Plurality Consensus in Arbitrary Graphs: Lessons Learned from Load Balancing

Petra Berenbrink[1], Tom Friedetzky[2], Peter Kling[*3], Frederik Mallmann-Trenn[4], and Chris Wastell[†5]

1 **Simon Fraser University, Burnaby, Canada**
2 **Durham University, Durham, U.K.**
3 **Simon Fraser University, Burnaby, Canada**
4 **Simon Fraser University, Burnaby, Canada; and École normale supérieure, Paris, France**
5 **Durham University, Durham, United Kingdom**

Abstract

We consider *plurality consensus* in networks of n nodes. Initially, each node has one of k opinions. The nodes execute a (randomized) distributed protocol to agree on the *plurality opinion* (the opinion initially supported by the most nodes). In certain types of networks the nodes can be quite cheap and simple, and hence one seeks protocols that are not only time efficient but also simple and space efficient. Typically, protocols depend heavily on the employed communication mechanism, which ranges from sequential (only one pair of nodes communicates at any time) to fully parallel (all nodes communicate with all their neighbors at once) and everything in-between.

We propose a framework to design protocols for a multitude of communication mechanisms. We introduce protocols that solve the plurality consensus problem and are, with probability $1 - o(1)$, both time and space efficient. Our protocols are based on an interesting relationship between plurality consensus and distributed load balancing. This relationship allows us to design protocols that generalize the state of the art for a large range of problem parameters.

1998 ACM Subject Classification C.2.2 Network Protocols

Keywords and phrases Plurality Consensus, Distributed Computing, Load Balancing

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.10

1 Introduction

The objective in the *plurality consensus* problem is to find the so-called *plurality opinion* (i.e., the opinion that is initially supported by the largest subset of nodes) in a network G where, initially, each of the n nodes has one of k opinions. Applications of this problem include distributed computing [20, 30, 31], social networks [29, 17, 28], as well as modeling of biological interactions [16, 15]. All these areas typically demand both very simple and space-efficient protocols. Communication models, however, can vary from anything between simple sequential communication with a single neighbor (often used in biological settings as a simple variant of asynchronous communication [5]) to fully parallel communication where all nodes communicate with all their neighbors simultaneously (like broadcasting models in distributed computing). This diversity turns out to be a major obstacle in algorithm

* Supported in part by the Pacific Institute for the Mathematical Sciences (PIMS).
† Supported in part by the Engineering and Physical Sciences Research Council (EPSRC).

design, since protocols (and their analyses) to a large degree depend upon the employed communication mechanism.

In this paper we present two simple plurality consensus protocols called Shuffle and Balance. Both protocols work in a very general discrete-time communication model. The communication partners are determined by a (possibly randomized) sequence $(\boldsymbol{M}_t)_{t\leq N}$ of *communication matrices*, where we assume[1] N to be some suitably large polynomial in n. That is, nodes u and v can communicate in round t if and only if $\boldsymbol{M}_t[u,v] = 1$. In that case, we call the edge $\{u, v\}$ *active*; see [6, 32] for related graph models. Our results allow for a wide class of communication patterns (which can even vary over time) as long as the communication matrices have certain "smoothing" properties (cf. Section 2). These smoothing properties are inspired by similar smoothing properties used by Thomas Sauerwald and He Sun [32] for load balancing in the dimension exchange model. In fact, load balancing is the source of inspiration for our protocols. Initially, each node creates a suitably chosen number of tokens labeled with its own opinion. Our Balance protocol then performs discrete load balancing on these tokens, allowing each node to get an estimate on the total number of tokens for each opinion. The Shuffle protocol keeps the number of tokens on every node fixed, but shuffles tokens between communication partners. By keeping track of how many tokens of their own opinion (label) were exchanged in total, nodes gain an estimate on the total (global) number of such tokens. Together with a simple broadcast routine, all nodes can determine the plurality opinion.

The running time of our protocols is the smallest time t for which all nodes have stabilized on the plurality opinion. That is, all nodes have determined the plurality opinion and will not change. This time depends on the network G, the communication pattern $(\boldsymbol{M}_t)_{t\leq N}$, and the initial bias towards the plurality opinion (cf. Section 2). For both protocols we show a strong correlation between their running time, the mixing time of certain random walks and the (related) *smoothing time*, both of which are used in the analysis of recent load balancing results [32]. To give some more concrete examples of our results, let $T \coloneqq \mathrm{O}\left(\log n/(1-\lambda_2)\right)$, where $1-\lambda_2$ is the spectral gap of G. If the bias is sufficiently high, then both our protocols Shuffle and Balance determine the plurality opinion in time

- $n \cdot T$ in the *sequential model* (only one pair of nodes communicates per time step);
- $d \cdot T$ in the *balancing circuit model* (communication partners are chosen according to d (deterministic) perfect matchings in a round-robin fashion); and
- T in the *diffusion model* (all nodes communicate with all their neighbors at once).

To the best of our knowledge, these match the best known bounds in the corresponding models. For an arbitrary bias (in particular, *arbitrarily small* bias), the protocols differ in their time and space requirements. More details of our results can be found in Section 1.2.

1.1 Related Work

There is a diverse body of literature that analyzes consensus problems under various models and assumptions. Results differ in the considered topology (e.g., arbitrary or complete), the restrictions on model parameters (e.g., the number of opinions or the *initial bias*[2]), the time model (synchronous or asynchronous), or the required knowledge (e.g., n, maximal degree,

[1] For simplicity and without loss of generality; our protocols run in polynomial time in all considered models.

[2] The bias is $\alpha \coloneqq (n_1 - n_2)/n$, n_1 and n_2 being the support of the most and second most common opinions.

Table 1 Summary of plurality consensus results.

	Arbitrary Graph	Number of Opinions	Required Bias α O-notation	Time O-notation	Model	Space O-notation
Shuffle	✓	arbitrary	arbitrary	$T \cdot t_{\text{mix}}$ $T \cdot \log(n)/(1-\lambda_2)$ (d-reg graph)	sync & async	see Theorem 2
Balance	✓	arbitrary	arbitrary	τ $\log(n)/(1-\lambda_2)$ (d-reg graph)	sync & async	$k \cdot \log(n)$
[26]	✓	arbitrary	arbitrary	$D + \frac{F_2}{n_1^2} \cdot \log(k)$	broadcast	–
[27]	✓	2	arbitrary	n^5	async	1
[21]	✓	2	arbitrary	$\log n/\delta(G, n_1/n)$	async	1
[19]	expander	2	$vol(1) - vol(2) \geq 4\lambda_2^2 \cdot \lvert E\rvert$	$\log(n)$	sync	1
[18]	random d-reg	2	$\sqrt{1/d + d/n}$	$\log(n)$	sync	1
[9]	✗	$\leq n$	$\sqrt{\min\left\{k, \sqrt[3]{\frac{n}{\log(n)}}\right\} \cdot \frac{\log(n)}{n}}$	$\min\left\{k, \sqrt[3]{\frac{n}{\log(n)}}\right\} \cdot \log n$	sync	$\log(k)$
[8]	✗	$O((\frac{n}{\log(n)})^{1/3})$	$\epsilon \cdot n_2/n$	$md(c) \cdot \log(n)$	sync	$\log(k)$
[23]	✗	$O(n^\epsilon)$	$\sqrt{\log n/n}$	$k + \log(n)$	sync	$\log(k)$
[13]	✗	$o(\sqrt{n/\log(n)})$	$\gg \sqrt{\log n/n}$	$\log(n) \cdot \log\log(n)$	sync	$\log(k)$
[2]	✗	2	arbitrary	$\frac{\log^2(n)}{s\alpha} + \log^2(n)$	async	$s = O(n)$ states
[3]	✗	2	$\gg \log(n)/\sqrt{n}$	$\log(n)$	async	1

Shuffle assumes rough bounds on t_{mix} and n. Bounds on α can reduce the space requirements of our protocols. [26] requires a spanning tree and a common set of quasi-random hash functions. Time in the async model use parallel time. All results, except for [21], hold w.p. $1 - o(1)$. [2] also gives an expected time of $o(\log(n)/(s\alpha) + \log(n) \cdot \log(s))$.

or spanning tree). To capture this diverse spectrum, we classify[3] results into *population protocols*, *sensor networks*, and *pull voting*. A condensed form of this discussion is given in Table 1. We will not discuss work whose focus is too far away from this paper's, e.g., consensus on some arbitrary opinion, leader election, robustness concerns, or Byzantine models.

Population Protocols. The first line of work considers *population protocols* for consensus and models interactions between large populations of very simple entities (like molecules). Entities are modeled as finite state machines with a small state space and communicate asynchronously. In each step, an edge is chosen uniformly at random and only the two connected nodes communicate. We refer to this communication model as the *sequential model*. See [5, 4] for detailed introductions. Dana Angluin, James Aspnes, and David Eisenstat [3] propose a 3-state population protocol for majority voting (i.e., $k = 2$) on the clique. If the initial bias α is $\omega(\log n/\sqrt{n})$, their protocol agrees (w.h.p.) on the majority opinion in $O(n \cdot \log n)$ steps. George B. Mertzios, Sotiris E. Nikoletseas, Christoforos Raptopoulos, and Paul G. Spirakis [27] suggest a 4-state protocol for *exact* majority voting, which always returns the majority opinion (independent of α) in time $O(n^6)$ in arbitrary graphs and in time $O(n^2 \cdot \log(n)/\alpha)$ in the clique. This is optimal in that no population protocol for exact majority can have fewer than four states [27]. Dan Alistarh, Rati Gelashvili, and Milan Vojnovic [2] gives a protocol for $k = 2$ in the clique that allows for a speed-memory trade-off. It solves exact majority and has expected *parallel running time*[4] $O(\frac{\log n}{s \cdot \alpha} + \log n \cdot \log s)$ and (w.h.p.) $O(\frac{\log^2 n}{s \cdot \alpha} + \log^2 n)$.

Here, s is the number of states and must be in the range $s = O(n)$ and $s = \Omega(\log n \cdot \log\log n)$.

[3] This classification is neither unique nor injective but merely an attempt to make the overview more accessible.

[4] The number of steps divided by n. A typical measure for population protocols, based on the intuition that each node communicates roughly once in n steps.

In contrast to these population results, our protocols consider the case of arbitrary $k \geq 2$. Also, with the notable exception of [27], the above results are restricted to the complete graph. These restrictions are not surprising, given that these protocols operate on a very constrained state space. Our protocols work in arbitrary, even dynamic graphs. BALANCE can be seen as a slightly simplified and generalized version of [2], and SHUFFLE uses a similar idea for a speed-memory trade-off.

Sensor Networks. Another line of work has a background in sensor networks. *Quantized interval consensus* draws its motivation from signal processing. Initially, nodes measure quantized values (signals) and then communicate through a network to agree on the quantized values that enclose the average. This can be used to solve majority consensus ($k = 2$). The communication model is typically sequential. Florence Bénézit, Patrick Thiran, and Martin Vetterli [10] propose a protocol that is equivalent to the 4-state population protocol of [27] and prove that with probability 1 it converges in finite time, but without bounds on that convergence time. A more recent result by Moez Draief and Milan Vojnovic [21] shows that this protocol (and thus [27]) needs $\mathrm{O}\big(\frac{\log n}{\delta(Q_S,\alpha)}\big)$ steps in expectation. Here, $\delta(Q_S, \alpha)$ depends on the bias α and on the spectrum of a set of matrixes Q_S related to the underlying graph. The authors give concrete bounds for several specific graphs (e.g., in the complete graph the consensus time is of order[5] $\mathrm{O}\left(\log n/\alpha\right)$). The only related result for $k > 2$ we are aware of is [11] which again proves only convergence in finite time.

Another consensus variant is *mode computation*. For example, Fabian Kuhn, Thomas Locher, and Stefan Schmid [26] consider a graph of diameter D where each node has one or several of k distinct elements. The authors use a protocol based on a complex hashing scheme to compute the *mode* (the most frequent element) w.h.p. in time $\mathrm{O}\big(D + F_2/n_1^2 \cdot \log k\big)$. Here, $F_2 = \sum_i n_i^2$ is the second frequency moment and n_i the frequency of the i-th most common element. $F_2/n_1^2 \in [1, k]$ can be seen as an alternative bias measure. Nodes communicate via synchronous broadcasts and need a precomputed spanning tree and hash functions. [26] can also be used for aggregate computation as done by David Kempe, Alin Dobra, and Johannes Gehrke [25] (where the authors provide an elegant protocol to compute sums or averages in complete graphs).

Overall, [21] and [26] are probably most closely related to our work, as they consider arbitrary graphs. However, we cover more general communication models, including dynamic graphs. Similar to [21], our results for $k = 2$ rely on spectral properties of the underlying graph (and are asymptotically the same for their concrete examples). However, our bounds are related to well-studied load balancing bounds and mixing times of random walks (which we believe are easier to get a handle on than their $\delta(Q_S, \alpha)$).

Gossip Protocols. The third major research line on plurality consensus has its roots in gossiping and rumor spreading. Here, communication is typically restricted to synchronous pull requests (nodes query other nodes' opinions and use simple rules to update their own). See [30] for a slightly dated but thorough survey. Colin Cooper, Robert Elsässer, and Tomasz Radzik [18] consider a voting process for $k = 2$ opinions on d-regular graphs. They pull two random neighbors and, if they have the same opinion, adopt it. For random d-regular graphs and $\alpha = \Omega\big(\sqrt{1/d + d/n}\big)$, all nodes agree (w.h.p.) in $\mathrm{O}\left(\log n\right)$ rounds on the plurality opinion. For an arbitrary d-regular graph G, they need $\alpha = \Omega\left(\lambda_2\right)$ (where $1 - \lambda_2$ is the spectral gap of G). In the follow up paper Colin Cooper, Robert Elsässer, Tomasz Radzik,

[5] We state their bound in terms of our $\alpha = (n_1 - n_2)/n$; their definition of α differs slightly.

Nicolas Rivera, and Takeharu Shiraga [19] extend these results to expanders: The run time is $\mathrm{O}(\log n)$ for a bias of $\mathrm{vol}(1) - \mathrm{vol}(2) \geq 4\lambda_2^2 \cdot |E|$, where $\mathrm{vol}(1)$ and $\mathrm{vol}(2)$ denote the sum of degrees over nodes having Opinion 1 and 2, respectively. Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, Riccardo Silvestri, and Luca Trevisan [9] consider a similar update rule on the clique for k opinions. Here, each node pulls the opinion of three random neighbors and adopts the majority among those. They need $\mathrm{O}(\log k)$ memory bits and prove (w.h.p.) a tight running time of $\Theta(k \cdot \log n)$ (given a sufficiently high bias α). Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Riccardo Silvestri [8] build upon the idea of the 3-state population protocol from [3] (but in the gossip model) and generalize it to k opinions. Nodes pull the opinion of a random neighbor in each round. If $n_1 \geq (1+\varepsilon) \cdot n_2$ for a constant $\varepsilon > 0$ and if $k = \mathrm{O}\left((n/\log n)^{1/3}\right)$, they agree (w.h.p.) on the plurality opinion in time $\mathrm{O}(\mathrm{md} \cdot \log n)$ on the clique and need $\log k + 1$ bits. The *monochromatic distance* $\mathrm{md} \in [1, k]$ is an alternative bias measure (based on an idea similar to the frequency moment in [26]). Petra Berenbrink, Tom Friedetzky, George Giakkoupis, and Peter Kling [13] build upon [3] and design a protocol that reaches plurality consensus (w.h.p.) in time $\mathrm{O}(\log n \cdot \log\log n)$ and uses $\log k + 4$ bits.

The running times of gossip protocols are relatively good when compared to other protocols, like population protocols or those introduced here (cf. Table 1). In particular, these results do typically not show a linear dependency on the bias, as our Shuffle protocol or [2, 21, 27] do. This efficiency however comes at the expense of parameter constraints. In particular, results like [8, 13] do not seem to easily extend to arbitrary graphs and have inherent constraints on both k and α. Comparing these results seems to indicate that, at least for arbitrary graphs, there is a jump in complexity depending on whether or not one allows the protocol to fail for small absolute bias values.

1.2 Our Contribution

We introduce two protocols for plurality consensus, called Shuffle and Balance. Both solve plurality consensus under a diverse set of (randomized or adversarial) communication patterns in arbitrary graphs for any positive bias. We continue with a detailed description of our results.

Shuffle. Our main result is the Shuffle protocol. In the first time step each node generates γ tokens labeled with its initial opinion. During round t, any pair of nodes connected by an active edge (as specified by the communication pattern $(\boldsymbol{M}_t)_{t \leq N}$) exchanges tokens. We show that Shuffle solves plurality consensus and allows for a trade-off between running time and memory. More exactly, let the number of tokens be $\gamma = \mathrm{O}\left(\log n/(\alpha^2 \cdot T)\right)$, where T is a parameter to control the trade-off between memory and running time[6]. Moreover, let t_{mix} be such that any time interval $[t, t + t_{\mathrm{mix}}]$ is ε-*smoothing*[7] (cf. Section 2). Given knowledge of the *maximum number of communication partners* Δ and the *mixing time* t_{mix} of the underlying communication pattern[8], Shuffle lets all nodes agree on the plurality opinion in $\mathrm{O}(T \cdot t_{\mathrm{mix}})$ rounds (w.h.p.), using $\mathrm{O}\left(\log n/(\alpha^2 T) \cdot \log k + \log(T \cdot t_{\mathrm{mix}})\right)$ memory bits per

[6] The protocol works for any integral choice of γ (this fixes the trade-off parameter T).

[7] Intuitively, this means that the communication pattern has good load balancing properties during any time window of length t_{mix}. This coincides with the worst-case mixing time of a lazy random walk on active edges.

[8] For static graphs, Δ is the maximal degree which can be easily computed in a distributed way, see for example [14]. For t_{mix}, good bounds are known for many static graphs [1, Chapter 5].

node. This implies, for example, that plurality consensus on expanders in the sequential model is achieved in $\mathrm{O}(T \cdot n \log n)$ time steps and with $\mathrm{O}(\log n \cdot \log k/T + \log(Tn))$ memory bits (assuming a constant initial bias). For arbitrary graphs, arbitrary bias, and many natural communication patterns (e.g., communicating with all neighbors in every round or communicating via random matchings), the time for plurality consensus is closely related to the spectral gap of the underlying communication network (cf. Corollary 3).

While our protocol is relatively simple, the analysis is quite involved. The idea is to observe that after t_{mix} time steps, each single token is on any node with (roughly) the same probability; the difficulty is that token movements are not independent. The main ingredients for our analysis are Lemmas 6 and 7, which generalize a result by Thomas Sauerwald and He Sun [32] (we believe that this generalization is interesting in its own right). These lemmas show that the joint distribution of token locations is negatively correlated, allowing us to derive a suitable Chernoff bound. Once this is proven, nodes can "count" tokens every t_{mix} time steps, building up over time an estimate of the total number of tokens labeled with their own opinion. By broadcasting these estimates, all nodes determine the plurality opinion.

Balance. The previous protocol, Shuffle, allows for a nice trade-off between running time and memory. If the number of opinions is relartively small, our much simpler Balance protocol gives better results. In Balance, each node u maintains a k-dimensional load vector. If j denotes u's initial opinion, the j-th dimension of this load vector is initialized with $\gamma \in \mathbb{N}$ (a sufficiently large value) and any other dimension is initialized with zero. In each time step, all nodes perform a simple, discrete load balancing on each dimension of these load vectors. Our results imply, for example, that plurality consensus on expanders in the sequential model is achieved in only $\mathrm{O}(n \cdot \log n)$ time steps with $\mathrm{O}(k)$ memory bits per node (assuming a constant initial bias).

Balance can be thought of as a (slightly simplified) version of [2] or [25] that generalizes naturally to $k \geq 2$ and arbitrary (even dynamic) graphs. In the setting of [2] (but as opposed to [2] for arbitrary k), it achieves plurality consensus with probability $1 - \mathrm{o}(1)$ in parallel time $\mathrm{O}(\log n)$ and uses $\mathrm{O}(k \cdot \log(1/\alpha)) = \mathrm{O}(k \cdot \log n)$ bits per node (Corollary 13), an improvement by a $\log(n)$ factor.

2 Model & General Definitions

We consider an undirected graph $G = (V, E)$ of $n \in \mathbb{N}$ nodes and let $1 - \lambda_2$ denote the eigenvalue (or spectral) gap of G. Each node u is assigned an *opinion* $o_u \in \{1, 2, \ldots, k\}$. For $i \in \{1, 2, \ldots, k\}$, we use $n_i \in \mathbb{N}$ to denote the number of nodes which have initially opinion i. Without loss of generality (w.l.o.g), we assume $n_1 > n_2 \geq \cdots \geq n_k$, such that 1 is the opinion that is initially supported by the largest subset of nodes. We also say that 1 is the *plurality opinion*. The value $\alpha := \frac{n_1 - n_2}{n} \in [1/n, 1]$ denotes the *initial bias* towards the plurality opinion. In the *plurality consensus problem*, the goal is to design simple, distributed protocols that let all nodes agree on the plurality opinion. Time is measured in discrete rounds, such that the (randomized) running time of our protocols is the number of rounds it takes until all nodes are aware of the plurality opinion. As a second quality measure, we consider the total number of memory bits per node that are required by our protocols. All our statements and proofs assume n to be sufficiently large.

Communication Model. In any given round, two nodes u and v can communicate if and only if the edge between u and v is *active*. We use $\boldsymbol{M}_t$ to denote the symmetric

communication matrix at time t, where $\boldsymbol{M}_t[u,v] = \boldsymbol{M}_t[v,u] = 1$ if $\{u,v\}$ is active and $\boldsymbol{M}_t[u,v] = \boldsymbol{M}_t[v,u] = 0$ otherwise. We assume (w.l.o.g) $\boldsymbol{M}_t[u,u] = 1$ (allowing nodes to "communicate" with themselves). Typically, the sequence $\boldsymbol{M} = (\boldsymbol{M}_t)_{t\in\mathbb{N}}$ of communication matrices (the *communication pattern*) is either randomized or adversarial, and our statements merely require that $\boldsymbol{M}$ satisfies certain smoothing properties (see below). For the ease of presentation, we restrict ourselves to polynomial number of time steps and consider only communication patterns $\boldsymbol{M} = (\boldsymbol{M}_t)_{t\le N}$ where $N = N(n)$ is an arbitrarily large polynomial. Let us briefly mention some natural and common communication models covered by such patterns:

- *Diffusion Model:* All edges of the graph are permanently activated.
- *Random matching model:* In every round t, the active edges are given by a random matching. We require that random matchings from different rounds are mutually independent[9]. Results for the random matching model dependent on $p_{\min} := \min_{t\in\mathbb{N},\{u,v\}\in E} \Pr(\boldsymbol{M}_t[u,v] = 1)$.
- *Balancing Circuit Model:* There are d perfect matchings $\boldsymbol{M}_0, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_{d-1}$ given. They are used in a round-robin fashion, such that for $t \ge d$ we have $\boldsymbol{M}_t = \boldsymbol{M}_{t \bmod d}$.
- *Sequential Model:* In each round t an uniformly random edge $\{u,v\} \in E$ is activated.

Notation. We use $\|\boldsymbol{x}\|_\ell$ to denote the ℓ-norm of vector $\boldsymbol{x}$, where the ∞-norm is the vector's maximum absolute entry. In general, bold font indicates vectors and matrices, and $x(i)$ refers to the i-th component of $\boldsymbol{x}$. The *discrepancy* of $\boldsymbol{x}$ is defined as $\operatorname{disc}(\boldsymbol{x}) := \max_i x(i) - \min_i x(i)$. For $i \in \mathbb{N}$ we define $[i] := \{1,2,\ldots,i\}$ as the set of the first i integers. We use $\log x$ to denote the binary logarithm of $x \in \mathbb{R}_{>0}$. We write $a \mid b$ if a divides b. For any node $u \in V$, we use $d(u)$ to denote u's degree in G and $d_t(u) := \sum_v \boldsymbol{M}_t[u,v]$ to denote its *active degree* at time t (i.e., its degree when restricted to active edges). Similarly, $N(u)$ and $N_t(u)$ refer to u's (active) neighborhood. Moreover, $\Delta := \max_{t,u} d_t(u)$ is the maximum active degree of any node. We assume knowledge of Δ. On static graphs it can be computed efficiently in a distributed manner [14] and it is given by many dynamic graph models (e.g., 1 for the sequential model, d for balancing circuits). We say an event happens with high probability (w.h.p.) if its probability is at least $1 - 1/n^c$ for $c \in \mathbb{N}$.

Smoothing Property. The running time of our protocols is closely related to the runnig time ("smoothing time") of diffusion load balancing algorithms, which in turn is a function of the mixing time of a random walk on G (see also [6, 32]). More exactly, we consider a random walk on G that is restricted to the active edges in each time step. As indicated in Section 1.2, this random walk should converge towards the uniform distribution over the nodes of G. This leads to the following definition of the random walk's transition matrices $\boldsymbol{P}_t$ based on the communication matrices $\boldsymbol{M}_t$:

$$\boldsymbol{P}_t[u,v] := \begin{cases} \frac{1}{2\Delta} & \text{if } \boldsymbol{M}_t[u,v] = 1 \text{ and } u \neq v, \\ 1 - \frac{d_t(u)}{2\Delta} & \text{if } \boldsymbol{M}_t[u,v] = 1 \text{ and } u = v, \\ 0 & \text{if } \boldsymbol{M}_t[u,v] = 0. \end{cases} \tag{1}$$

Obviously, $\boldsymbol{P}_t$ is doubly stochastic for all $t \in \mathbb{N}$. Moreover, note that the random walk is trivial in any matching-based model, while we get $\boldsymbol{P}_t[u,v] = \frac{1}{2d}$ for every edge $\{u,v\} \in E$ in

[9] Note that there are several simple, distributed protocols to obtain such matchings [24, 14].

the diffusion model on a d-regular graph. We are now ready to define the required smoothing property.

▶ **Definition 1** (ε-smoothing)**.** Consider a fixed sequence $(\boldsymbol{M}_t)_{t\leq N}$ of communication matrices and a time interval $[t_1, t_2]$. We say $[t_1, t_2]$ is *ε-smoothing* (under $(\boldsymbol{M}_t)_{t\leq N}$) if for any non-negative vector $\boldsymbol{x}$ with $\|\boldsymbol{x}\|_\infty = 1$ it holds that $\mathrm{disc}(\boldsymbol{x} \cdot \prod_{t=t_1}^{t_2} \boldsymbol{P}_t) \leq \varepsilon$. Moreover, we define the *mixing time* $t_{\mathrm{mix}}(\varepsilon)$ as the smallest number of steps such that any time window of length $t_{\mathrm{mix}}(\varepsilon)$ is ε-smoothing. That is, $t_{\mathrm{mix}}(\varepsilon) := \min\{\, t' \mid \forall t \in \mathbb{N}\colon [t, t+t'] \text{ is } \varepsilon\text{-smoothing} \,\}$.

The mixing time can be seen as the worst-case time required by a random walk to get "close" to the uniform distribution. If the parameter ε is not explicitly stated, we consider $t_{\mathrm{mix}} := t_{\mathrm{mix}}(n^{-5})$. Note that Shuffle assumes knowledge of a bound on t_{mix} (cf. Section 1.2).

3 Protocol Shuffle

Our main result is the following theorem, stating the correctness as well as the time-/space-efficiency of Shuffle. The protocol is described in Section 3.1, followed by its analysis in Section 3.2.

▶ **Theorem 2.** *Let $\alpha = \frac{n_1 - n_2}{n} \in [1/n, 1]$ denotes the initial bias. Consider a fixed communication pattern $(\boldsymbol{M}_t)_{t\leq N}$ and an arbitrary parameter $T \in \mathbb{N}$. Protocol* Shuffle *ensures that all nodes know the plurality opinion after $\mathrm{O}(T \cdot t_{mix})$ rounds (w.h.p.) and requires $\left(12 \cdot \frac{\log(n)}{\alpha^2 \cdot T} + 4\right) \cdot \log(k) + 4\log\left(\frac{12 \cdot \log(n)}{\alpha^2}\right) + \log(T \cdot t_{mix})$ memory bits per node.*

The parameter T in the statement serves as a lever to trade running time for memory. Since t_{mix} depends on the graph and communication pattern, Theorem 2 might look a bit unwieldy. The following corollary gives a few concrete examples for common communication patterns on general graphs.

▶ **Corollary 3.** *Let G be an arbitrary d-regular graph.* Shuffle *ensures that all nodes agree on the plurality opinion (w.h.p.) using $\left(12 \cdot \frac{\log(n)}{\alpha^2 \cdot T} + 4\right) \cdot \log(k) + 4\log\left(\frac{12 \cdot \log(n)}{\alpha^2}\right) + \log(T \cdot t_{mix})$ bits of memory in time*

- $\mathrm{O}\left(T \cdot \frac{\log(n)}{1-\lambda_2}\right)$ *in the diffusion model,*
- $\mathrm{O}\left(\frac{T}{d \cdot p_{min}} \cdot \frac{\log(n)}{1-\lambda_2}\right)$ *in the random matching model,*
- $\mathrm{O}\left(T \cdot d \cdot \frac{\log(n)}{1-\lambda_2}\right)$ *in the balancing circuit model, and*
- $\mathrm{O}\left(T \cdot n \cdot \frac{\log(n)}{1-\lambda_2}\right)$ *in the sequential model.*

3.1 Protocol Description

We continue to explain the Shuffle protocol given in Listing 1. Our protocol consists of three parts that are executed in each time step: the *shuffle* part, the *broadcast* part, and the *update* part.

Every node u is initialized with $\gamma \in \mathbb{N}$ tokens labeled with u's opinion o_u. Our protocol sends 2Δ tokens chosen uniformly at random (without replacement) over each edge $\{\, u, v \,\} \in E$. Here, $\gamma \geq 2\Delta^2$ is a parameter depending on T and α to be fixed during the analysis[10]. Shuffle maintains the invariant that, at any time, all nodes have exactly γ tokens. In addition to storing the tokens, each node maintains a set of auxiliary variables. The variable

[10] Shuffle needs not to know α, it works for any choice of γ; such a choice merely fixes the trade-off parameter T.

```
1  for {u, v} ∈ E with M_t[u, v] = 1:                          {shuffle part}
2    send 2Δ tokens chosen u.a.r. (without replacement) to v
3
4  for {u, v} ∈ E with M_t[u, v] = 1:                          {broadcast
   part}
5    send     (dom_u, e_u)
6    receive  (dom_v, e_v)
7  v := w with  e_w ≥ e_w'   ∀w, w' ∈ N_t(u) ∪ {u}
8  (dom_u, e_u) := (dom_v, e_v)
9
10 if t ≡ 0 (mod t_mix):                                       {update part}
11   increase c_u by the number of tokens labeled o_u held by u
12   plu_u := dom_u              {plurality guess: last broadcast's dom. op.}
13   (dom_u, e_u) := (o_u, c_u)   {reset broadcast}
```

Listing 1 Protocol SHUFFLE as executed by node u at time t. At time zero, each node u creates γ tokens labeled o_u and sets $c_u := 0$ and $(dom_u, e_u) := (o_u, c_u)$.

c_u is increased during the update part and counts tokens labeled o_u. The variable pair (dom_u, e_u) is a temporary guess of the plurality opinion and its frequency. During the broadcast part, nodes broadcast these pairs, replacing their own pair whenever they observe a pair with higher frequency. Finally, the variable plu_u represents the opinion currently believed to be the plurality opinion. The shuffle and broadcast parts are executed in each time step, while the update part is executed only every t_{mix} time steps

Waiting t_{mix} time steps for each update gives the broadcast enough time to inform all nodes and ensures that the tokens of each opinion are well distributed. The latter implies that, if we consider a node u with opinion $o_u = i$ at time $T \cdot t_{\text{mix}}$, the value c_u is a good estimate of $T \cdot \gamma n_i/n$ (which is maximized for the plurality opinion). When we reset the broadcast (Line 13), the subsequent t_{mix} broadcast steps ensure that all nodes get to know the pair (o_u, c_u) for which c_u is maximal. Thus, if we can ensure that c_u is a good enough approximation of $T \cdot \gamma n_i/n$, all nodes get to know the plurality.

3.2 Analysis of Shuffle

Fix a communication pattern $(\boldsymbol{M}_t)_{t \le N}$ and an arbitrary parameter $T \in \mathbb{N}$. Remember that $t_{\text{mix}} = t_{\text{mix}}(n^{-5})$ denotes the smallest number such that any time window of length t_{mix} is n^{-5}-smoothing under $(\boldsymbol{M}_t)_{t \le N}$. We set the number of tokens stored in each node to $\gamma := \lceil c \cdot \frac{\log n}{\alpha^2 T} \rceil$, where c is a suitable constant. The analysis of SHUFFLE is largely based on Lemma 11, which states that, after $\mathrm{O}(T \cdot t_{\text{mix}})$ time steps, the counter values c_u can be used to reliably separate the plurality opinion from any other opinion. The main technical difficulty is the huge dependency between the tokens' movements, rendering standard Chernoff-bounds inapplicable. Instead, we show that certain random variables satisfy the negative regression condition (Lemma 6), which allows us to majorize the token distribution by a random walk (Lemma 7) and to derive the Chernoff type bound in Lemma 10. This Chernoff type bound can be used to show that all counter values are concentrated which is the main pillar of the proof of Theorem 2.

Majorizing Shuffle by Random Walks

While our SHUFFLE protocol assumes that 2Δ divides γ, here we assume the slightly weaker requirement that $\boldsymbol{P}_t[u, v] \cdot \gamma \in \mathbb{N}$ for any $u, v \in V$ and $t \in \mathbb{N}$. Let us first introduce some

notation for the shuffle part at time t of our protocol. To ease the discussion, we consider u as a neighbor of itself and speak of $d_t(u)+1$ neighbors. For $i \in [d_t(u)+1]$, let $N_t(u,i) \in V$ denote the i-th neighbor of u (in an arbitrary order). Fix a node u and let u's tokens be numbered from 1 to γ. Our assumption on γ allows us to partition the tokens into $d_t(u)+1$ disjoint subsets (*slots*) $S_i \subseteq [\gamma]$ of size $\boldsymbol{P}_t[u,v]\cdot\gamma$ each, where $v = N_t(u,i)$. Let $\pi_{t,u}\colon [\gamma] \to [\gamma]$ be a random permutation. Token j with $\pi_{t,u}(j) \in S_i$ is sent to u's i-th neighbor. To ease notation, we drop the time index t and write π_u instead of $\pi_{t,u}$ (and, similarly for $d(u)$ and $N(u,i)$).

A *configuration* c describes the location of *all* γn tokens at a given point in time. For a token $j \in [\gamma n]$ we use $u_j \in V$ to denote its location in configuration c (which will always be clear from the context). For each such token j we define a random variable $X_j \in [d(u_j)+1]$ with $X_j = i$ if and only if $\pi_{u_j}(j) \in S_i$. In other words, X_j indicates to which of u_j's neighbors token j is sent. Our key technical lemma (Lemma 6) establishes the *negative regression condition* for these $(X_j)_{j\in[\gamma n]}$ variables. Negative regression is defined as follows:

▶ **Definition 4** (Neg. Regression [22, Def. 21])**.** A vector $(X_1, X_2, \dots, X_n)$ of random variables is said to satisfy the *negative regression* condition if $\mathbb{E}\left[f(X_l, l\in\mathcal{L}) \mid X_r = x_r, r\in\mathcal{R}\right]$ is non-increasing in each x_r for any disjoint $\mathcal{L},\mathcal{R}\subseteq[n]$ and for any non-decreasing function f.

▶ **Lemma 5** ([22, Lemma 26])**.** *Let $(X_1, X_2, \dots, X_n)$ satisfy the negative regression condition and consider an arbitrary index set $I \subseteq [n]$ as well as any family of non-decreasing functions f_i $(i \in \{\, I \,\})$. Then, we have*

$$\mathbb{E}\left[\prod_{i\in I} f_i(X_i)\right] \le \prod_{i\in I} \mathbb{E}\left[f_i(X_i)\right] \tag{2}$$

▶ **Lemma 6** (NRC)**.** *Fix a configuration c and consider the random variables $(X_j)_{j\in[\gamma n]}$. Then $(X_j)_{j\in[\gamma n]}$ satisfies the negative regression condition (*NRC*).*

Proof. Remember that u_j is the location of token j in configuration c and that $X_j \in [d(u_j)+1]$ indicates where token j is sent in the next step. We show for any $u \in V$ that $(X_j)_{j\colon u_j=u}$ satisfies the NRC. The lemma's statement follows since the π_u are chosen independently (if two independent vectors (X_j) and (Y_j) satisfy the NRC, then so do both together).

Fix a node u and disjoint subsets $\mathcal{L},\mathcal{R} \subseteq \{\, j\in[\gamma n] \mid u_j = u \,\}$ of tokens on u. Define $d := d(u)$ and let $f\colon [d+1]^{|\mathcal{L}|} \to \mathbb{R}$ be an arbitrary non-decreasing function. We have to show that $\mathbb{E}\left[f(X_l, l\in\mathcal{L}) \mid X_r = x_r, r\in\mathcal{R}\right]$ is non-increasing in each x_r (cf. Definition 4). That is, we need

$$\mathbb{E}\left[f(X_l, l\in\mathcal{L}) \mid X_r = x_r, r\in\mathcal{R}\right] \le \mathbb{E}\left[f(X_l, l\in\mathcal{L}) \mid X_r = \tilde{x}_r, r\in\mathcal{R}\right], \tag{3}$$

where $x_r = \tilde{x}_r$ holds for all $r \in \mathcal{R}\setminus\{\,\hat{r}\,\}$ and $x_{\hat{r}} > \tilde{x}_{\hat{r}}$ for a fixed index $\hat{r}\in\mathcal{R}$. We prove Inequality (3) via a coupling of the processes on the left-hand side (LHS process) and right-hand side (RHS process) of that inequality. Since $x_{\hat{r}} \neq \tilde{x}_{\hat{r}}$, these processes involve two slightly different probability spaces Ω and $\tilde{\Omega}$, respectively. To couple these, we employ a common uniform random variable $U_i \in [0,1)$. By partitioning $[0,1)$ into $d+1$ suitable slots for each process (corresponding to the slots S_i mentioned above), we can use the outcome of U_i to set the X_j in both Ω and $\tilde{\Omega}$. We first explain how to handle the case $x_{\hat{r}} - \tilde{x}_{\hat{r}} = 1$. The case $x_{\hat{r}} - \tilde{x}_{\hat{r}} > 1$ follows from this by a simple reordering argument.

So assume $x_{\hat{r}} - \tilde{x}_{\hat{r}} = 1$. We reveal the yet unset random variables X_j (i.e., $j \in [\gamma n]\setminus\mathcal{R}$) one by one in order of increasing indices. To ease the description assume (w.l.o.g.) that the tokens from $\mathcal{R}$ are numbered from 1 to $|\mathcal{R}|$. When we reveal the j-th variable (which indicates

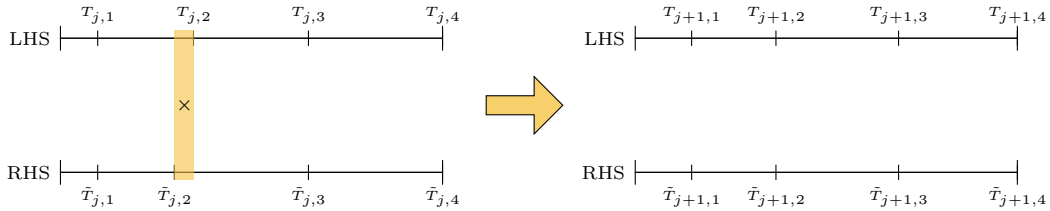

Figure 1 Illustration showing the $d+1=4$ different slots for the LHS and RHS process and how they change. In this example, $x_{\hat{r}}=3$ and $\tilde{x}_{\hat{r}}=2$. On the left, the uniform random variable U_j falls into slot $[T_1, T_2)$ for the LHS process (causing j to be sent to node $N(u,2)$) and into slot $[\tilde{T}_2, \tilde{T}_3)$ for the RHS process (causing j to be sent to node $N(u,3)$).

the new location of the j-th token), note that the probability $p_{j,i}$ that token j is assigned to $N(u,i)$ depends solely on the *number* of previous tokens $j'<j$ that were assigned to $N(u,i)$. Thus, we can consider $p_{j,i}\colon \mathbb{N}\to[0,1]$ as a function mapping $x\in\mathbb{N}$ to the probability that j is assigned to $N(u,i)$ conditioned on the event that exactly x previous tokens were assigned to $N(u,i)$. Note that $p_{j,i}$ is non-increasing. For a vector $\boldsymbol{x}\in\mathbb{N}^{d+1}$, we define a threshold function $T_{j,i}\colon \mathbb{N}^{d+1}\to[0,1]$ by $T_{j,i}(\boldsymbol{x}) \coloneqq \sum_{i'\le i} p_{j,i'}(x_{i'})$ for each $i\in[d+1]$. To define our coupling, let $\beta_{j,i} \coloneqq |\{\, j'<j \mid X_{j'}=i \,\}|$ denote the number of already revealed variables with value i in the LHS process and define, similarly, $\tilde{\beta}_{j,i} \coloneqq |\{\, j'<j \mid \tilde{X}_{j'}=i \,\}|$ for the RHS process. We use $\boldsymbol{\beta_j}, \boldsymbol{\tilde{\beta}_j}\in\mathbb{N}^{d+1}$ to denote the corresponding vectors. Now, to assign token j we consider a uniform random variable $U_j\in[0,1)$ and assign j in both processes using customized partitions of the unit interval. To this end, let $T_{j,i}\coloneqq T_{j,i}(\boldsymbol{\beta_j})$ and $\tilde{T}_{j,i}\coloneqq T_{j,i}(\boldsymbol{\tilde{\beta}_j})$ for each $i\in[d+1]$. We assign X_j in the LHS and RHS process as follows:

- **LHS Process:** $X_j=x_j=i$ if and only if $U_j\in[T_{j,i-1}, T_{j,i})$,
- **RHS Process:** $X_j=\tilde{x}_j=i$ if and only if $U_j\in[\tilde{T}_{j,i-1}, \tilde{T}_{j,i})$.

See Figure 1 for an illustration. Our construction guarantees that, considered in isolation, both the LHS and RHS process behave correctly.

At the beginning of this coupling, only the variables X_r corresponding to tokens $r\in\mathcal{R}$ are set, and these differ in the LHS and RHS process only for the index $\hat{r}\in\mathcal{R}$, for which we have $X_{\hat{r}}=x_{\hat{r}}$ (LHS) and $X_{\hat{r}}=\tilde{x}_{\hat{r}}=x_{\hat{r}}-1$ (RHS). Let $\iota\coloneqq x_{\hat{r}}$. For the first revealed token $j=\hat{r}+1$, this implies $\beta_{j,\iota}=\tilde{\beta}_{j,\iota}+1$, $\beta_{j,\iota-1}=\tilde{\beta}_{j,\iota-1}-1$, and $\beta_{j,i}=\tilde{\beta}_{j,i}$ for all $i\notin\{\iota,\iota-1\}$. By the definitions of the slots for both processes, we get $T_{j,i}=\tilde{T}_{j,i}$ for all $i\neq\iota-1$ and $T_{j,\iota-1}>\tilde{T}_{j,\iota-1}$ (cf. Figure 1). Thus, the LHS and RHS process behave different if and only if $U_i\in[\tilde{T}_{j,\iota-1}, T_{j,\iota-1})$. If this happens, we get $x_j<\tilde{x}_j$ (i.e., token j is assigned to a smaller neighbor in the LHS process). This implies $\boldsymbol{\beta_{j+1}}=\boldsymbol{\tilde{\beta}_{j+1}}$ and both processes behave identical from now on. Otherwise, if $U_i\notin[\tilde{T}_{j,\iota-1}, T_{j,\iota-1})$, we have $\boldsymbol{\beta_{j+1}}-\boldsymbol{\beta_{j+1}}=\boldsymbol{\beta_j}-\boldsymbol{\beta_j}$ and we can repeat the above argument. Thus, after all X_j are revealed, there is at most one $j\in\mathcal{L}$ for which $x_j\neq\tilde{x}_j$, and for this we have $x_j<\tilde{x}_j$. Since f is non-decreasing, this guarantees Inequality (3). To handle the case $x_{\hat{r}}-\tilde{x}_{\hat{r}}>1$, note that we can reorder the slots $[T_{j,i-1}, T_{j,i})$ used for the assignment of the variables such that the slots for $x_{\hat{r}}$ and $\tilde{x}_{\hat{r}}$ are neighboring. Formally, this merely changes in which order we consider the neighbors in the definition of the functions $T_{j,i}$. With this change, the same arguments as above apply. ◀

Before proving the majorization of tokens with random walks (Lemma 7) we require further notation. Let $\mathcal{S}$ denote our random Shuffle process, and $\mathcal{W}$ the random walk process in which each of the γn tokens performs an independent random walk according to the sequence of random walk matrices $(\boldsymbol{P}_t)_{t\in\mathbb{N}}$ (i.e., a token on u uses $P_t(u,\cdot)$ for the transition probabilities). We use $w_j^{\mathcal{P}}(t)$ to denote the position of token j after t steps of a

process $\mathcal{P}$. We assume (w.l.o.g.) $w_j^{\mathcal{S}}(0) = w_j^{\mathcal{W}}(0)$ for all j. While there are strong correlations between the tokens' movements in $\mathcal{S}$ (e.g., not all tokens can move to the same neighbor), Lemma 7 shows that these correlations are negative.

▶ **Lemma 7** (Majorizing RWs). *Consider a time $t \geq 0$, a token j, and node v. Let $B \subseteq [\gamma n]$ and $D \subseteq V$ be arbitrary subsets of tokens and nodes, respectively. The following holds:*
1. $\Pr\left(w_j^{\mathcal{S}}(t) = v\right) = \Pr\left(w_j^{\mathcal{W}}(t) = v\right)$ *and*
2. $\Pr\left(\bigcap_{j\in B}\left(w_j^{\mathcal{S}}(t) \in D\right)\right) \leq \Pr\left(\bigcap_{j\in B}\left(w_j^{\mathcal{W}}(t) \in D\right)\right) = \prod_{j\in B}\Pr\left(w_j^{\mathcal{W}}(t) \in D\right)$.

Proof. The first statement follows immediately from the definition of our process. For the second statement, note that the equality on the right-hand side holds trivially, since the tokens perform independent random walks in $\mathcal{W}$. To show the inequality, we define the intermediate process $\mathcal{SW}(t')$ ($t' \leq t$) that performs t' steps of $\mathcal{S}$ followed by $t - t'$ steps of $\mathcal{W}$. By this definition, $\mathcal{SW}(0)$ is identical to $\mathcal{W}$ restricted to t steps and, similar, $\mathcal{SW}(t)$ is identical to $\mathcal{S}$ restricted to t steps. Define

$$\mathcal{E}_{t'} \coloneqq \bigcap_{j\in B}\left(w_j^{\mathcal{SW}(t')}(t) \in D\right) \tag{4}$$

(the event that all tokens from B end up at nodes from D under process $\mathcal{SW}(t')$). The lemma's statement is equivalent to $\Pr(\mathcal{E}_t) \leq \Pr(\mathcal{E}_0)$. To prove this, we show $\Pr(\mathcal{E}_{t'+1}) \leq \Pr(\mathcal{E}_{t'})$ for all $t' \in \{0, 1, \ldots, t-1\}$. Combining these inequalities yields the desired result.

Fix an arbitrary $t' \in \{0, 1, \ldots, t-1\}$ and note that $\mathcal{SW}(t')$ and $\mathcal{SW}(t'+1)$ behave identical up to and including step t'. Hence, we can fix an arbitrary configuration (i.e., the location of each token) $c(t') = c$ immediately before time step $t'+1$. Remember that $u_j \in V$ denotes the location of j in configuration c. The auxiliary functions $h_j \colon [d(u_j)+1] \to [0,1]$ describe the probability that a random walk starting at time $t'+1$ from u_j's i-th neighbor ends up in a node from D. Formally,

$$h_j(i) \coloneqq \Pr\left(w_j^{\mathcal{W}}(t) \in D \;\middle|\; w_j^{\mathcal{W}}(t'+1) = N(u_j, i)\right). \tag{5}$$

We can assume (w.l.o.g.) that all h_j are non-decreasing (by reordering the neighborhood of u_j).

Now, by Lemma 6 the variables $(X_j)_{j\in B}$ satisfy the negative regression condition. Thus, we can apply Lemma 5 (a well-known characterization of negative regression) to the functions h_j. Using another simple auxiliary result (Claim 8) we can relate the (conditioned) probabilities of the events $\mathcal{E}_{t'}$ and $\mathcal{E}_{t'+1}$ to the expectations over the different $h_j(X_j)$. That is, for $p \coloneqq \Pr(\mathcal{E}_{t'+1} \mid c(t') = c)$ we compute

$$p \overset{Clm.\ 8(a)}{=} \mathbb{E}\left[\prod_{j\in B} h_j(X_j) \;\middle|\; c(t') = c\right] \overset{Lem.\ 5}{\leq} \prod_{j\in B}\mathbb{E}\left[h_j(X_j) \mid c(t') = c\right]$$
$$\overset{Clm.\ 8(b)}{=} \Pr(\mathcal{E}_{t'} \mid c(t') = c).$$

Using the law of total probability, we conclude $\Pr(\mathcal{E}_{t'+1}) \leq \Pr(\mathcal{E}_{t'})$, as required. ◀

▶ **Claim 8.** *Fix a time $t' \in \{0, 1, \ldots, t-1\}$ and consider an arbitrary configuration c. Then the following identities hold:*
(a) $\Pr(\mathcal{E}_{t'+1} \mid c(t') = c) = \mathbb{E}\left[\prod_{j\in B} h_j(X_j) \;\middle|\; c(t') = c\right]$, *and*
(b) $\Pr(\mathcal{E}_{t'} \mid c(t') = c) = \prod_{j\in B}\mathbb{E}\left[h_j(X_j) \mid c(t') = c\right]$.

Proof. Remember the definitions from Lemma 7 and its proof. We use the shorthand $d(u_j) = d_{t'+1}(u_j)$. Remember that each X_j indicates to which of the $d(u_j)+1$ neighbors of u_j (where u_j is considered a neighbor of itself) a token j moves during time step $t'+1$. Thus, given the configuration $c(t') = c$ immediately before time step $t'+1$, there is a bijection between any possible configuration $c(t'+1)$ and outcomes of the random variable vector $\boldsymbol{X} = (X_j)_{j\in[\gamma n]}$. Let $c_{\boldsymbol{x}}$ denote the configuration corresponding to a concrete outcome $\boldsymbol{X} = \boldsymbol{x} \in [d(u_j)+1]^{\gamma n}$. Thus, we have $\Pr\left(c(t'+1) = c_{\boldsymbol{x}} \mid c(t') = c\right) = \Pr\left(\boldsymbol{X} = \boldsymbol{x} \mid c(t') = c\right)$, and conditioning on $c(t'+1)$ is equivalent to conditioning on $\boldsymbol{X}$ and $c(t')$. For the claim's first statement, we calculate

$$\begin{aligned}
\Pr\left(\mathcal{E}_{t'+1} \;\middle|\; c(t') = c\right) &= \sum_{c_{\boldsymbol{x}}} \Pr\left(\mathcal{E}_{t'+1} \;\middle|\; c(t'+1) = c_{\boldsymbol{x}}\right) \cdot \Pr\left(c(t'+1) = c_{\boldsymbol{x}} \;\middle|\; c(t') = c\right) \\
&= \sum_{c_{\boldsymbol{x}}} \prod_{j\in B} \Pr\left(w_j^{\mathcal{SW}(t'+1)}(t) \in D \;\middle|\; \boldsymbol{X} = \boldsymbol{x}, c(t') = c\right) \cdot \Pr\left(\boldsymbol{X} = \boldsymbol{x} \;\middle|\; c(t') = c\right) \\
&= \sum_{c_{\boldsymbol{x}}} \prod_{j\in B} h_j(x_j) \cdot \Pr\left(\boldsymbol{X} = \boldsymbol{x} \;\middle|\; c(t') = c\right) \\
&= \sum_{\boldsymbol{x}} \prod_{j\in B} h_j(x_j) \cdot \Pr\left(\boldsymbol{X} = \boldsymbol{x} \;\middle|\; c(t') = c\right) = \mathbb{E}\left[\prod_{j\in B} h_j(X_j) \;\middle|\; c(t') = c\right].
\end{aligned}$$

Here, we first apply the law of total probability. Then, we use the bijection between $c(t'+1)$ and $\boldsymbol{X}$ (if $c(t')$ is given) and that the process $\mathcal{SW}(t'+1)$ consists of independent random walks if $c(t'+1)$ is fixed. Finally, we use the definition of the auxiliary functions $h_j(i)$, which equal the probability that a random walk starting at time $t'+1$ from u_j's i-th neighbor reaches a node from D.

For the claim's second statement, we do a similar calculation for the process $\mathcal{SW}(t')$. By definition, this process consists already from time t' onward of a collection of independent random walks. Thus, we can swap the expectation and the product in the last term of the above calculation, yielding the desired result. ◀

Separating the Plurality via Chernoff

▶ **Lemma 9** ([7, Lemma 3.1]). *Let $X_1, X_2, \ldots, X_n$ be a sequence of random variables with values in an arbitrary domain and let $Y_1, Y_2, \ldots, Y_n$ be a sequence of binary random variables with the property that $Y_i = Y_i(X_1, \ldots, X_i)$. If $\Pr\left(Y_i = 1 \mid X_1, \ldots, X_{i-1}\right) \le p$, then*

$$\Pr\left(\sum Y_i \ge \ell\right) \le \Pr\left(\mathrm{Bin}(n,p) \ge \ell\right) \tag{6}$$

and, similarly, if $\Pr\left(Y_i = 1 \mid X_1, \ldots, X_{i-1}\right) \ge p$, then

$$\Pr\left(\sum Y_i \le \ell\right) \le \Pr\left(\mathrm{Bin}(n,p) \le \ell\right). \tag{7}$$

Here, $\mathrm{Bin}(n,p)$ denotes the binomial distribution with parameters n and p.

We are finally able to prove the following Chernoff-like bound.

▶ **Lemma 10** (Token Concentration). *Consider any subset B of tokens, a node $u \in V$, and an integer T. Let $X \coloneqq \sum_{t\le T}\sum_{j\in B} X_{j,t}$, where $X_{j,t}$ is 1 if token j is on node u at time $t \cdot t_{mix}$. With $\mu \coloneqq (1/n + 1/n^5)\cdot|B|\cdot T$, we have $\Pr\left(X \ge (1+\delta)\cdot\mu\right) \le e^{\delta^2\mu/3}$.*

Proof. Let $v_{j,t}$ denote the location of token j at time $(t-1)\cdot t_{\mathrm{mix}}$. For all $t \le T$ and $\ell \in \mathbb{N}$ define the random indicator variable $Y_{j,t}$ to be 1 if and only if the random walk starting at

$v_{j,t}$ is at node u after t_{mix} time steps. By Lemma 7 we have for each $B' \subseteq B$ and $t \leq T$ that

$$\Pr\left(\bigcap_{i\in B'} X_{j,t} = 1\right) \leq \prod_{j\in B'} \Pr\left(Y_{j,t} = 1\right). \tag{8}$$

Hence for all $t \leq T$ and $\ell \in \mathbb{N}$ we have $\Pr\left(\sum_{j\in B} X_{j,t} \geq \ell\right) \leq \Pr\left(\sum_{j\in B} Y_{j,t} \geq \ell\right)$ and

$$\Pr\left(X \geq \ell\right) = \Pr\left(\sum_{t\leq T}\sum_{j\in B} X_{j,t} \geq \ell\right) \leq \Pr\left(\sum_{t\leq T}\sum_{j\in B} Y_{j,t} \geq \ell\right). \tag{9}$$

Let us define $p \coloneqq 1/n + 1/n^5$. By the definition of t_{mix}, we have for all $j \in B$ and $t \leq T$ that

$$\Pr\left(Y_{j,t} = 1 \;\middle|\; Y_{1,1}, Y_{2,1}, \ldots, Y_{|B|,1}, Y_{1,2}, \ldots, Y_{j-1,t}\right) \leq p. \tag{10}$$

Combining our observations with Lemma 9 (see above), we get $\Pr\left(X \geq \ell\right) \leq \text{Bin}(T \cdot |B|, p)$. Recall that $\mu = T \cdot |B| \cdot p$. Thus, by applying standard Chernoff bounds we get

$$\Pr\left(X \geq (1+\delta)\mu\right) \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \leq e^{\delta^2\mu/3}, \tag{11}$$

which yields the desired statement. ◀

Together, these lemmas generalize a result given in [32] to a setting with considerably more dependencies. Equipped with this Chernoff bound, we prove concentration of the counter values.

▶ **Lemma 11** (Counter Seperation). *Let $c \geq 12$. For every time $t \geq c \cdot T \cdot t_{mix}$ there exist values $\ell_\top > \ell_\bot$ such that*

(a) *For all nodes w with $o_w \geq 2$ we have (w.h.p.) $c_w \leq \ell_\bot$.*

(b) *For all nodes v with $o_v = 1$ we have (w.h.p.) $c_v \geq \ell_\top$.*

Proof. For two nodes v and w with $o_v = 1$ and $o_w \geq 2$, $\mu_i \coloneqq (1/n + 1/n^5)c \cdot T \cdot \gamma \cdot n_k$ for all $i \in [k]$, and $\mu' \coloneqq (1/n + 1/n^5)c \cdot T \cdot \gamma \cdot (n - n_1)$. For $i \in [k]$ define

$$\ell_\bot(i) \coloneqq \mu_i + \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \frac{n_i}{n}} \qquad \text{and} \qquad \ell_\top \coloneqq T\gamma - \mu' - \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \frac{n - n_1}{n}}.$$

We set $\ell_\bot \coloneqq \ell_\bot(2)$. It is easy to show that $\ell_\bot < \ell_\top$. Now, let all γn tokens be labeled from 1 to γn. It remains to prove the lemma's statements:

- For the first statement, consider a node w with $o_w \geq 2$ and set $\lambda(o_w) \coloneqq \ell_\bot(o_w) - \mu_{o_w} = \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \cdot n_{o_w}/n}$. Set the random indicator variable $X_{i,t}$ to be 1 if and only if i is on node w at time t and if i's label is o_w. Let $c_w = \sum_{i\in[\gamma n]} \sum_{j\leq T} X_{i,j\cdot t_{\text{mix}}}$. We compute

$$\begin{aligned}\Pr\left(c_w \geq \ell_\bot\right) \leq \Pr\left(c_w \geq \mu_{o_w} + \lambda(o_w)\right) &= \Pr\left(c_w \geq \left(1 + \frac{\lambda(o_w)}{\mu_{o_w}}\right) \cdot \mu_{o_w}\right) \\ &\leq \exp\left(-\frac{\lambda^2(o_w)}{3\mu_{o_w}}\right) \leq \exp\left(-\frac{c}{6}\log n\right),\end{aligned} \tag{12}$$

  where the last line follows by Lemma 10 applied to $c_w = \sum_{i\in[\gamma n]} \sum_{j\leq T} X_{i,j\cdot t_{\text{mix}}}$ and setting B to the set of all tokens with label o_w. Hence, the claim follows for c large enough after taking the union bound over all $n - n_1 \leq n$ nodes w with $o_w \geq 2$.

- For the lemma's second statement, consider a node v with $o_v = 1$ and set $\lambda' := \mu' - \ell_\top$. Define the random indicator variable $Y_{i,t}$ to be 1 if and only if token i is on node v at time t and if i's label is not 1. Set $Y = \sum_{j \le T} \sum_{i \in [\gamma n]} Y_{i,j \cdot t_{\text{mix}}}$ and note that $c_v = T\gamma - Y$. We compute

$$\begin{aligned}\Pr\left(c_v \le \ell_\top\right) &= \Pr\left(T\gamma - Y \le \ell_\top\right) = \Pr\left(T\gamma - Y \le T\gamma - \mu' - \lambda'\right) = \Pr\left(Y \ge \mu' + \lambda'\right) \\ &= \Pr\left(Y \ge \left(1 + \frac{\lambda'}{\mu'}\right) \cdot \mu'\right) \le \exp\left(-\frac{\lambda'^2}{3\mu'}\right) \le \exp\left(\frac{c}{6} \log n\right),\end{aligned}$$

  where the first inequality follows by Lemma 10 applied to Y and using B to denote the set of all tokens with a label other than 1. Hence, the claim follows for c large enough after taking the union bound over all $n_1 \le n$ nodes v with $o_u \ge 2$. ◀

We now give the proof of our main theorem.

Proof of Theorem 2. Fix an arbitrary time $t \in [c \cdot T \cdot t_{\text{mix}}, N]$ with $t_{\text{mix}} \mid t$, where c is the constant from the statement of Lemma 11. From Lemma 11 we have that (w.h.p.) the node u with the highest counter c_u has $o_u = 1$ (ties are broken arbitrarily). In the following we condition on $o_u = 1$. We claim that at time $t' = t + t_{\text{mix}}$ all nodes $v \in V$ have $plu_v = 1$. This is because the counters during the "broadcast part" (Lines 4 to 8) propagate the highest counter received after time t. The time τ until all nodes $v \in V$ have $plu_v = 1$ is bounded by the mixing by definition: In order for $[t, t']$ to be $1/n^5$-smoothing, the random walk starting at u at time t is with probability at least $1/n - 1/n^5$ on node v and, thus, there exists a path from u to v (with respect to the communication matrices). If there is such a path for every node v, the counter of u was also propagated to that v and we have $\tau \le t_{\text{mix}}$. Consequently, at time t' all nodes have the correct majority opinion. This implies the desired time bound. For the memory requirements, note that each node u stores γ tokens with a label from the set $[k]$ ($\gamma \cdot \mathrm{O}(\log k)$ bits), three opinions (its own, its plurality guess, and the dominating opinion; $\mathrm{O}(\log k)$ bits), the two counters c_u and e_u and the time step counter. The memory to store the counter c_u and e_u is $\mathrm{O}(\gamma T)$. Finally, the time step counter is bounded by $\mathrm{O}(\log(T \cdot t_{\text{mix}}))$ bits. This yields the claimed space bound. ◀

4 Protocol Balance

Protocol Description. The idea of our Balance protocol is quite simple: Every node u stores a k-dimensional vector $\boldsymbol{\ell_t(u)}$ with k integer entries, one for each opinion. Balance performs an entry-wise load balancing on $\boldsymbol{\ell_t(u)}$ according to the communication pattern $\boldsymbol{M} = (\boldsymbol{M}_t)_{t \le N}$ and the corresponding transition matrices $\boldsymbol{P}_t$ (cf. Section 2). Once the load is properly balanced, the nodes look at their largest entry and assume that this is the plurality opinion (stored in the variable plu_u).

In order to ensure a low memory footprint, we must not send fractional loads over active edges. To this end, we use a rounding scheme from [12, 32], which works as follows: Consider a dimension $i \in [k]$ and let $\ell_{i,t}(u) \in \mathbb{N}$ denote the current (integral) load at u in dimension i. Then u sends $\lfloor \ell_{i,t}(u) \cdot \boldsymbol{P}_t[u, v] \rfloor$ tokens to all neighbors v with $\boldsymbol{M}_t[u, v] = 1$. This results in at most $d_t(u)$ remaining *excess tokens* ($\ell_{i,t}(u)$ minus the total number of tokens sent out). These are then randomly distributed (without replacement), where neighbor v receives a token with probability $\boldsymbol{P}_t[u, v]$. In the following we call the resulting balancing algorithm *vertex-based balancing* algorithm. The formal description of protocol Balance is given in Listing 2.

```
1  for i ∈ [k]:
2    for { u, v } ∈ E with M_t[u, v] = 1:
3      send ⌊ℓ_{i,t}(u) · P_t[u, v]⌋ tokens from dimension i to v
4    x := ℓ_{i,t}(u) − Σ_{v: M_t[u,v]=1} ⌊ℓ_{i,t}(u) · P_t[u, v]⌋      {excess tokens}
5    randomly distribute x tokens such that:
6      every v ≠ v with M_t[u, v] = 1 receives 1 token w.p. P_t[u, v]
7      (and zero otherwise)
8  plu_u := i with ℓ_{i,t}(u) ≥ ℓ_{j,t}(u)   ∀1 ≤ i, j ≤ k          {plurality guess}
```

Listing 2 Protocol Balance as executed by node u at time t. At time zero, each node initializes $\ell_{o_u,0}(u) := \gamma$ and $\ell_{j,0}(u) := 0$ for all $j \neq o_u$.

Analysis of Balance. Consider initial load vectors $\boldsymbol{\ell_0}$ with $\|\boldsymbol{\ell_0}\|_\infty \leq n^5$. Let $\tau := \tau(g, \boldsymbol{M})$ be the first time step when Vertex-Based Balancer under the (fixed) communication pattern $\boldsymbol{M} = (\boldsymbol{M}_t)_{t \leq N}$ is able to balance any such vector $\boldsymbol{\ell_0}$ up to a g-discrepancy. With this, we show:

▶ **Theorem 12.** *Let $\alpha = \frac{n_1 - n_2}{n} \in [1/n, 1]$ denote the initial bias. Consider a fixed communication pattern $\boldsymbol{M} = (\boldsymbol{M}_t)_{t \leq N}$ and an integer $\gamma \in [3 \cdot \frac{g}{\alpha}, n^5]$. Protocol* Balance *ensures that all nodes know the plurality opinion after $\tau(g, \boldsymbol{M})$ rounds and requires $k \cdot \log(\gamma)$ memory bits per node.*

Proof. Recall that $\gamma \geq 3\frac{g}{\alpha} = 3g \cdot \frac{n}{n_1 - n_2}$. For $i \in [k]$ let $\bar{\ell}_i := n_i \cdot \gamma / n$. The definition of $\tau(g, \boldsymbol{M})$ implies $\ell_{1,t}(u) \geq \bar{\ell}_1 - g$ and $\ell_{i,t}(u) \leq \bar{\ell}_i + g$ for all nodes u and $i \geq 2$. Consequently, we get

$$\ell_{1,t}(u) - \ell_{i,t}(u) \geq \bar{\ell}_1 - \bar{\ell}_i - 2g = 3g \cdot \frac{n_1 - n_i}{n_1 - n_2} - 2g > 0\,. \tag{13}$$

Thus, every node u has the correct plurality guess at time t. ◀

The memory usage of Balance depends on the number of opinions (k) and on the number of tokens generated on every node (γ). The algorithm is very efficient for small values of k but it becomes rather impractical if k is large. Note that if one chooses γ sufficiently large, it is easy to adjust the algorithm such that every node knows the frequency of *all* opinions in the network. The next corollary gives a few concrete examples for common communication patterns on general graphs.

▶ **Corollary 13.** *Let G be an arbitrary d-regular graph.* Balance *ensures that all nodes agree on the plurality opinion with probability $1 - e^{-(\log(n))^c}$ for some constant c*

- **(a)** *using $\mathrm{O}(k \cdot \log n)$ bits of memory in time $\mathrm{O}\big(\frac{\log n}{1-\lambda_2}\big)$ in the diffusion model,*
- **(b)** *using $\mathrm{O}(k \cdot \log n)$ bits of memory in time $\mathrm{O}\big(\frac{1}{d \cdot p_{min}} \cdot \frac{\log n}{1-\lambda_2}\big)$ in the random matching model,*
- **(c)** *using $\mathrm{O}\big(k \cdot \log(\alpha^{-1})\big)$ bits of memory in time $\mathrm{O}\big(d \cdot \frac{\log n}{1-\lambda_2}\big)$ in the balancing circuit model, and*
- **(d)** *using $\mathrm{O}\big(k \cdot \log(\alpha^{-1})\big)$ bits of memory in time $\mathrm{O}\big(n \cdot \frac{\log n}{1-\lambda_2}\big)$ in the sequential model.*

Proof. Part (a) follows directly from [33, Theorem 6.6] and Part (c) follows directly from [33, Theorem 1.1]. To show Part (b) and (d) we choose τ such that $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_\tau$ enable Vertex-Based Balancer to balance any vector $\boldsymbol{\ell_0}$ (with initial discrepancy of at most n^5) up to a *g-discrepancy*. The bound on τ then follows from [33, Theorem 1.1]. ◀

References

1. D. Aldous and J. Fill. Reversible markov chains and random walks on graphs, 2002. Unpublished. http://www.stat.berkeley.edu/~aldous/RWG/book.html.
2. Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, (PODC)*, pages 47–56, 2015.
3. Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.
4. Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
5. James Aspnes and Eric Ruppert. An introduction to population protocols. *Bulletin of the EATCS*, 93:98–117, 2007.
6. Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Automata, Languages and Programming, 35th International Colloquium, ICALP*, pages 121–132, 2008.
7. Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal of Computing*, 29(1):180–200, 1999.
8. Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Riccardo Silvestri. Plurality consensus in the gossip model. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 371–390, 2015. doi:10.1137/1.9781611973730.27.
9. Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, Riccardo Silvestri, and Luca Trevisan. Simple dynamics for plurality consensus. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, (SPAA)*, pages 247–256, 2014. doi:10.1145/2612669.2612677.
10. Florence Bénézit, Patrick Thiran, and Martin Vetterli. Interval consensus: From quantized gossip to voting. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pages 3661–3664. IEEE, 2009.
11. Florence Bénézit, Patrick Thiran, and Martin Vetterli. The distributed multiple voting problem. *J. Sel. Topics Signal Processing*, 5(4):791–804, 2011.
12. Petra Berenbrink, Colin Cooper, Tom Friedetzky, Tobias Friedrich, and Thomas Sauerwald. Randomized diffusion for indivisible loads. *J. Comput. Syst. Sci.*, 81(1):159–185, 2015.
13. Petra Berenbrink, Tom Friedetzky, George Giakkoupis, and Peter Kling. Efficient plurality consensus, or: The benefits of cleaning up from time to time. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2016. to appear.
14. Stephen P. Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.
15. Luca Cardelli and Attila Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific reports*, 2, 2012.
16. Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from dna. *Nature nanotechnology*, 8(10):755–762, 2013.
17. Andrea E. F. Clementi, Miriam Di Ianni, Giorgio Gambosi, Emanuele Natale, and Riccardo Silvestri. Distributed community detection in dynamic graphs. *Theor. Comput. Sci.*, 584:19–41, 2015.
18. Colin Cooper, Robert Elsässer, and Tomasz Radzik. The power of two choices in distributed voting. In *Automata, Languages, and Programming - 41st International Colloquium, (ICALP)*, pages 435–446, 2014.

19 Colin Cooper, Robert Elsässer, Tomasz Radzik, Nicolas Rivera, and Takeharu Shiraga. Fast consensus for voting on general expander graphs. In *Proceedings of the 29th International Symposium on Distributed Computing (DISC)*, pages 248–262, 2015. doi:10.1007/978-3-662-48653-5_17.

20 Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. Stabilizing consensus with the power of two choices. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, (SPAA)*, pages 149–158, 2011.

21 Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. *SIAM J. Control and Optimization*, 50(3):1087–1109, 2012.

22 Devdatt P. Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Struct. Algorithms*, 13(2):99–124, 1998.

23 Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Frederik Mallmann-Trenn, and Horst Trinker. Efficient k-party voting with two choices. *CoRR*, abs/1602.04667, 2016. URL: http://arxiv.org/abs/1602.04667.

24 Bhaskar Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *J. Comput. Syst. Sci.*, 53(3):357–370, 1996. doi:10.1006/jcss.1996.0075.

25 David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings 44th Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2003.

26 Fabian Kuhn, Thomas Locher, and Stefan Schmid. Distributed computation of the mode. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 15–24, 2008.

27 George B. Mertzios, Sotiris E. Nikoletseas, Christoforos Raptopoulos, and Paul G. Spirakis. Determining majority in networks with local interactions and very small local memory. In *Automata, Languages, and Programming - 41st International Colloquium, (ICALP)*, pages 871–882, 2014.

28 Elchanan Mossel, Joe Neeman, and Omer Tamuz. Majority dynamics and aggregation of information in social networks. *Autonomous Agents and Multi-Agent Systems*, 28(3):408–429, 2014. doi:10.1007/s10458-013-9230-4.

29 Elchanan Mossel and Grant Schoenebeck. Reaching consensus on social networks. In *Innovations in Computer Science - (ICS)*, pages 214–229, 2010.

30 David Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theor. Comput. Sci.*, 282(2):231–257, 2002. doi:10.1016/S0304-3975(01)00055-X.

31 Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *28th IEEE International Conference on Computer Communications, (INFOCOM)*, pages 2527–2535, 2009.

32 Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 341–350, 2012.

33 Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. *CoRR*, abs/1201.2715, 2012. full version of FOCS'12. arXiv:1201.2715.