Load Balancing with Bounded Convergence in Dynamic Networks

Michael Dinitz* Johns Hopkins University
Jeremy Fineman† Georgetown University
Seth Gilbert‡ National University of Singapore
Calvin Newport§ Georgetown University

***Abstract*—Load balancing is an important activity in distributed systems. At a high-level of abstraction, this problem assumes processes in the system begin with a potentially uneven assignment of "work" which they must then attempt to spread evenly. There are many different approaches to solving this problem. In this paper, we focus on local load balancing—an approach in which work is balanced in an iterative and distributed manner by having processes exchange work in each round with neighbors in an underlying communication network. The goal with local load balancing is to prove that these local exchanges will lead over time to a global balance. We describe and analyze a new local load balancing strategy called max neighbor, and prove a bound on the number of rounds required for it to obtain a parameterized level of balance, with high probability, in a general dynamic network topology. We then prove this analysis tight (within logarithmic factors) by describing a network and initial work distribution for which max neighbor matches its upper bound, and then build on this to prove that no load balancing algorithm in which every node exchanges work with at most one partner per round can converge asymptotically faster than max neighbor.**

I. Introduction

Load balancing is an important activity in distributed systems. At a high-level of abstraction, this problem assumes processes in the system begin with a potentially uneven assignment of "work" which they must then attempt to spread evenly. Load balancing is relevant to a variety of scenarios, from distributing tasks in multiprocessor systems [11], to optimizing virtual machine placement in data centers [15], to enabling fairness in cooperative downloading/uploading in mobile peer-to-peer networks [6], [23], [10]. Accordingly, there exist a variety of different approaches to solving this problem.

This paper focuses on *local* load balancing—an approach in which work is balanced in an iterative and distributed manner by having processes exchange work in each round with neighbors in an underlying communication network. The rules determining these exchanges tend to be simple. The goal is to prove that they will lead over time to a global balance.

Local load balancing was first formalized and analyzed mathematically by Cybenko [11] and Boillat [8]. It has since been well-studied by the algorithms community in both *static* [11], [8], [14], [29], [28], [30] and *dynamic* [3], [4], [5], [24], [13] network topologies. In this paper, we describe a natural local load balancing strategy that satisfies the useful property that each process connects with at most one nearby process per round. We then prove a concrete bound on the time required for it to reach a parameterized level of convergence. Our analysis assumes an arbitrary dynamic network. We then prove our strategy's performance near optimal (i.e., within logarithmic factors) with a pair of lower bound results.

Our Results. In this paper, we study a natural local load balancing strategy called *max neighbor*. This strategy has nodes attempt, in each round, to balance their current workload with the single neighbor in their network neighborhood with the most different workload. For example, if you have workload 5, and have three neighbors with workloads 1, 6, and 15, respectively, this strategy would have you attempt to balance your load with the third neighbor from this list—leaving you both with workload $(5 + 15)/2 = 10$. We emphasize that this strategy is *memory-free* in the sense that the only state a node must maintain between rounds is its current workload. We also emphasize that it satisfies the strong property that each node connects with at most one neighbor per round. As noted in Section II, this property is desirable as many distributed systems cannot support simultaneous work transfers with multiple neighbors (c.f., the discussion in [11]).

We study the max neighbor strategy in dynamic networks that guarantee only that the network is connected in every round (and can otherwise change arbitrarily). Given any network size $n \geq 1$, total amount of work in the system $T \in \mathbb{R}_{\geq 0}$, and *convergence factor* $\tau \in \mathbb{R}_{>0}$, we analyze how many rounds are required before the max neighbor strategy guarantees (with high probability in n) that every pair of workloads in the system are within an additive factor of τ; i.e., for every pair of workloads x, y: $|x - y| \leq \tau$. In more detail, we prove that this strategy achieves this parameterized convergence in the following number of rounds:

$$O\left(\min\left\{n^2 \log\left(\frac{Tn}{\tau}\right) \log n, \frac{Tn \log n}{\tau}\right\}\right).$$

Note that these two bounds essentially coincide at $\tilde{O}(n^2)$ with $\tau = \Theta(T/n)$, where the notation $\tilde{O}$ hides logarithmic factors. In other words, if we want all nodes to have the same workload up to a constant factor, the max neighbor strategy uses $\tilde{O}(n^2)$ rounds. We show that our analysis is tight by identifying a network and initial work distribution for which max neighbor does require $\Omega(n^2)$ rounds when $\tau = \Theta(T/n)$.

*Supported in part by NSF awards 1464239 and 1535887.
†Supported in part by NSF grants CCF-1314633 and CCF-1617727.
‡Supported in part by Singapore MOE Tier 2 ARC project 2014-T2-1-157.
§Supported in part by NSF grant CCF-1320279.

We then build on this result to establish a more general lower bound that establishes that $\Omega(n^2)$ rounds are required for a large family of load balancing strategies that include max neighbor (in more detail, this family includes all strategies that allow each node to exchange work with at most one node per round).

Techniques. To achieve our time bounds, we borrow general techniques from Lyapunov optimization—a well-used approach in the study of dynamical systems. In more detail, we capture a measure of the current amount of imbalance in the system in a given round r with a potential function $\phi(r)$ that maps the vector of workloads in r to a single scalar value. We then prove that this function decreases by an amount in $\Omega(D_r)$ during this round, where D_r is the sum of the size of the workload gaps that are averaged in r.

Once we bound the behavior of ϕ, we establish that, with constant probability, D_r is relatively large. In more detail, we identify a path in G_r such that the sum of the workload gaps represented by its edges is large. We then prove that a constant fraction of these edges will either connect and exchange their workload, or belong to a small graph neighborhood in which an edge will connect and exchange at least as much workload. Given these tools we can then bound the rounds required for ϕ to decrease from its theoretical maximum value down to τ—at which point convergence is complete. We emphasize that this approach is different than many existing load balancing strategies which instead tend to model a balance strategy by repeated matrix multiplication and leverage spectral techniques to prove convergence.

II. Related Work

Load balancing on static graphs. In his seminal 1989 paper, Cybenko [11] noted that in distributed multiprocessor systems *static* load balancing (i.e., where work is divided in advance) does not work well for many problems. In these cases, system designers were instead relying on simple heuristics that balanced load on the fly in a distributed and local manner (e.g., *if a nearby processor has significantly less work, pass it some of your own*). Cybenko was one of the first to capture these *local load balancing* strategies in a form amenable to mathematical analysis. He proved that a *diffusion* strategy, in which processes iteratively adjust their workload to the weighted average of the workloads in their network neighborhood, will converge to a uniform distribution in the limit. He then argued, however, that many systems cannot support diffusion strategies because these systems lack the ability to exchange work with many neighbors concurrently.

With these limits in mind, Cybenko also proposed and analyzed a strategy, called *dimension exchange*, that only requires each process to communicate with a single neighbor. The strategy, however, assumes that processes are connected in a static hypercube topology. He proved dimension exchange converges efficiently in the dimensionality of the hypercube.

In 1990, Boillat [8] independently formalized and analyzed diffusion strategies. Whereas Cybenko leveraged spectral analysis techniques in his convergence proof, Boillat interpreted diffusion as a Poisson heat equation, allowing him to leverage known techniques from Markov chain analysis. In particular, Boillat proved that diffusion combined with a somehwat complicated averaging rule (which requires nodes to solve a local optimization problem in determining their new workload in each round) converges for a static graph to uniform in $O(n^2)$ steps.

In the years that followed, many papers have built on the general foundation of Cybenko and Boillat to formally analyze the convergence properties of diffusion and dimension exchange local load balancing under different network assumptions and strategy variations. For example, Hosseini et al. [14] and Xu et al. [29], studied generalizations of Cybenko's dimension exchange strategy to more general topologies. They prove convergence in the limit. A core technique in these papers is to edge color the network and then restrict balancing in a given round to a single edge color. (This strategy is echoed in our max neighbor algorithm which uses a simpler mechanism to ensure that the connections in a given round form a matching.) In follow up work, Xu et al. [28], [30] study the optimal weighted average parameters to use for diffusion-style load balancing in a variety of network topology types—including k-ary n-cubes, n-dimensional torii, and meshes.

Load balancing on dynamic graphs. The papers cited above assume the network topology is *static*. A series of subsequent papers by Bahi et al. [3], [4], [5], by contrast, prove convergence (in the limit) of local load balancing strategies in *dynamic* network topologies. In [3], the authors prove the convergence of a dimension exchange algorithm in hypercubes with edges that can come and go. And in [4], [5], the authors prove that diffusion strategies converge (in the limit) in general network topologies with edges (and in the case of [5], also nodes) that can come and go. Bounds on the convergence rate for diffusion strategies in dynamic graphs were given in [13]

More relevant to our own work, however, is an additional result in [4], which proves that dimension exchange strategies eventually converge in dynamic networks. The max neighbor algorithm we study falls into this general class of load balancing strategies, as it has nodes connect with at most one neighbor per round. It follows from [4], therefore, that our solution converges in the limit. Our focus, however, is not on eventual convergence, but instead on upper bounding the rounds required to achieve convergence. We emphasize the techniques we use to prove these time bounds differ significantly from those used in these existing studies of eventual convergence.

Also related to our work is a paper by Sider and Couturier [24] which examines a dimension exchange strategy in which processes select neighbors in a manner that maximizes the amount of work transferred in the round. Our max neighbor strategy can be seen as a simple approximation of this optimality. Unlike our work, however, [24] does not contain convergence bounds. It instead explores (complicated) distributed algorithms that can be executed in a static graph to identify the globally optimal set of neighbors to exchange

work, and presents simulation results that imply that this general approach is effective.

Load balancing and consensus. We note that there are close connections between diffusion style load balancing and the study of *average consensus*—a problem in which all nodes start with an initial value and the goal is for all nodes to converge toward the same value (typically required to be the average). The problem originated in a 1973 paper be DeGroot [12], though it was subsequently independently identified in other fields. (Vicsek et al. [25], for example, studied particle systems in which, in each time step, each particle adopted the average heading of nearby particles—proving that particles in this system will converge to the same heading.) Many variations of this same general neighborhood averaging strategy were subsequently studied; c.f., [16], [22], [7], [26], [21], [17], [27], [9].

Dynamic network algorithms. We also note that there is a growing literature on general distributed algorithms in dynamic network models similar to the one we study in this paper (see [19] for a survey). A well-known result [18] provides an $O(n^2)$ round algorithm for all-to-all gossip in a dynamic network of size n in which nodes communicate through local broadcasts. The superficial similarity between the $O(n^2)$ result of [18] and the $\tilde{O}(n^2)$ result described in this paper hints at intriguing connections. However we have not been able to show a formal connection. It is not obvious, for example, how to implement load balancing (which requires work to be transferred in a pairwise fashion) using the gossip technique of [18], which leverages network-wide floods as its main communication strategy.

III. The Load Balancing Problem

Fix a set V consisting of $n \geq 1$ nodes, each corresponding to a probabilistic computational process. We assume time proceeds in synchronous rounds that we label $1, 2, ...$ At the beginning of round 1, each node $u \in V$ is assigned an initial workload $x_u(0)$ from $\mathbb{R}_{\geq 0}$. More generally, we use $x_u(r)$ to denote the workload of node u at the end of round r. Let $T = \sum_{u \in V} x_u(0)$ be the sum of the initial workloads. We call T the *total workload* of the system. The goal of the load balancing problem is for the nodes to coordinate to balance the total workload such that every node is committed to a workload close to the global average of T/n.

To accomplish this balancing goal, the nodes can attempt in each round to exchange some of their workload with neighbors in an underlying communication network. Formally, we describe the communication network in each round $r \geq 1$ with a connected undirected graph $G_r = (V, E_r)$. The network topology over time, therefore, is defined by a sequence $G_1, G_2, ...$ of graphs. In the *static* setting, for all $r, r' \geq 1 : G_r = G_{r'}$. In the *dynamic* setting, the graphs can change. In this paper, we assume the more general dynamic setting.

At the beginning of each round $r \geq 1$, each node $u \in V$ learns the current workload of its neighbors in G_r. That is, for each $v \in N_r(u) = \{v \mid v \text{ neighbors } u \text{ in } E_r\}$, u learns $x_v(r-1)$. At this point, u can exchange work with nodes in its neighborhood. In more detail, we assume that each node u can send *connection proposals* to any subset of its neighborhood. It can also choose to *accept* any subset of the connection proposals it receives from its neighborhood. If a node v accepts a connection proposal from neighbor u, then v and u can exchange work. The logic that specifies how nodes decide which neighbors to connect with and how to exchange work over connections is called a *load balancing strategy* (equiv., "load balancing algorithm") in the following.

We now formalize what it means for a particular load balancing strategy to *solve* the load balancing problem in our model. Fix some *convergence factor* $\tau \in \mathbb{R}_{>0}$. We say a given round $r \geq 0$ is **τ-converged** if for all $u, v \in V$, $|x_u(r) - x_v(r)| \leq \tau$. In this paper, we analyze the time a given load balance strategy requires to guarantee τ-convergence (with high probability), for a network size, total workload, and convergence factor. Formally:

Definition III.1. *We say a load balancing strategy* solves the load balancing problem with convergence factor τ in $f(n, T, \tau)$ rounds*, for some* $f : \mathbb{Z}_{>0} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{>0} \to \mathbb{N}$*, if and only if for every network size* n*, total workload* T*, and convergence factor* τ*, with high probability in* n*: for every round* $r \geq f(n, T, \tau)$*, round* r *is* τ*-converged.*

We sometimes use the natural generalization of the above definition by allowing the possible values of n, T, and/or τ to be restricted.

IV. Max Neighbor Load Balancing

In this paper, we study a natural load balancing strategy (also called an "algorithm" in the following) in which nodes attempt to balance their workload with the neighbor representing the largest current workload gap. Our algorithm requires no advance knowledge of n, T, or τ, and is *memory-free* in the sense that the only state each node needs to maintain between rounds is its current workload. In more detail:

> **Max Neighbor Load Balancing.**
> In each round $r \geq 1$, each node $u \in V$ flips a fair coin to determine whether to *send* or *receive* connection proposals. If u decides to send, it selects neighbor $v = \text{argmax}_{v \in N_r(u)}\{|x_v(r-1) - x_u(r-1)|\}$ and sends v a connection proposal. If u decides to receive, and it receives one or more connection proposals, it will accept the proposal from $v = \text{argmax}_{v \in S}\{|x_v(r-1) - x_u(r-1)|\}$ (where S is the set of node(s) sending proposals to u), and we say that u and v are *connected* in round r. In both argmax decisions, ties can be broken by any fixed deterministic criteria. If two nodes u and v are connected in round r, they evenly balance their respective workloads by setting $x_u(r) = x_v(r) = \frac{1}{2}(x_u(r-1) + x_v(r-1))$.

Notice, our max neighbor algorithm satisfies the strong property that no node is involved in more than one connection per round. This allows it to be implemented in distributed systems

such as smartphone peer-to-peer meshes, or certain multiprocessor networks, in which concurrent local connections are not possible.

V. Analysis

We now analyze how long it takes the max neighbor algorithm to solve the load balancing problem for a given set of system parameters. In this section, we will prove three theorems. The first handles the trivial case where $\tau \geq T$. The second and third theorems provide two different convergence time bounds for the more interesting case where $\tau < T$. We pull out the $\tau \geq T$ case into its own theorem so that we can simplify the statements of the results that then follow. Due to space constraints, many proofs are omitted or only sketched, but full proofs can be found in the full version [1].

We first tackle the simple case where $\tau \geq T$. When the convergence factor is at least as large as the initial total workload, the initial workload is already τ-converged, and nothing we do can possibly make it not τ-converged. Hence we immediately get the following theorem.

Theorem V.1. *Fix some total workload $T > 0$, network size $n \geq 1$, and convergence factor $\tau \geq T$. Given these parameters, the max neighbor algorithm solves the load balancing problem for convergence factor τ in 0 rounds.*

A. Convergence for Large T.

Moving forward, we assume $\tau < T$. We begin by proving the below bound on convergence that is optimized for large T (as compared to n). We will later prove a bound optimized for smaller T.

Theorem V.2. *Fix some total workload $T > 0$, network size $n \geq 1$, and convergence factor τ, $0 < \tau < T$. Given these parameters, the max neighbor algorithm solves the load balancing problem for convergence factor τ in $O\left(n^2 \log\left(\frac{Tn}{\tau}\right) \log n\right)$ rounds.*

Proof Overview. Our analysis technique leverages core ideas from Lyapunov optimization. In particular, we begin by defining a *potential function*, ϕ, that measures at each round the total magnitude of workload *gaps* in the system:

$$\forall r \geq 0 : \phi(r) = \sum_{u,v \in V} |x_u(r) - x_v(r)|.$$

Similar to a Lyapunov function, ϕ maps the system state (in this case, a vector of workloads) at any given round to a non-negative scalar value that describes the desirability of the current system state. As ϕ decreases toward 0, the system state becomes more desirable in the sense that the workload in the system becomes more balanced. As in standard Lyapunov optimization, our analysis below will show that the modifications to system state caused by executing a single round of our max neighbor algorithm will drift the value of ϕ toward zero in a non-decreasing manner. To obtain our time bounds, we will establish a probabilistic lower bound for the *amount* of drift in a given round. Combining these lower bounds with an upper bound on $\phi(0)$ will provide the needed result.

Preliminaries. For a given round $r \geq 0$ and node pair $u, v \in V$, we define $d_{u,v}(r) = |x_u(r) - x_v(r)|$ to describe the gap between u and v's workload at the end of that round. For each such r, we also define: $A_r = \{\{u, v\} \mid u \text{ and } v \text{ connected and averaged their workloads in round } r\}$, i.e., the set of node pairs that connect and average in r, and $D_r = \sum_{\{u,v\} \in A_r} d_{u,v}(r-1)$, i.e., the sum of gaps averaged in r. Finally, we define $t_{max}(r) = \max_{u,v \in V}\{d_{u,v}(r)\}$ to describe the largest gap between any two nodes at the end of round r.

We now proceed with the main proof argument for Theorem V.2 in three steps that build upon each other.

Step #1: Prove that ϕ is Nonincreasing and Decreases Proportional to D_r. The goal of this step is to prove that in any execution of our algorithm, we can lower bound $\phi(r-1) - \phi(r)$ with D_r. In particular, we begin by bounding this value in the case where A_r contains only a single pair in the round in question. We will then apply that result iteratively to handle other possibilities for A_r.

Lemma V.3. *Fix some execution of the max neighbor load balancing algorithm and some round r of the execution such that $A_r = \{\{u, v\}\}$. Then $\phi(r-1) - \phi(r) \geq d_{u,v}(r-1)$.*

Proof. Assume without loss of generality that $x_u(r-1) < x_v(r-1)$ (the opposite case is symmetric). Fix $V' = V \backslash \{u, v\}$. We partition the nodes in V' into three sets depending on the value of their workload after $r-1$ rounds in comparison to $x_u(r-1)$ and $x_v(r-1)$:

$$\begin{aligned} V_{low} &= \{w \in V' \mid x_w(r-1) < x_u(r-1)\} \\ V_{mid} &= \{w \in V' \mid x_u(r-1) \leq x_w(r-1) \leq x_v(r-1)\} \\ V_{high} &= \{w \in V' \mid x_w(r-1) > x_v(r-1)\} \end{aligned}$$

For each $w \in V_{low}$, we know that $d_{w,u}$ and $d_{w,v}$ potentially change from round $r-1$ to round r, and that these are the only gaps involving w that change in this round (as only u and v change their workload). In more detail, we know: $d_{w,u}(r) = d_{w,u}(r-1) + d_{u,v}(r-1)/2$, as u increases by $d_{u,v}(r-1)/2$ during round r. At the same time, we know $d_{w,v}(r) = d_{w,v}(r-1) - d_{u,v}(r-1)/2$, as v decreases by $d_{u,v}(r-1)/2$ during round r. Hence

$$\begin{aligned} d_{w,u}(r) + d_{w,v}(r) &\equiv d_{w,u}(r-1) + d_{u,v}(r-1)/2 \\ &\quad + d_{w,v}(r-1) - d_{u,v}(r-1)/2 \\ &= d_{w,u}(r-1) + d_{w,v}(r-1), \end{aligned}$$

from which it follows that the contribution of $d_{w,*}$ variables to $\phi(r)$ is the same as their contribution to $\phi(r-1)$. A symmetric argument can be applied to prove the same holds for every value $w' \in V_{high}$; that is, v's workload moves away from the workload of each such w' by the same amount that u's workload moves toward it, balancing out their impact on $\phi(r)$.

We are left to consider nodes in V_{mid}. For each $w \in V_{mid}$ there are two relevant cases. The first case is that w's round

$r-1$ workload is closer to u than v. Formally: $x_w(r-1) \leq x_u(r-1) + d_{u,v}(r-1)/2$.

In this case, we know that $d_{w,v}(r) = d_{w,v}(r-1) - d_{u,v}(r-1)/2$, as v moves the full $d_{u,v}(r-1)/2$ size gap toward w. The most we can say about $d_{w,u}(r)$, however, is that $d_{w,u}(r) \leq d_{w,u}(r-1) + d_{u,v}(r-1)/2$. The actual amount $d_{w,u}$ changes depends on $x_w(r-1)$: if $x_w(r-1)$ is close to $x_u(r-1)$ then $d_{w,u}$ can grow by close to the full $d_{u,v}(r-1)/2$ amount, whereas if $x_w(r-1)$ is in the middle of this interval $d_{w,u}$ does not change, and if $x_w(r-1)$ is closer to $x_u(r-1) + d_{u,v}(r-1)/2$ then $d_{w,u}$ can actually shrink. The claim that $d_{w,u}(r) \leq d_{w,u}(r-1) + d_{u,v}(r-1)/2$, therefore, is potentially a crude upper bound—but sufficient for our purposes. In particular, we now have that

$$\begin{aligned} d_{w,v}(r) + d_{w,u}(r) &\leq d_{w,v}(r-1) - d_{u,v}(r-1)/2 \\ &\quad + d_{w,u}(r-1) + d_{u,v}(r-1)/2 \\ &= d_{w,v}(r-1) + d_{w,u}(r-1), \end{aligned}$$

from which it follows that the contribution of $d_{w,*}$ variables to $\phi(r)$ is no more than their contribution to $\phi(r-1)$, and is potentially less. A symmetric argument applies to the second case where we consider $w' \in V_{mid}$ with a round $r-1$ workload closer to v than u; i.e., $x_w(r-1) > x_u(r-1) + d_{u,v}(r-1)/2$.

To pull together the pieces, fix $Q = \{\{x, y\} \mid x, y \in V\}$ and let $Q' = Q \setminus \{\{u, v\}\}$. The following inequality is a direct implication of our above results: $\sum_{\{x,y\} \in Q'} d_{x,y}(r) \leq \sum_{\{x,y\} \in Q'} d_{x,y}(r-1)$. We also know by assumption that $d_{u,v}(r) = 0$. It follows that $\phi(r) = \sum_{\{x,y\} \in Q'} d_{x,y}(r)$. Therefore

$$\begin{aligned} \phi(r) &= \sum_{\{x,y\} \in Q'} d_{x,y}(r) \leq \sum_{\{x,y\} \in Q'} d_{x,y}(r-1) \\ &= \phi(r-1) - d_{u,v}(r-1), \end{aligned}$$

which directly establishes the lemma statement. □

We will now apply Lemma V.3 iteratively to generalize our bound on ϕ's reduction to handle any feasible set of averaging nodes in a given round. Recall from the preliminaries above that D_r is the sum of gaps averaged in round r.

Lemma V.4. *Fix some execution of the max neighbor load balancing algorithm and some round r of this execution. It follows: $\phi(r-1) - \phi(r) \geq D_r$.*

The below corollary, which will prove useful in our later analysis, follows directly from Lemma V.4 and the fact that by definition D_r is always non-negative:

Corollary V.5. *The potential function ϕ is monotonic nonincreasing.*

Step #2: Lower Bound D_r. In the previous step we proved the deterministic property that $\phi(r)$ decreases by at least D_r from $\phi(r-1)$. In this step, we prove a probabilistic lower bound on D_r. In particular, we prove that with constant probability, $D_r \in \Omega(t_{max}(r-1)/\log n)$. In other words, the decrease in the potential is proportional to the largest workload gap, even if that gap is between nodes that are far away from each other in the graph and thus not able to connect to each other. We will start by fixing some important notation concerning a path in G_r with a large sum of pairwise gaps. We will then study the behavior of nodes near this path during round r.

Setup. Fix some $r-1$ round execution of the max neighbor algorithm. In this proof step, we will argue about the behavior of the algorithm during round r. To do so, we begin by noting that by the definition of t_{max} there must exist some u_{max} and v_{max} such that $d_{u_{max},v_{max}}(r-1) = t_{max}(r-1)$. Fix P to be a shortest path in G_r between u_{max} and v_{max}, where we define "shortest" with respect to hops. We know such a path must exist as we assume G_r is connected. We label the nodes in this path as $P = u_1, u_2, ..., u_k$, where $u_1 = u_{max}$ and $u_k = v_{max}$, and call P the *maximum gap path.*

In the lemmas below for this step, for each $i \in [k-1]$, we define $e_i = \{u_i, u_{i+1}\}$; i.e., the i^{th} edge in P. In a slight abuse of notation, we sometimes use $d_e(r-1)$, for an edge $e = \{u_i, u_{i+1}\}$ from P, to describe $d_{u_i,u_{i+1}}(r-1)$. Finally, for any node u_i in P, we define $\hat{u}_i$ to be the neighbor of u_i that has the $r-1$ workload most different from u_i. Formally, $\hat{u}_i = \text{argmax}_{v \in N_r(u_i)}\{d_{u_i,v}(r-1)\}$, where we break ties in the same way that u_i would in the max neighbor algorithm.

We begin with a useful observation that follows from the assumption that $|x_{u_k}(r-1) - x_{u_1}(r-1)| = t_{max}(r-1)$:

Lemma V.6. $\sum_{i=1}^{k-1} d_{u_i,u_{i+1}}(r-1) \geq t_{max}(r-1)$.

We next analyze the neighborhood surrounding each edge (equiv., step or hop) in P. Our goal is to show that for each such edge $\{u_i, u_{i+1}\}$, there is a reasonable probability that a nearby pair of neighbors $\{x, y\}$ with $d_{x,y}(r-1) \geq d_{u_i,u_{i+1}}(r-1)$ connects during round r. We will then build on this argument to show that many of the steps in P have nearby pairs with similar sized gaps (or larger) connect. Combined with Lemma V.6, it will follow that there is a good probability of D_r being lower bounded by something close to $t_{max}(r-1)$. We first formalize the type of behavior we are seeking:

Definition V.7. *We say an edge $\{u_i, u_{i+1}\}$ from P is* productive *if there exists an edge $\{v, w\}$ in G_r that satisfies the following properties:*

1) *v and w are both within three hops of u_i and u_{i+1} in G_r;*
2) *v and w connect during round r; and*
3) *$d_{v,w}(r-1) \geq d_{u_i,u_{i+1}}(r-1)$.*

We now establish a particular probabilistic event which guarantees productivity (recall that $\hat{u}_i$ is the neighbor of u_i in G_r with the largest workload gap with u_i):

Lemma V.8. *For any edge $\{u_i, u_{i+1}\}$ in P, if u_i decides to send proposals in round r and $\hat{u}_i$ decides to receive proposals in round r, then $\{u_i, u_{i+1}\}$ will be a productive edge regardless of the outcomes of other nodes' coin flips during this round.*

Proof. Assume that u_i decides to send and $\hat{u}_i$ decides to receive proposals in r. We now perform an exhaustive case analysis on the fate of u_i's proposal.

1) The first case is that $\hat{u}_i = u_{i+1}$ and u_{i+1} accepts u_i's proposal. In this case, the definition of a productive edge is clearly met for $v = u_i$ and $w = u_{i+1}$.
2) The second case is that $\hat{u}_i = u_{i+1}$ but u_{i+1} accepts a proposal from a different neighbor $u' \neq u_i$. By the definition of the algorithm, it must follow that $d_{u',u_{i+1}}(r-1) \geq d_{u_i,u_{i+1}}(r-1)$ (as u_{i+1} will accept the proposal that contains the largest workload gap). Because u' neighbors u_{i+1} which neighbors u_i, it is also clear that u' is within 2 hops of both u_i and u_{i+1}. In this case, the definition of a productive edge is met for $v = u'$ and $w = u_{i+1}$.
3) The third case is that $\hat{u}_i \neq u_{i+1}$ and $\hat{u}_i$ accepts u_i's proposal. Similar to the above case, we know here that $d_{u_i,\hat{u}_i}(r-1) \geq d_{u_i,u_{i+1}}(r-1)$ and $\hat{u}_i$ is within 2 hops of both u_i and u_{i+1}. In this case, the definition of a productive edge is met for $v = u_i$ and $w = \hat{u}_i$.
4) The fourth case is that $\hat{u}_i \neq u_{i+1}$ and $\hat{u}_i$ accepts a proposal from a different neighbor $u'' \neq u_i$. In this case, by definition, $d_{\hat{u}_i,u''}(r-1) \geq d_{\hat{u}_i,u_i}(r-1) \geq d_{u_i,u_{i+1}}(r-1)$. We also note that u'' is within 3 hops of u_i and u_{i+1}. In this case, the definition of a productive edge is met for $v = u''$ and $w = \hat{u}_i$.

In all possible cases $\{u_i, u_{i+1}\}$ ends up productive. □

We now leverage Lemma V.8 and Definition V.7 to bound the probability that enough edges are productive, and they represent enough independent workload averages, to ensure $D_r \in \Omega(t_{max}(r-1)/\log n)$. The below lemma will leverage the shortest path property of P to ensure that the set of productive edges we identify are sufficiently isolated to behave sufficiently independently.

Lemma V.9. *With probability at least* $1/8$*:* $D_r \geq t_{max}(r-1)/(60\ln(2n))$.

Proof. We begin by selecting a subset S of the edges in P for further consideration in this proof. We do so by applying the following iterative greedy selection procedure:

1) Initialize $\hat{P}$ to contain every edge in P (i.e., $\{u_1,u_2\},\{u_2,u_3\},...,\{u_{k-1},u_k\}$) and initialize $S = \emptyset$.
2) Fix $\{v,w\} = \text{argmax}_{\{v,w\}\in\hat{P}}\{d_{v,w}(r-1)\}$ (i.e., the edge in $\hat{P}$ with the largest gap; breaking ties arbitrarily).
3) Add $\{v,w\}$ to S and remove from $\hat{P}$ the edge $\{v,w\}$ as well as any other edge containing a node within 6 hops of v or w in G_r.
4) While $\hat{P}$ remains non-empty, loop back to step (2).

For the remainder of this proof, let S be the set of edges defined after the above iterative procedure terminates. We continue by proving several key properties regarding S then leverage these properties to establish the lemma statement.

Property #1: For each $e_i \in P$, there are at most 14 other edges in P within 6 hops of e_i in G_r.

Property #2: $\sum_{e\in S} d_e(r-1) \geq t_{max}(r-1)/15$.

Property #3: For each $e, e' \in S$, $e \neq e'$: the three hop neighborhoods of the endpoints in e share no nodes in common with the three hop neighborhoods of the endpoints in e'.

These three properties make it straightforward to prove the lemma, since they imply that the edges of S contain significant workload gaps, and that if two edges of S are both productive they each contribute separately to this sum of gaps (since they are far enough apart in G_r). Since each edge has a constant probability of being productive (and these are independent), this establishes the lemma. Details can be found in [1]. □

Step #3: Bounding the Number of Reductions Needed to Achieve τ-Convergence. We are now ready to prove Theorem V.2. The interesting argument is bounding the number of rounds until the system is τ-converged for a given convergence factor τ. We will leverage the key results from Steps #1 and #2 to bound this value. In particular, in Step #1 we established that $\phi(r)$ decreases by at least D_r in each round r, and in Step #2 we showed that there is a constant probability that $D_r \in \Omega(t_{max}(r-1)/\log n)$. These pieces are sufficient to obtain the claimed converge time complexity bound.

Proof (of Theorem V.2). We first note that if we arrive at a round r in which $\phi(r) \leq \tau$, then the system ends this round τ-converged: the sum of the gaps is at most τ, and thus clearly any individual gap is at most τ. Since ϕ is monotonic nonincreasing (by Corollary V.5), it follows that every round $r' \geq r$ is also τ-converged. So we just need to show that with high probability, ϕ will decrease to τ in the time bound stated by the theorem statement.

For each $r \geq 1$, call r "good" if and only if $\phi(r-1)-\phi(r) \geq t_{max}(r-1)/(60\ln(2n))$. We next calculate how many good rounds guarantee that ϕ falls below τ. To do so, we first note that by Corollary V.5, non-good rounds cannot increase ϕ, so we are safe to focus only on reductions generated by good rounds in calculating our bound.

By the definition of ϕ, for each $r \geq 0$ we know that $\phi(r) < t_{max}(r)n^2$. It follows that if r is a good round, then it decreases $\phi(r-1)$ by a multiplicative factor less than $(1-\frac{1}{60n^2\ln(2n)})$. Finally, we also observe that $t_{max}(0) \leq T$ and therefore $\phi(0) < Tn^2$. Leveraging these observations, to find the number of good rounds needed to decrease ϕ below τ we just need to find the minimum t such that

$$Tn^2\left(1-\frac{1}{60n^2\ln(2n)}\right)^t \leq \tau.$$

A simple calculation implies that $t_{con} = 60n^2\ln(2n)\ln(Tn^2\tau^{-1})$ is sufficient to satisfy this inequality.

We have now established that after t_{con} good rounds the system will be τ-converged for all future rounds. We are left to bound the number of rounds required to generate t_{con} good rounds with high probability.

For each round r, let X_r be the random indicator variable that evaluates to 1 if round r is good and otherwise evaluates to 0. By Lemmas V.4 and V.9, we know a given round r is good with probability at least $1/8$, regardless of the history of the execution through round $r-1$. We cannot, however, directly leverage this observation to calculate (and concentrate) the expected sum of X variables for a given execution length, as the distribution determining a given X_r might depend in part on the outcome of previous experiments. To overcome this issue, we define for each round r, a trivial random indicator variable $\hat{X}_r$ that evaluates to 1 with independent probability $1/8$ and otherwise evaluates to 0. Notice that for each r, X_r stochastically dominates $\hat{X}_r$, and therefore for each $t > 0$, $Y_t = \sum_{r=1}^{t} X_r$ stochastically dominates $\hat{Y}_t = \sum_{r=1}^{t} \hat{X}_r$. It follows for any $t > 0$, if $\hat{Y}_t \geq t_{con}$ with some probability p then $Y_t \geq t_{con}$ with probability at least p.

A Chernoff bound applied to $\hat{Y}_t$, for $t = c \cdot t_{con}$ (where $c \geq 1$ is a sufficiently large constant defined with respect to the constants in t_{con} and the constants in the Chernoff form used), provides that $\hat{Y}_t \geq t_{con}$ with high probability, and therefore so is Y_t. To conclude the proof, we note that $c \cdot t_{con} \in O(n^2 \log(\frac{Tn}{\tau}) \log n)$, as required by the theorem statement. □

B. Convergence for Small T

We now prove an alternative form of the convergence bound optimized for smaller values of T (as compared to n). Proof of the following leverages lemmas and techniques from Theorems V.1 and V.2.

Theorem V.10. *Fix some total workload $T > 0$, network size $n \geq 1$, and convergence factor τ, $0 < \tau < T$. Given these parameters, the max neighbor algorithm solves the load balancing problem for convergence factor τ in $O\left(\frac{Tn \log n}{\tau}\right)$ rounds.*

VI. Lower Bound

We now show two things: that our analysis is essentially tight, and that no algorithm from a natural family can do any better than max neighbor. In particular, we consider the natural convergence factor of $\tau = \Theta(T/n)$, meaning that all nodes end with workloads that are within a constant factor of the average. For this natural target, the bounds provided by Theorems V.2 and V.10 solve to $\tilde{O}(n^2)$ rounds.

We first show that this analysis is nearly tight: there are instances where our algorithm takes $\Omega(n^2)$ rounds to achieve $\Theta(T/n)$-convergence. We then build-up a more general lower bound: for any algorithm in which each node connects with at most one neighbor in each round, there exists *some* instance requiring $\Omega(n^2)$ rounds to achieve $\Theta(T/n)$-convergence. All missing proofs can be found in the full version [1].

A. Tightness of Analysis

To show that our analysis is tight, we give a specific instance (or really a family of instances parameterized by n) and show that $\Theta(T/n)$-convergence takes $\Omega(n^2)$ rounds. Our instance is actually extremely simple: the static line. In particular, consider the graph G with vertices $[n]$ and edges $\{\{i, i+1\} : i \in [n-1]\}$. We initially place the entire workload, normalized to 1, on the right endpoint and 0 everywhere else, i.e. we initialize $x_n(0) = 1$ and $x_i = 0$ for all $i \in [n-1]$ (normalizing $T = 1$). We will prove that it takes $\Omega(n^2)$ rounds before node 1 has value at least $\Omega(1/n)$. Note that our upper bounds hold even for very general dynamic graphs, while this lower bound is for the simple static line.

Our proof has two main steps: arguing that a related algorithm we call AVERAGE takes $\Omega(n^2)$ time, and then arguing that our algorithm takes at least as much time as AVERAGE.

The AVERAGE algorithm. The AVERAGE algorithm is very simple: every node connects with its neighbors, and the workload across every edge is equalized.

Definition VI.1. *Let $\{x_i(r-1)\}_{i\in[n]}$ be the weight distribution after round $r-1$. Then in round r of the AVERAGE algorithm, each vertex i with $i \geq 2$ sends $\frac{x_i(r-1)-x_{i-1}(r-1)}{2}$ work to node $i-1$ (if this amount of work is negative then work is transferred from $i-1$ to i).*

A simple calculation implies that we can think of AVERAGE as simply averaging the values of its neighbors (hence the name of the algorithm).

Lemma VI.2. *Let $\{x_i(r-1)\}_{i\in[n]}$ be a weight distribution at the end of round $r-1$, and let $\{x_i(r)\}_{i\in[n]}$ be the weight distribution obtained by applying the AVERAGE algorithm to $\{x_i(r-1)\}_{i\in[n]}$ in round r. Then $x_i(r) = \frac{x_{\max(i-1,0)}(r-1)+x_{\min(i+1,n)}(r-1)}{2}$ for all $i \in [n]$.*

We can now use this to prove that AVERAGE takes a long time to achieve $\Theta(T/n)$-convergence. This is essentially trivial after making one easy observation: the recurrence in Lemma VI.2 is *exactly* the same recurrence as the probability distribution of a random walk on the line (with loops at nodes 1 and n)! So we are essentially just trying to analyze the mixing time of a random walk on a line, which is well-known to be $\Theta(n^2)$ (see e.g. [20], [2]). A proof can be found in the full version [1].

Theorem VI.3. *When using the AVERAGE algorithm, for any constant $c < 1$, there is a constant k so that if $r < kn^2$ then $x_1(r) < c/n$.*

Relating AVERAGE to Max Neighbor: We now want to argue that Theorem VI.3 implies the same lower bound on the actual algorithm we use, max neighbor load balancing. The randomness in the max neighbor algorithm makes it tricky to analyze directly, so we instead introduce a generalization which will make the analysis simpler.

Definition VI.4. *Let $\{x_i(0)\}_{i\in[n]}$ be an initial weight distribution. In an* equal matching algorithm *there is a series of matchings $\{M_r\}_{r\geq 0}$ where M_r is a matching in the graph for*

each r, and in each round r every node balances its workload with its neighbor in matching M_r (if such a neighbor exists). Slightly more formally, for each edge $\{i, i+1\} \in M_r$ we set $x_i(r) = x_{i+1}(r) = \frac{1}{2}(x_i(r-1) + x_{i+1}(r-1))$.

The following lemma is direct from the definition of the max neighbor algorithm and an equal matching algorithm:

Lemma VI.5. *Max neighbor load balancing is an equal matching algorithm.*

We will now prove that AVERAGE converges faster than *any* equal matching algorithm, and thus converges faster than the max neighbor algorithm. In order to do this we will need a definition allowing us to compare the state of two algorithms by considering the cumulative sum of work across the line:

Definition VI.6. *Let $\{x_i\}_{i\in[n]}$ and $\{y_i\}_{i\in[n]}$ be two weight distributions. We say that x is* better *than y if $\sum_{i=1}^{k} x_i \geq \sum_{i=1}^{k} y_i$ for all $k \in [n]$.*

Lemma VI.7. *Let $\{x_i(r)\}_{i\in[n]}$ and $\{y_i(r)\}_{i\in[n]}$ be two weight distributions, let $\{x_i(r+1)\}_{i\in[n]}$ be the weight distribution resulting from applying AVERAGE to $\{x_i(r)\}_{i\in[n]}$, and let $\{y_i(r+1)\}_{i\in[n]}$ be a weight distribution resulting from applying an equal matching algorithm to $\{y_i(r)\}_{i\in[n]}$. If $\{x_i(r)\}_{i\in[n]}$ is better than $\{y_i(r)\}_{i\in[n]}$, then $\{x_i(r+1)\}_{i\in[n]}$ is better than $\{y_i(r+1)\}_{i\in[n]}$.*

We are now essentially finished, since Lemma VI.7 makes the following theorem easy to prove.

Theorem VI.8. *On the static line, max neighbor load balancing takes $\Omega(n^2)$ rounds to achieve convergence factor $\tau = \Theta(T/\log n)$.*

B. General Lower Bound

We now prove a more general lower bound: *no* randomized algorithm can achieve $\Theta(T/n)$-convergence in $o(n^2)$ rounds if in every round each node can connect with at most one other node. This would imply that our analysis of max neighbor Load Balancing is tight, but in order to prove the more general lower bound we will need more complicated instances, while Theorem VI.8 holds even on the static line. In fact, it is easy to see that the static line is not a general lower bound, but just a lower bound on max neighbor Load Balancing, as there are simple load balancing algorithms which do achieve $\Theta(T/n)$-convergence in only $O(n)$ time. But here we show that max neighbor is in fact the best possible algorithm in dynamic networks, subject to the requirement that every node exchanges work with at most one other node in every round.

Definition VI.9. *A load balancing algorithm is a* matching algorithm *if in every round r there is a matching M_r of G_r and work is exchanged in round r only between nodes connected by an edge of M_r.*

That is, a matching algorithm is a generalization of our previous notion of equal matching algorithm where we no longer require nodes that communicate to split work evenly. So, again, max neighbor load balancing is a matching algorithm.

We will assume an *online adaptive adversary*. At the beginning of every round r, the adversary knows the initial weight distribution $\{x_v(r-1)\}_{v\in V}$, and can choose the graph G_r using this knowledge. The algorithm then gets to run on G_r using random bits which are unknown to the adversary. It is easy to see that our upper bounds for max neighbor load balancing hold against such an adversary.

The rest of this section is devoted to proving the following:

Theorem VI.10. *Every matching algorithm requires $\Omega(n^2)$ rounds in order to achieve $\Theta(T/n)$-convergence against an online adaptive adversary.*

To prove this, fix some matching algorithm ALG. By the definition of the load-balancing problem, this means that when given a graph $G_r = (V, E_r)$ and a weight distribution $\{x_v(r)\}_{v\in V}$, the algorithm ALG will (possibly randomly) choose a matching $M_r \subseteq E_r$ and for every edge $\{u, v\}$ of M_r will distribute $x_u(r-1) + x_v(r-1)$ work between u and v in some way so that $x_u(r) + x_v(r) = x_u(r-1) + x_v(r-1)$.

Since our adversary is adaptive, we define our dynamic graph with respect to the choices of ALG. The dynamic graph will be a series of lines, where the ordering will depend on the previous choices made by the algorithm. Initially, let $V = [n]$, let G_1 be the line on $[n]$ (i.e. $E_1 = \{\{i, i+1\} : i \in [n-1]\}$), set $x_n(0) = 1$, and set $x_i(0) = 0$ for all $i < n$. Let $\phi_1 : [n] \to [n]$ be the identity function (we will think of ϕ_r as an ordering for the r'th round, so $\phi_r(i)$ will be the node at position i in G_r). We will maintain the invariant that G_r is the line defined by the ordering ϕ_r. (Note that this ϕ has nothing to do with the ϕ used in the upper bound as a potential function).

Informally, to get G_r from G_{r-1}, we look at the matching M_{r-1} and swap the nodes for each edge so that the smaller-weight node occurs first (but only for matching edges). More formally, let $\{u, v\} \in M_{r-1}$ with $u = \phi_{r-1}(i)$ and $v = \phi_{r-1}(i+1)$ (any edge must have this structure since G_{r-1} is the line defined by ordering ϕ_{r-1}). If $x_u(r-1) \leq x_v(r-1)$ then we set $\phi_r(i) = u$ and set $\phi_r(i+1) = v$, and otherwise we set $\phi_r(i) = v$ and set $\phi_r(i+1) = u$. For every node $w = \phi_{r-1}(j)$ that is not part of a matching edge, we will leave it in place by setting $\phi_r(j) = w$. This completely defines ϕ_r, and we let $G_r = (V, E_r)$ be defined by $E_r = \{\{\phi_r(i), \phi_r(i+1)\} : i \in [n-1]\}$.

We will proceed by trying to argue that the weight distributions we get from "equal" load balancing are better than the weight distributions we get from ALG. We first define the procedure EQUAL, which is very simple: in round r it uses the same matching M_r as ALG, but instead of balancing across edges in the same way as ALG, it will always evenly split the total workload of the two endpoints of each matching edge. We will also need to slightly redefine our definition of "better" in order to incorporate the dynamism of the instance.

Definition VI.11. *Let $\{x_i(r)\}_{i\in[n]}$ and $\{y_i(r)\}_{i\in[n]}$ be two weight distributions. We say that $\{x_i(r)\}_{i\in[n]}$ is* better *than $\{y_i(r)\}_{i\in[n]}$ if $\sum_{i=1}^{k} x_{\phi_{r+1}(i)}(r) \geq \sum_{i=1}^{k} x_{\phi_{r+1}(i)}(r)$, for all $i \in [k]$.*

In other words, if we have two weight distributions at the end of round r, then one distribution is better than the other if when using the order for round $r+1$ it is better in the same way as in Definition VI.6. We can now prove the main lemma.

Lemma VI.12. *Let $r \geq 1$ be a round. Let $\{x_i(r)\}_{i\in[n]}$ and $\{y_i(r)\}_{i\in[n]}$ be two weight distributions. Let $\{x_i(r+1)\}_{i\in[n]}$ be the weight distribution resulting from applying EQUAL to $\{x_i(r)\}_{i\in[n]}$ in graph G_{r+1}, and let $\{y_i(r+1)\}_{i\in[n]}$ be a weight distribution from applying ALG to $\{y_i(r)\}_{i\in[n]}$ in graph G_{r+1}. If $\{x_i(r)\}_{i\in[n]}$ is better than $\{y_i(r)\}_{i\in[n]}$, then $\{x_i(r+1)\}_{i\in[n]}$ is better than $\{y_i(r+1)\}_{i\in[n]}$.*

We can now prove Theorem VI.10. First, applying Lemma VI.12 inductively and with $k = 1$ implies that EQUAL always puts at least as much weight on the node in location 1 as ALG does. To prove that ALG takes $\Omega(n^2)$ rounds to achieve $\Theta(T/n)$-convergence, we just need to show that it will be $\Omega(n^2)$ rounds before EQUAL puts at least $\Omega(T/n) = \Omega(1/n)$ weight on location 1.

To see this, note that since EQUAL splits weight evenly between connected nodes, the vertex swaps in the graph (which are determined by ALG) do not matter—whether two nodes swap or not, the amount of weight on a location is the same if we are running EQUAL. Let $G = (V, E) = (V, E_1)$ be the static line on $[n]$. For each round r let $M'_r = \{\{i, i+1\} : \{\phi_r(i), \phi_r(i+1)\} \in M_r\}$ be the matching on the static line that consists of each edge $\{i, i+1\}$ where the nodes which were at those locations in G_r were connected in M_r. If we run EQUAL on the static line using matching M'_r in round r, we get the *exact* same weight distribution with respect to locations as if we run EQUAL in our dynamic graph.

We just need to show that running EQUAL using the $\{M'_r\}$ matchings on the static line takes $\Omega(n^2)$ rounds to get weight $\Omega(1/n)$ on node 1. Since EQUAL with these matchings is an equal matching algorithm (as in Definition VI.4), Lemma VI.7 and Theorem VI.3 imply exactly this lower bound.

REFERENCES

[1] Load balancing with bounded convergence in dynamic networks, 2016. Available at https://www.dropbox.com/s/lwte5hjz2q2mxj0/dynamicLB.pdf?dl=0.

[2] David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at http://www.stat.berkeley.edu/∼aldous/RWG/book.html.

[3] Jacques Bahi, Raphaël Couturier, and Flavien Vernier. Broken edges and dimension exchange algorithms on hypercube topology. In *Proceedings. Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 140–145. IEEE, 2003.

[4] Jacques Bahi, Raphaël Couturier, and Flavien Vernier. Synchronous distributed load balancing on dynamic networks. *Journal of Parallel and Distributed Computing*, 65(11):1397–1405, 2005.

[5] Jacques M Bahi, Raphaël Couturier, and Flavien Vernier. Synchronous distributed load balancing on totally dynamic networks. In *IPDPS 2007*, pages 1–8. IEEE, 2007.

[6] Nimantha Thushan Baranasuriya, Seth Lewis Gilbert, Calvin Newport, and Jayanthi Rao. Aggregation in smartphone sensor networks. In *2014 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 101–110. IEEE, 2014.

[7] Randal W Beard and Vahram Stepanyan. Information consensus in distributed multiple vehicle coordinated control. In *42nd IEEE Conference on Decision and Control*, volume 2, pages 2029–2034. IEEE, 2003.

[8] Jacques E. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Practice and Experience*, 2(4):289–313, 1990.

[9] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In *Proceedings of the 2015 International Colloquium on Automata, Languages, and Programming*, pages 528–539, 2015.

[10] Alejandro Cornejo, Seth Gilbert, and Calvin Newport. Aggregation in dynamic networks. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 195–204. ACM, 2012.

[11] George Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of parallel and distributed computing*, 7(2):279–301, 1989.

[12] Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

[13] R. Elsasser, B. Monien, and S. Schamberger. Load balancing in dynamic networks. In *Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on*, pages 193–200, May 2004.

[14] Seyed H. Hosseini, Bruce Litow, M Malkawi, Joseph McPherson, and K Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *Journal of Parallel and Distributed Computing*, 10(2):160–166, 1990.

[15] Chris Hyser, Bret Mckee, Rob Gardner, and Brian J Watson. Autonomic virtual machine placement in the data center. *Hewlett Packard Laboratories, Tech. Rep. HPL-2007-189*, pages 2007–189, 2007.

[16] Ali Jadbabaie, Jie Lin, and A Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988–1001, 2003.

[17] Soummya Kar and José MF Moura. Distributed average consensus in sensor networks with random link failures and communication channel noise. In *Proceedings of the 2007 Asilomar Conference on Signals, Systems and Computers*, pages 676–680. IEEE, 2007.

[18] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 513–522. ACM, 2010.

[19] Fabian Kuhn and Rotem Oshman. Dynamic networks: models and algorithms. *ACM SIGACT News*, 42(1):82–96, 2011.

[20] L. Lovász. Random walks on graphs: A survey. In D. Miklós, V. T. Sós, and T. Szőnyi, editors, *Combinatorics, Paul Erdős is Eighty*, volume 2, pages 353–398. János Bolyai Mathematical Society, Budapest, 1996.

[21] Luc Moreau. Stability of multiagent systems with time-dependent communication links. *Automatic Control, IEEE Transactions on*, 50(2):169–182, 2005.

[22] Reza Olfati Murray and Saber Richard M. Consensus protocols for networks of dynamic agents. In *Proceedings of the 2003 American Controls Conference*, 2003.

[23] Alok Nandan, Shirshanka Das, Giovanni Pau, Mario Gerla, and MY Sanadidi. Co-operative downloading in vehicular ad-hoc wireless networks. In *Second Annual Conference on Wireless On-demand Network Systems and Services*, pages 32–41. IEEE, 2005.

[24] Abderrahmane Sider and Raphaël Couturier. Fast load balancing with the most to least loaded policy in dynamic networks. *The Journal of Supercomputing*, 49(3):291–317, 2009.

[25] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75(6):1226, 1995.

[26] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[27] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.

[28] C-Z Xu and Francis C. M. Lau. Optimal parameters for load balancing using the diffusion method in k-ary n-cube networks. *Information Processing Letters*, 47(4):181–187, 1993.

[29] Cheng-Zhong Xu and Francis C. M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. *Journal of Parallel and Distributed Computing*, 16(4):385–393, 1992.

[30] Cheng-Zhong Xu and Francis CM Lau. Optimal parameters for load balancing with the diffusion method in mesh networks. *Parallel Processing Letters*, 4(01n02):139–147, 1994.