a distributed network, the particular sequence of job arrivals is not available in advance, and hence, the assignment of jobs to processors has to be performed on-line. Similarly, in many parallel mesh computations, the mesh regions are continuously refined, or coarsened, thereby modifying the number of mesh points (and hence the load) in different mesh regions (for a more elaborate description, see [31]).

Due to the potentially arbitrary nature of the on-line load arrival and departure process, dynamic load balancing is substantially more challenging than the static version. In order to make the study of dynamic load balancing somewhat tractable, most of the previous work has assumed either a particular statistical model of load variation or a specific network topology (for example, see [21, 30]). In this paper, we initiate an *adversarial* study of dynamic load balancing for *arbitrary network topologies*.

Our Results. We model a distributed system by an arbitrary network in which the nodes represent the processors and the edges represent the communication links. We assume that the load consists of independent *tokens* that may be processed anywhere. We adopt the standard multi-port unit capacity model of communication [3, 14, 22, 26], whereby each node can send or receive at most *one* token along each of its incident links.

An important issue arises in this formulation: *how do we best model the load variation process?* In our model, the arrival and departure of load is controlled by an *adversary* that determines the locations and the number of tokens that are added or deleted at any time. A natural objective in our adversarial model is to maintain a low imbalance in the load distribution at all times. We characterize this objective by the requirement that the maximum imbalance in load be bounded at all times. We refer to an algorithm that satisfies the preceding objective as a *stable* algorithm.

Clearly, some restrictions need to be placed on the adversary to disallow scenarios in which an unbounded amount of imbalance may be trivially created by adding, for example, a large number of tokens at a single node. Therefore, given any subset S of nodes, we place an upper bound on the amount of imbalance that an adversary can create within S; the particular upper bound depends on the *expansion* of set S. See Section 2 for a formal definition of our adversary; as we argue there, this adversary is more powerful than alternative natural formulations of an adversary. We think of this clean formulation of a powerful adversary as one of our contributions in this paper.

Our main technical contribution is a proof that a *simple local control algorithm* is stable under our adversarial model. Each step of the algorithm is simply the following (here, d is the maximum degree of the underlying network): For each edge (u, v), if u has at least $2d + 1$ tokens more than v, then u sends a token to v. We refer to this algorithm as the *local balancing algorithm*. To prove the stability of the local balancing algorithm, we define an appropriate potential function and argue that if the potential at the start of a step is above a certain "threshold" then the increase in the potential due to the adversary (namely, adding and removing tokens) is no larger than the decrease in the potential obtained by running one step of the local balancing algorithm. Our analysis relies on partitioning the nodes of the graph into groups based on the tokens they have, and then identifying the strategy for

[1]While we have used the terms "static" and "dynamic" to characterize load, some papers in the load balancing literature use the term to characterize the load balancing algorithms. These papers, (e.g., [30]) define a static load balancing algorithm (resp., dynamic load balancing algorithm) to be an algorithm in which the decisions of transferring load does not depend (resp., may depend) on the current system state.

the adversary to change the token distribution amongst the groups so as to minimize the net decrease in total potential. Our results also extend to other models that take into consideration bursty traffic or a single-port mode of communication.

Related Work. Static load balancing has been studied extensively for the arbitrary network model [3, 14, 22, 26]. Here the question is how quickly can the imbalance be reduced to a small quantity. The local algorithm we described above was proposed in this context [3], and it is known to take asymptotically optimal number of steps to balance the load [14].

Our model and our results on stability for arbitrary networks are in the same spirit as previous work on adversarial models for a different problem, namely packet routing [5, 10]. Also related is the work of Awerbuch, Kutten, and Peleg who introduce a novel extension of competitive analysis for dynamic job scheduling in arbitrary networks [7]. They develop a distributed algorithm whose competitive ratio is polylogarithmic in the size of the network. Their results are not applicable in our model, however, because their model places no bound on the amount of information or load that can be sent along any edge at any time. In addition, their work involves certain non-local operations that are inefficient in our model. (See [4, 13, 27] for more work on this problem.) The unbounded versus unit capacity assumption on the model of communication makes for a big difference in the flavor of the problems. For instance, cut capacity provides an upper bound on the number of tokens (amount of flow) that *any* algorithm may remove from a vertex set (in any given step) in the unit capacity case, while this does not apply in the unbounded edge capacity case.

In [8, 9], Awerbuch and Leighton study the multi-commodity flow problem and present an elegant local algorithm for continuously shipping a set of commodities from the sources to their respective sinks. Their results do not apply to our problem since they allow only a fixed number of tokens to be inserted/deleted at a fixed set of nodes (sources or sinks) in each step; in contrast, the bulk of our difficulty in the analysis comes from coping with a clever adversary that may distribute tokens amongst any subset of nodes in each step. On the other hand, while our work involves one commodity only, the significance of [8, 9] lies in the fact that they consider multiple commodities.

A common theme that our work shares with [8, 9] and a growing body of work including [3, 14, 15] is that a number of basic network problems admit efficient solutions in the form of local control algorithms. Several practical systems employ the local balancing algorithm to balance load. See [24] for an excellent survey and [1, 2] for a bibliography of such systems.

In recent work, subsequent to the first publication of our work [23], Anshelevich et al [6] provide an improved analysis for a class of local balancing algorithms, that includes the algorithm considered in this paper, and also introduce a multicommodity variant of dynamic load balancing.

2 Our Model

We represent the network by an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of bidirectional links. The network load is modeled by tokens, each of which may be processed at any node of the network. In order to balance the load distribution, tokens may be transferred among different nodes. We adopt the standard synchronous multi-port model for this communication [3, 14, 22, 26]: in each step, each node can send or receive at most one token along each of its incident links.

In order to study the dynamic aspect of load balancing, we introduce an *adversarial* model. In this model, tokens are created and/or destroyed at the various nodes in each step, and an adversary decides the number and location of these tokens. The goal of the balancing algorithm is to keep the imbalance as small as possible at all time.

Before formally defining the adversary, we need some additional notation. Let $w_t(v)$ denote the number of tokens at node v at the start of step t. For any subset S of V, let $w_t(S)$ denote the total number of tokens in S at the start of step t. Let the average number of tokens $(w_t(V)/|V|)$ at the start of step t be denoted by a_t. We define the *imbalance* of G at the start of step t to be $\max\{|w_t(v) - a_t| : v \in V\}$.

The Adversary. We divide each step into two phases. In the first phase, one "step" of the balancing algorithm is executed. In the second phase of each step, an adversary inserts and deletes tokens from the network. Let $e(S)$ denote the number of edges coming out of a set S of nodes. Let $d_t(S)$ denote the net increase in the number of tokens at nodes in set S in the second phase of step t. (Note that $d_t(S)$ may be negative.) An adversary with *rate* r, where $0 \le r \le 1$, can insert and delete any number of tokens on any subset of nodes subject to the following constraint for every subset S of nodes:

$$|d_t(S) - (a_{t+1} - a_t)|S|| \le r \cdot e(S) \tag{1}$$

Under this adversarial model, we seek stable balancing algorithms, which are formally defined as follows. We say that an algorithm $\mathcal{A}$ is *stable* for rate r if there exists a Δ independent of t such that for all $t \ge 0$, the imbalance of G with respect to A at the start of step t is at most Δ. In this paper, we restrict our attention to a study of stability and do not attempt to optimize the associated value of Δ.

3 The Power of the Adversary

As formulated in Section 2, the rate of the adversary is required to be at most 1. This upper bound on the rate is necessary since a straightforward argument based on edge-cuts shows that there is no stable load balancing algorithm for $r > 1$. In Section 4, we show that the local balancing algorithm is stable for all $r < 1$, thus settling this problem for all values of r but $r = 1$.

The particular formulation of the adversary is largely motivated by the need for a model under which strong performance guarantees for dynamic load balancing can be proved. To illustrate this, consider replacing Equation 1 by the following more "uniform" constraint where α is the *edge expansion* of the network:

$$|d_t(S) - (a_{t+1} - a_t)|S|| \le r \cdot \alpha \cdot |S| \tag{2}$$

Again for $r > 1$, there can be no stable load balancing algorithm because there exists a set S^* such that $e(S^*) = \alpha|S^*|$ and the adversary would choose to add all the tokens to that set each step and ensure that greater than $\alpha|S^*|$ tokens must leave S^* in each step. The adversary under Equation 1 is stronger than an adversary under Equation 2, however, since the upper bound on $d_t(S)$ in Equation 1 is dependent on the expansion of the particular set S rather than the expansion of the *smallest* expanding subset of G. Formally put, since $e(S) \ge \alpha|S|$, it follows that Equation 1 implies Equation 2.

Another feature of Equation 1 is that the upper bound is placed on the increase in imbalance rather than the actual number of tokens that are added to or deleted from any set S. Consequently, there is no absolute upper (resp., lower) bound on the number of tokens that can be added (resp., deleted) by the adversary in any step. We finally note that since the deletion of tokens is controlled by the adversary, the model allows for realistic heterogeneous scenarios in which the load may consist of different kinds of jobs and the processors may have different processing speeds.

4 Stability of the local balancing algorithm

In step t of the local balancing algorithm, each node u executes the following operation: for each edge (u, v), if $w_t(u) - w_t(v) \ge 2d + 1$, then u sends a token to v. This section is devoted to the proof of the following theorem.

Theorem 1 *For any $\varepsilon > 0$, the local balancing algorithm is stable for rate $1 - \varepsilon$.*

To prove the stability of the local balancing algorithm, we adopt a standard potential function argument. We define a quadratic potential function and argue that if the potential at the start of a step is above a certain "threshold" then the increase in the potential due to the adversary (namely, adding and removing tokens as per Equation 1), is no larger than the decrease in the potential, obtained by running one step of the local balancing algorithm. We remark that previous analyses of the local balancing algorithm [3, 14, 15] characterized the performance in terms of the expansion of the network. Therefore, the lower bounds that these analyses derive for the potential drop during a single step of the local balancing algorithm are limited by the least-expanding subset of the network. As a result, these lower bounds are too weak to establish stability against an adversary that has the freedom to overload any subset of nodes in the network.

We now describe the potential function used in our analysis. At the start of step t, for each node v, we assign a potential $\phi_t(v)$ of $(w_t(v) - a_t)^2$. We define the *height* $h_t(v)$ of a node v at the start of step t to be $(w_t(v) - a_t)$. Let Φ_t denote the sum of the potentials of all of the nodes. Let V_t^+ (resp., V_t^-) denote the set of nodes with nonnegative (resp., negative) heights at the start of step t. Let Φ_t^+ (resp., Φ_t^-) denote the sum of the potentials of all of the nodes in V_t^+ (resp., V_t^-). We note that Φ_t equals $\Phi_t^+ + \Phi_t^-$. Let $w_t'(v)$ denote the number of tokens at v at the start of the adversarial phase of step t. Let $\phi_t'(v)$ denote $(w_t'(v) - a_t)^2$.

We prove the stability of the local balancing algorithm by placing time-independent upper bounds on both Φ_t^+ and Φ_t^-. The key step in our analysis is the following lemma.

Lemma 4.1 *If there exists a node with height at least $13n^2d^2/(2\varepsilon)$ at the start of step t, then Φ_{t+1}^+ is at most Φ_t^+. If there exists a node with height at most $-13n^2d^2/(2\varepsilon)$ at the start of step t, then Φ_{t+1}^- is at most Φ_t^-.*

Proof: We only prove the first claim of the lemma. The proof of the second claim is symmetric.

It is useful to extend the notion of height to tokens as well. For this purpose, we assign, for every node v, a unique *rank* from $[1, w_t(v)]$ to each token at v. Let the height of a token be its rank minus a_t. We note that for any node v with nonnegative height, $\phi_t(v)$ is the sum, over all the tokens x with positive height $h_t(x)$, of the term $2h_t(x) - 1$.

Assume that there exists a node with height at least $13n^2d^2/(2\varepsilon)$ at the start of step t. We divide V_t^+ into distinct sets in the following way. For any i in $\mathbf{N}$, if $\cup_{0 \le j < i} S_j$ is not equal to V_t^+, then let S_i denote the minimal nonempty set of nodes such that for all u in S_i and v in $V_t^+ \setminus \cup_{0 \le j \le i} S_j$, $w_t(v) - w_t(u)$ is at least $4d$. Let k be the maximum value of i for which S_i is defined. Since $13n^2d^2/(2\varepsilon) > 4nd$, k is positive. Let $S_{\ge i}$ denote $\cup_{j \ge i} S_j$. Let u_i and ℓ_i denote $\max_{u \in S_i}(w_t(u) - a_t)$ and $\min_{u \in S_i}(w_t(u) - a_t)$, respectively.

We first study the balancing phase. Consider a token x transferred from a node u in S_i to a node v not in S_i. Since a token never gains height, the edge (u, v) belongs to the cut $(S_{\ge i}, V \setminus S_{\ge i})$. In fact, (u, v) belongs to the cut $(S_{\ge j}, V \setminus S_{\ge j})$ for each $j \le i$ such that v is not in $S_{\ge j}$. (For example, if v is not in $V_t^+ = S_{\ge 0}$, then (u, v) belongs to $(S_{\ge j}, V \setminus S_{\ge j})$ for all $j \le i$.) A lower bound on the drop in the potential of the nodes in V_t^+ as a result of the transfer of the token x is given by $\sum_{(u,v) \in (S_{\ge j}, V \setminus S_{\ge j})} (\ell_j - u_{j-1} - 2d)$. Thus, for any $i > 0$, the potential drop due to the cut $(S_{\ge i}, V \setminus S_{\ge i})$ is at least $2e(S_{\ge i})(\ell_i - u_{i-1} - 2d)$.

During the balancing phase, in addition to the potential drop, there may be a positive contribution to the potential of the nodes with nonnegative heights by nodes from $V \setminus V_t^+$ which gain tokens and achieve nonnegative height. Since each node gains at most d tokens, this positive contribution to the potential is at most nd^2. Thus, we have the total potential drop in the balancing phase to be at least

$$-nd^2 + \sum_{i > 0} 2e(S_{\ge i})(\ell_i - u_{i-1} - 2d). \tag{3}$$

Let us now consider the adversarial phase in which the potential may increase due to the tokens added by the adversary. The potential of any node v in V_t^+ at the start of step $t+1$ is $(w_{t+1}(v)-a_{t+1})^2 = ((w'_t(u) - a_t) + (d_t(u) - (a_{t+1} - a_t)))^2$. Thus, the potential at the start of step $t+1$ due to nodes in V_t^+ is at most

$$\begin{aligned}
& \sum_i \sum_{u \in S_i} \phi'_t(u) + (d_t(u) - (a_{t+1} - a_t))^2 \\
& + 2(w'_t(u) - a_t)(d_t(u) - (a_{t+1} - a_t)) \\
\leq \; & (\sum_i (\phi'_t(S_i) + \sum_{u \in S_i} 2(w'_t(u) - a_t)(d_t(u) - (a_{t+1} - a_t))) \\
& + nd^2 \\
\leq \; & (\sum_i (\phi'_t(S_i) + \sum_{u \in S_i} 2u_i(d_t(u) - (a_{t+1} - a_t))) \\
& + 8n^2d^2 + 3nd^2.
\end{aligned}$$

The last step follows from the following observations: (i) $|d_t(u) - (a_{t+1} - a_t)| \leq d$ for all u by Equation 1; (ii) $\ell_i - d \leq w'_t(u) - a_t \leq u_i + d$ for $u \in S_i$; (iii) $u_i - \ell_i \leq 4nd$ for all i. Therefore, $(w'_t(u) - a_t)(d_t(u) - (a_{t+1} - a_t))$ is at most $u_i(d_t(u) - (a_{t+1} - a_t)) + 4n^2d^2 + d^2$.

In addition to nodes in V_t^+ that remain in V_{t+1}^+, there may be nodes that do not belong to V_t^+ but belong to V_{t+1}^+. By Equation 1, the potential of any such node is at most d^2. Thus, the total increase in potential in the adversarial phase is at most

$$2nd^2 + \sum_i 2u_i(d_t(S_i) - (a_{t+1} - a_t)|S_i|). \tag{4}$$

It follows from Lemma 4.2 below that the right hand side of Equation 4 is maximized if the adversary adds as many tokens to S_k as the constraint in Equation 1 allows, then adds as many tokens to S_{k-1} as the constraint allows, and continues in this manner to add tokens to S_i in the order of decreasing i. Lemma 4.2 thus implies that an upper bound on the increase in potential is obtained by making the following substitution in Equation 4 for all i in $[k+1]$

$$d_t(S_i) - (a_{t+1} - a_t)|S_i| = (1 - \varepsilon)(e(S_{\geq i}) - e(S_{>i})). \tag{5}$$

By Equations 3, 4, and 5, we obtain that the net decrease in potential is at least

$$\begin{aligned}
& -8n^2d^2 - 5nd^2 - \sum_{i \geq 0} 2u_i(1-\varepsilon)(e(S_{\geq i}) - e(S_{>i})) \\
& + \sum_{i>0} 2e(S_{\geq i})(\ell_i - u_{i-1} - 2d) \\
= \; & -8n^2d^2 - 5nd^2 - 2u_0(1-\varepsilon)e(S_{\geq 0}) \\
& + \sum_{i>0} 2e(S_{\geq i})(\ell_i - (1-\varepsilon)u_i - \varepsilon u_{i-1} - 2d) \\
= \; & -8n^2d^2 - 5nd^2 - 2u_0(1-\varepsilon)e(S_{\geq 0}) - \sum_{i>0} 2e(S_{\geq i})(u_i - l_i) \\
& + \sum_{i>0} 2e(S_{\geq i})(\varepsilon(u_i - u_{i-1}) - 2d) \\
\geq \; & -12n^2d^2 - 5nd^2 + \sum_{i>0} 2e(S_{\geq i})(\varepsilon(u_i - u_{i-1}) - 2d) \\
\geq \; & -12n^2d^2 - 5nd^2 + 2\varepsilon(u_k - 4nd) - 2nd^2 \\
= \; & -(12n^2d^2 + 7nd^2 + 8\varepsilon nd) + 2\varepsilon u_k \\
\geq \; & -13n^2d^2 + 2\varepsilon u_k,
\end{aligned}$$

for n sufficiently large. (In the first and second inequalities, we rearrange the summands. For the third inequality we note that: (i) $u_i - \ell_i$ is at most $4nd$ for all d, and (ii) the total number of edges in the network is at most $nd/2$. For the fourth inequality, we note that: (i) $e(S_{\geq i})$ is at least one for all i, (ii) $e(S)$ is at most $nd/2$ for all S, and (iii) u_0 is at most $4nd$.)

Since there exists a node with height at least $13n^2d^2/(2\varepsilon)$, u_k is at least $13n^2d^2/(2\varepsilon)$. Hence, the net decrease in potential is nonnegative. ■

In the following lemma, we use the notation $\langle x \rangle$ to denote a sequence $x_0, x_1, \ldots,$ of reals.

Lemma 4.2 *Let $\langle \alpha \rangle$ be a sequence of k nonincreasing nonnegative reals. Let $\langle \beta \rangle$ be a sequence of k reals. Then, an optimal solution to the linear program $\mathcal{P}$*

$$\begin{array}{ll} \text{maximize} & \sum_{i \in [k]} \alpha_i x_i \\ \text{subject to} & \sum_{i \in [j]} x_i \leq \beta_i, \textit{for all } j \textit{ in } [1,k), \end{array}$$

is obtained when x_0 is β_0 and x_i is $\beta_i - \beta_{i-1}$ for all i in $[1,k)$.

Proof: Given any solution $\langle y \rangle$ to $\mathcal{P}$ and any i in $[k]$, we say that i is *good* if $\sum_{0 \leq j \leq i} y_j$ equals β_i. Let $\langle x^* \rangle$ be defined as follows: x^*_0 is β_0 and x^*_i is $\beta_i - \beta_{i-1}$ for all i in $[1,k)$. We note that $\langle x^* \rangle$ is the unique solution that has k good indices.

Given an optimal solution $\langle z \rangle$ to $\mathcal{P}$ that has fewer than k good indices, we construct a new optimal solution $\langle z' \rangle$ that has more good indices than $\langle z \rangle$. This construction suffices to establish the lemma.

Let ℓ be the largest index that is not good; thus, ℓ is the largest index such that $\sum_{0 \leq j \leq \ell} z_j$ is less than β_ℓ. If ℓ is $k-1$, then we set z'_j to z_j for all j in $[k-1]$ and z'_{k-1} to $\beta_{k-1} - \sum_{0 \leq j < k-1} z_j$. We note that $\langle z' \rangle$ is a feasible solution to $\mathcal{P}$. Also, $\langle z' \rangle$ is an optimal solution since $z'_{k-1} > z_{k-1}$ and α_{k-1} is nonnegative. Moreover, the number of good indices of $\langle z' \rangle$ is one more than that of $\langle z \rangle$.

If ℓ is less than $k-1$, then we set z'_j to z_j for all j in $[k] \setminus \{\ell, \ell+1\}$. Let δ denote the term $\beta_\ell - \sum_{0 \leq j < \ell} z_j$. We set z'_ℓ to $z_\ell + \delta$ and $z'_{\ell+1}$ to $z_{\ell+1} - \delta$. It follows that $\langle z' \rangle$ is a feasible solution to $\mathcal{P}$. Also, $\langle z' \rangle$ is an optimal solution since

$$\begin{aligned} \sum_{0 \leq i < k} \alpha_i z'_i &= (\alpha_\ell - \alpha_{\ell+1})\delta + \sum_{0 \leq i < k} \alpha_i z_i \\ &\geq \sum_{0 \leq i < k} \alpha_i z_i, \end{aligned}$$

where the last inequality holds since $\alpha_\ell \geq \alpha_{\ell+1}$. An index i in $[k] \setminus \{\ell\}$ is good for solution $\langle z \rangle$ if and only if i is good for solution $\langle z' \rangle$. Moreover, ℓ is a good index for $\langle z' \rangle$. Therefore, the number of good indices of $\langle z' \rangle$ is one more than that of $\langle z \rangle$. This completes the proof of the desired claim. ■

Proof of Theorem 1: We now show that for all t, Φ^+_t and Φ^-_t are both at most $n(13n^2d^2/(2\varepsilon) + d)^2$. Let, if possible, t be the first step such that Φ^+_t is greater than $n(13n^2d^2/(2\varepsilon) + d)^2$. Therefore, there exists at least one node v such that $u_t(v)$ is at least $(13n^2d^2/(2\varepsilon) + d)$. Since the height of a node increases by at most d in the balancing phase and by at most d in the adversarial phase, we obtain that $u_t(v) \leq u_{t-1}(v) + 2d$ for all v. Therefore, for all v in V, $u_{t-1}(v)$ is at least $13n^2d^2/(2\varepsilon)$. This implies that by Lemma 4.1, Φ^+_{t-1} is at least Φ^+_t, which contradicts our choice of t. A symmetric argument establishes that Φ^-_t is at most $n(13n^2d^2/(2\varepsilon) + d)^2$.

Hence, the imbalance at the start of any step t is at most $\sqrt{n(13n^2d^2/(2\varepsilon) + 2d)^2}$, which is at most $7n^{5/2}d^2/\varepsilon$ for n sufficiently large. This establishes the stability of the local balancing algorithm. ■

5 Extensions

We now present two variants of the adversarial model for which our results can be extended. In both cases, proving appropriate versions of Lemma 4.2 is the crux.

Bursty Load. We can modify the model of Section 2 to account for bursty changes in the load. Instead of limiting the adversary by the constraint of Equation 1, we allow the adversary to add and delete token arbitrarily, subject to the constraint that for some fixed integer w, the following inequality is satisfied for all t and S

$$\sum_{t-w<t'\leq t} |d_{t'}(S) - (a_{t'+1} - a_{t'})|S|| \leq r \cdot w \cdot e(S). \tag{6}$$

In other words, while the ratio $|d_{t'}(S) - (a_{t'+1} - a_{t'})|S||/(w \cdot e(S))$ need not be at most r for any given step t', the average of the ratio for any window of w steps must be at most r. Our definition for the bursty load pattern is based on the definitions of bursty packet traffic in [5, 28]. Our analysis in Section 4 can be modified to show that the local balancing algorithm is stable under bursty adversaries as well.

Single-port Communication. The model of Section 2 assumes multi-port communication; i.e., in each step each node can send and/or receive at most one token along each of its incident edges. A more suitable communication model for certain applications is the *single-port* model. In the single-port model, each node can send and/or receive a token along at most one of its incident edges. Since communication is typically assumed to be in handshaking mode, this implies that at any step, tokens may only move along a set of edges that form a graph matching. The single-port model has been studied in [11, 15, 14, 17, 18]. We define an adversarial model for the single-port model in much the same way as for the multi-port model except that Equation 1 is replaced by the following inequality.

$$|d_t(S) - (a_{t+1} - a_t)|S|| \leq r \cdot m(S), \tag{7}$$

where $m(S)$ is the size of the maximum matching between the sets S and $V \setminus S$. A variant of the local balancing algorithm for the single-port model was studied in [15]. Each step of this algorithm consists of two phases. In the first phase the nodes compute a random matching. In the second phase, for each node u, if edge (u, v) is chosen in the matching and $w(u) > w(v) + 1$, then u sends a token to v. In [15], it is shown that the probability that an edge appears in the random matching of any step is at least $1/(8d)$. Using this property of the random matching, we can modify the analysis in Section 4 and prove that if $r < 1/8$, then the expected global imbalance at any given step is bounded. On the other hand, it trivially follows that if $r > 1$, then no load balancing algorithm is stable in this model.

6 Concluding Remarks

We have established the stability of the local balancing algorithm for any rate less than 1 and for any arbitrary network. In recent work [6], it has been shown that the local balancing algorithm is stable for rate 1 as well, thus settling our conjecture in [23]. A direction for future research is to consider variations of the adversarial model that may be suitable for job scheduling applications involving heterogeneous processors (e.g., processors working at different speeds and capable of executing different subsets of jobs) and heterogeneous workload (e.g., jobs of varying sizes).

Acknowledgments

The authors wish to thank Elliot Anshelevich for pointing out a calculation mistake in an earlier version of the paper.

References

[1] The collection of computer science bibliographies: Bibliography on load balancing. At URL http://liinwww.ira.uka.de/bibliography/Parallel/ Load.Balance.1.html.

[2] References to online documents about process migration, checkpointing and load balancing. At URL http://wwwbode.informatik.tu-muenchen.de/~petri/pbeamrefs.html.

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

[4] N. Alon, G. Kalai, M. Ricklin, and L. Stockmeyer. Lower bounds on the competitive ratio for mobile user tracking and distributed job scheduling. *Theoretical Computer Science*, 130:175–201, 1994.

[5] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 380–389, October 1996.

[6] E. Anshelevich, D. Kempe, and J. Kleinberg. Stability of load balancing algorithms in dynamic adversarial systems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, May 2002. To appear.

[7] B. Awerbuch, S. Kutten, and D. Peleg. Competitive distributed job scheduling. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 571–580, May 1992.

[8] B. Awerbuch and F. T. Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 487–496, May 1994.

[9] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multi-commodity flow. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468, October 1993.

[10] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queueing theory. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 376–385, May 1996.

[11] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 2:279–301, 1989.

[12] D. Eager, D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 12:662–675, 1986.

[13] P. Fizzano, D. Karger, C. Stein, and J. Wein. Job scheduling in rings. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 210–219, June 1994.

[14] 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*, 29:29–62, 1999.

[15] B. Ghosh and S. Muthukrishnan. Dynamic load balancing in parallel and distributed networks by random matchings. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 226–235, June 1994.

[16] A. Heirich and S. Taylor. A parabolic theory of load balance. Technical Report Caltech-CS-TR-93-22, Caltech Scalable Concurrent Computation Lab, March 1993.

[17] J. Hong, M. Chen, and X. Tan. Dynamic cyclic load balancing on hypercubes. In *Proceedings of the 4th Conference on Hypercubes, Concurrent Computers and Applications*, pages 595–598, 1989.

[18] S. H. Hosseini, B. Litow, M. Malkawi, J. McPherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *Journal of Parallel and Distributed Computing*, 10:160–166, 1990.

[19] E. L. Lawler and D. E. Wood. Branch and bound methods: a survey. *Operations Research*, 14:699–719, 1966.

[20] F. C. H. Lin and R. M. Keller. The gradient model load balancing method. *IEEE Transactions on Software Engineering*, 13:32–38, 1986.

[21] M. Livny and M. Melman. Load balancing in homogeneous broadcast distributed systems. *ACM Performance Evaluation Review*, 11(1):47–55, 1982.

[22] F. Meyer auf der Heide, B. Oesterdiekhoff, and R. Wanka. Strongly adaptive token distribution. *Algorith-*

mica, 15:413–427, 1996.

[23] 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*, pages 47–54, June 1998.

[24] M. Nuttall. Survey of systems providing process or object migration. Technical Report DoC94/10, Imperial College, London, 1994.

[25] D. Peleg and E. Upfal. The generalized packet routing problem. *Theoretical Computer Science*, 53:281–293, 1987.

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

[27] C. Phillips, C. Stein, and J. Wein. Task scheduling in networks. In *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory*, pages 290–301, 1994.

[28] Y. Rabani and É. Tardos. Distributed packet switching in arbitrary networks. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pages 366–375, May 1996.

[29] B. Shirazi, A. Hurzon, and K. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Society Press, 1995.

[30] A. N. Tantawi and D. Towsley. Optimal static load balancing in distributed computer systems. *Journal of the ACM*, 32:445–465, 1985.

[31] R. D. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience*, 3:457–481, 1991.