Communication Patterns for Randomized Algorithms

CHRISTOPHER MICHAEL WASTELL

Communication Patterns for Randomized Algorithms

Christopher M. Wastell

A thesis presented for the degree of

Doctor of Philosophy

Engineering and Computing Sciences

Durham University

UK

August 2017

Abstract

Examples of large scale networks include the Internet, peer-to-peer networks, parallel computing systems, cloud computing systems, sensor networks, and social networks. Efficient dissemination of information in large networks such as these is a fundamental problem. In many scenarios the gathering of information by a centralised controller can be impractical. When designing and analysing distributed algorithms we must consider the limitations imposed by the heterogeneity of devices in the networks. Devices may have limited computational ability or space. This makes randomised algorithms attractive solutions. Randomised algorithms can often be simpler and easier to implement than their deterministic counterparts. This thesis analyses the effect of communication patterns on the performance of distributed randomised algorithms. We study randomized algorithms with application to three different areas.

Firstly, we study a generalization of the balls-into-bins game. Balls into bins games have been used to analyse randomised load balancing. Under the GREEDY$[d]$ allocation scheme each ball queries the load of d random bins and is then allocated to the least loaded of them. We consider an infinite, parallel setting where expectedly λn balls are allocated in parallel according to the GREEDY$[d]$ allocation scheme in to n bins and subsequently each non-empty bin removes a ball. Our results show that for $d = 1, 2$, the GREEDY$[d]$ allocation scheme is self-stabilizing and that in *any* round the maximum system load for high arrival rates is exponentially smaller for $d = 2$ compared to $d = 1$ (w.h.p).

Secondly, we introduce protocols that solve the plurality consensus problem on arbitrary graphs for arbitrarily small bias. Typically, protocols depend heavily on the employed communication mechanism. Our protocols are based on an interesting relationship between plurality consensus and distributed load balancing. This relationship allows us to design protocols that are both time and space efficient and generalize the state of the art for a large range of problem parameters.

Finally, we investigate the effect of restricting the communication of the classical PULL algorithm for randomised rumour spreading. Rumour spreading (broadcast) is a fundamental task in distributed computing. Under the classical PULL algorithm, a node with the rumour that receives multiple requests is able to respond to all of them in a given round. Our model restricts nodes such that they can respond to at most one request per round. Our results show that the restricted PULL algorithm is optimal for several graph classes such as complete graphs, expanders, random graphs and several Cayley graphs.

Contents

List of Figures

List of Tables

Declaration

Parts of the work presented in this thesis have been published in preliminary form [24, 25] in collaboration with my colleagues.

Statement of Copyright

Acknowledgements

First and foremost I must thank my supervisor Dr. Tom Friedetzky for his guidance and support throughout my studies. Moreover, I am indebted to him for my coffee appreciation (addiction) without which this work would certainly not have been possible.

Special thanks go to my co-authors and colleagues Prof. Dr. Petra Berenbrink, Dr. Peter Kling, Frederik Mallmann-Trenn, and Dr. Lars Nagel for their invaluable and enjoyable collaborations during time spent in both Durham and Vancouver. I would also like to thank my examiners, Dr. Russell Martin and Dr. Ioannis Ivrissimtzis.

My thanks go to EPSRC, Durham University, and Simon Fraser University for the financial support that has allowed me to produce this thesis.

During my time in Durham I have been fortunate to meet many great people. Without them I am sure that my experiences in Durham would not have been anywhere near as enjoyable. Although it is not possible to name them all I must thank Dr. David Roberts, Dr. Chris Watson, Dr. Robert Powell, and (future Doctor) Lewis McArd for their time, guidance, and life advice.

Finally, thank you to my friends and family for their support during my time in Durham. They have all been endlessly patient and understanding during my studies. I am sure there were many points where you thought I was crazy for undertaking such studies and you may yet be right!

Chapter 1

Introduction

There are numerous examples of large scale networks. Many of these are prevalent in well known applications. The Internet, peer-to-peer (P2P) networks, parallel computing systems, Cloud computing systems, sensor networks, and social networks are all well known examples of large scale networks to name but a few. Given the wide spread application of these networks, the study and design of efficient algorithms for related problems is highly relevant. This has lead to the development of numerous models that aid the development and understanding of algorithms for problems related to such networks.

Each of the previous examples of networks has their own requirements that must be accounted for when designing suitable models and algorithms. The challenges that any algorithm must cope with may include but are not limited to dynamic changes in the network, heterogeneous devices, communicational limitations, and failure of both nodes and communication channels. Whilst present in the previous examples, these challenges are highlighted by emerging paradigms such as the Internet of Things (IoT) [10]. The Internet of Things considers the connection of everyday objects through embedded computing devices. In this paradigm, a single network can consist of many different types of devices e.g. computers, mobile phones, sensors, and fridges. The computational ability of these devices can vary greatly. For example, whilst desktop/laptop computers have widely adopted multi-core processors (with mobiles phones following this pattern) other devices such as

sensors and other embedded devices operate with limited computational and communication resources [27]. The ability of an algorithm to cope with the challenges outlined above ultimately determines the suitability of the algorithm.

The first (and most immediate) aspect of these networks that any potential algorithm must cope with is the size of the network. The size of a network is typically characterised by the number of nodes that constitute the network. In order to be applicable to networks such as the previous examples, models and algorithms must assume that the number of nodes in the network is large. For example, the number of devices connected to the Internet is estimated to be in the billions and this number continues to grow. In 2013, Cisco predicted that the number of devices connected to the Internet will hit 50 billion by 2020. [1] Other sources suggested this figure may even be an underestimate. By 2016, this prediction had been revised down however the number of devices connected to the internet is of the order of 10^9 and continues to grow. [2] The size of the network presents a major hurdle for certain approaches that might be considered when designing our algorithms. For example, the adoption of centralised approaches may not be suitable when considering problems on large networks. Where a centralised controller is required to solve a problem that requires knowledge of the network as a whole this will typically entail the gathering of information from the nodes of the network. This information could represent the current state of the nodes in the network e.g. for load balancing problems this might represent the nodes current load. The gathering of this information by a centralised controller can be impractical since it may be costly and time consuming. In many scenarios that consider large scale networks it is therefore assumed that there is no central controller present.

In the absence of a centralised controller it is necessary to develop decentralised algorithms for our problems. That is we are interested in designing algorithms that are executed locally on the nodes of the network. In particularly, these algorithms

[1] https://blogs.cisco.com/news/cisco-connections-counter

[2] http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

should use only local knowledge when making decisions. This is to avoid incurring the previously discussed costs associated with gathering the information centrally. Adopting a decentralised approach has the additional benefit that unlike the centralised approach that can have a single point of failure, decentralised approaches allow a system to be more resilient to changes in the network. This is an attractive feature when considering networks that are subject to change such as nodes leaving or joining the network.

In order to take advantage of the benefits of a decentralised approach there are additional challenges and considerations that arise due to the local execution of the algorithm that we must be mindful of. Firstly, we require that each node uses only local knowledge. In other words, each node must execute the algorithm based on only partial knowledge of the whole problem. Therefore, in many cases nodes must communicate in order to solve the problem. For this reason special attention must be given to the communication required by our algorithms. As stated previously, attempting to communicate with all nodes in the network is often infeasible due to the costs and memory overheads. For this reason, communication is often assumed to be restricted to adjacent nodes in the network.

Secondly, special attention must be given to the capabilities of the nodes. When a network consists of heterogeneous devices, the nodes in the network may vary greatly in their computational abilities. It is therefore necessary to design algorithms that incur low computational overheads. In doing so we ensure that our algorithm is suitable to be run on all nodes in the network. Additionally, in networks where the nodes are limited in their capabilities this may introduce further restrictions on the communication that is possible between nodes. For example, depending on the capability of the node it may be able to communicate with many neighbouring nodes concurrently or may be restricted to communicating with only with a single node.

Given the locality considerations, special attention must be paid to the communication between nodes in the network. In particular the efficient dissemination of

information in large scale networks is a fundamental problem [64]. Tasks such as broadcasting, gossiping, sorting, routing, leader election, load balancing, etc. can be regarded as special cases of information dissemination [88]. In subsequent chapters we study models and algorithms for three of these problems: namely load balancing, plurality consensus, and rumour spreading. Due to their importance these problems have been widely studied under different models and assumptions. Further discussion of previous studies can be found in the subsequent chapters.

When designing algorithms for problems related to the efficient dissemination of information it must be done so with the previously mentioned considerations and constraints in mind. That is, we require our algorithms to be capable of solving their respective problems through local decision making whilst incurring low computational overheads. In many cases randomised algorithms have often been seen as attractive solutions. This is in part due to the robustness of randomised algorithms. Whilst it may be possible to formulate a special input where a deterministic algorithm performs poorly, it can be more difficult to do the same for a randomised algorithm. Randomised algorithms not only serve a purpose where the deterministic counterpart is inappropriate, there are applications such as Monte Carlo simulations and primality testing where randomised algorithms are significantly more efficient than deterministic solutions [77]. Moreover, randomised algorithms can be easier to implement than their deterministic counter parts. As ever there is a price to be paid for these advantageous properties. When adopting a randomised algorithm we have to be aware that it is possible that the solution may be of poor quality or even incorrect. It is therefore important that proposed algorithms are shown to have a small probability of failure for the improvement in speed or memory requirements to be worthwhile.

In this thesis we consider the effect of communication patterns on the performance of randomised algorithms for problems relating to the efficient dissemination of information in large networks. As previously stated, communication between the nodes in large networks may be restricted. This may be due to, for example, the

application considered, the network topology, or device limitations. Subsequently in these cases the communication patterns may be determined by the limitations of the devices themselves e.g. devices in sensor networks. On the other hand, it may also be the case that communication between nodes is restricted in order to minimise overheads. For example, when considering the problem of load balancing it is desirable to reduce the communication overheads that are incurred by the centralised algorithm to avoid costly information gathering. In this case it is advantageous to design algorithms that restrict communication.

1.1 Notation and Terminology

In this section we introduce some basic notation and terminology. In subsequent sections further specialised notation will be introduced where necessary. We assume the reader is familiar with concepts in basic combinatorics and probability theory that are used in the analysis of randomized algorithms. For good introductions see [62, 77, 81, 87]. Throughout this thesis we will use asymptotic notation. For an overview we refer the reader to Graham et al. [62, Chapter 9].

In general, bold font indicates vectors and matrices, and $x(i)$ refers to the i-th component of $\boldsymbol{x}$. For $i \in \mathbb{N}$, we define $[i] \coloneqq \{1, 2, \ldots, i\}$ as the set of the first i integers.

1.1.1 Basic Definitions

Definition 1.1.1 (Binomial Distribution; See [81])**.** *Let $X_1, X_2, \ldots X_n$ be independent 0/1 random variables such that $\Pr(X_i = 1) = p$ and $X = \sum^n X_i$. The random variable X has the binomial distribution with parameter n and p.*

$$\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Definition 1.1.2 (Binomial Theorem)**.**

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i} \tag{1.1}$$

By substituting $y = 1$ *we obtain the following special case*

$$(1+x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i \tag{1.2}$$

Since the binomial distribution is used regularly throughout this thesis we give the following useful bounds for the binomial coefficient.

Proposition 1.1.3 (See Motwani and Raghavan [81])**.** *For* $n \geq k > 0$*,*

$$\binom{n}{k} \leq \frac{n^k}{k!} \tag{1.3}$$

$$\binom{n}{k} \geq \left(\frac{n}{k}\right)^k \tag{1.4}$$

$$\binom{n}{k} \leq \left(\frac{e \cdot n}{k}\right)^k \tag{1.5}$$

The following identities are useful for obtaining bounds.

Definition 1.1.4 (Geometric Series)**.** *If* $x \in \mathbb{R}$ *and* $|x| < 1$ *then,*

$$\sum^{\infty} x^k = \frac{1}{1-x}$$

Lemma 1.1.5 (Property of Exponential Function)**.** *For all* $x \in \mathbb{R}$ *and* $n > 0$*,*

$$\left(1 + \frac{x}{n}\right)^n \leq e^x$$

Key quantities related to the performance of randomised algorithms are described by

random variables. To compare algorithms we will compare these random variables. The following definitions will be useful when comparing two random variables.

Definition 1.1.6 (Stochastic Dominance)**.** *Let X and Y be random variables on a sample space Ω. If*

$$\Pr(X \geq c) \geq \Pr(Y \geq c)$$

for all c then X stochastically dominates Y, denoted by $(Y \prec X)$.

1.1.2 Deviation Bounds

Since it is possible for a randomised algorithm to perform "badly" it is often desirable to show that the values of the random variable associated with key quantities associated with the algorithm are within a certain range *with high probability* (w.h.p.). The definition is based on a parameter n, that is related to the problem size.

Definition 1.1.7. *An event $\mathcal{E}$ holds with high probability (w.h.p) if*

$$\Pr(\mathcal{E}) \geq 1 - \frac{1}{n^c}$$

for a constant $c > 0$.

The following tools have been developed to show tail bounds for random variables. Combined with our other techniques, these tools allow us to show desirable properties of our randomized algorithms.

Lemma 1.1.8 (Markov's; See [81] pg.46)**.** *Let X be a random variable assuming only non negative values. Then for all $a > 0$*

$$\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

Lemma 1.1.9 (Chernoff Bound; See Dubhashi and Panconesi [45] pg.6)**.** *Let* $X = \sum_{i\in[n]} X_i$ *where* X_i *are independent* $0/1$ *random variables. Then,*

For $0 < \varepsilon \leq 1$

$$\Pr\left(X > (1+\varepsilon)\cdot\mathbb{E}\left[X\right]\right) \leq \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}}\right)^{\mathbb{E}[X]} \leq \exp\left(-\frac{\varepsilon^2}{3}\cdot\mathbb{E}\left[X\right]\right) \tag{1.6}$$

and for $0 < \varepsilon < 1$

$$\Pr\left(X < (1-\varepsilon)\cdot\mathbb{E}\left[X\right]\right) \leq \left(\frac{e^{-\varepsilon}}{(1-\varepsilon)^{(1-\varepsilon)}}\right)^{\mathbb{E}[X]} \leq \exp\left(-\frac{\varepsilon^2}{2}\cdot\mathbb{E}\left[X\right]\right) \tag{1.7}$$

1.1.3 Graphs

When discussing distributed systems on networks we model the network as a graph. A (simple) undirected graph $G = (V, E)$ is defined as a pair of sets V and E. V denotes the vertex set. $E \subseteq V \times V$ denotes the edge set. If the tuple $(u, v) \in E$ for $u, v \in V$ then we say that vertices u and v are *adjacent*. Let $N(u)$ denote the neighbourhood of a vertex u. The *neighbourhood* of a vertex is defined to be the set of vertices adjacent to u. The *degree* of a vertex u is defined to be the number of adjacent edges i.e, $|N(u)|$.

Alternatively, it is also possible to consider the matrix representation of a graph. For an undirected graph $G = (V, E)$, let $\mathbf{A}$ be the adjacency matrix. $\mathbf{A}$ is defined as follows,

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } i, j \in E \\ 0 & \text{otherwise} \end{cases}$$

1.1.4 Markov Chains

A *Markov Chain* is a discrete time, stochastic process $M = (S, \mathbf{P})$ where S denotes the set of states and $\mathbf{P}$ the transition matrix. For a Markov Chain in state $i \in S$,

the entry $\mathbf{P}_{i,j}$ defines the probability that the next state will be $j \in S$. Note that Markov Chains are *Memoryless*. That is that the next state only depends on the current state. This is also referred to as the Markov Property. More formally, for a Markov Chain M, let X_t denote the state of M at time t. The following equation describes the memoryless property.

$$\begin{aligned}\Pr\left(X_{t+1} = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_t = i_t\right) &= \Pr\left(X_{t+1} = j \mid X_t = i_t\right) \\ &= \mathbf{P}_{i,j}\end{aligned}$$

Let $q^t = (q_1^t, q_2^t, \ldots q_n^t)$ be a n-dimensional probability vector. q_i^t denotes the i-th entry in q^t and describes the probability that a Markov chain is in state i at time t.

A Markov chain is *irreducible* if for any two states x, y there exists an integer t such that $\Pr\left(q_x^t \mid X_0 = y\right) > 0$. In other words, it is possible to get from one state to any other state. A Markov chain is *aperiodic* if the greatest common divisor for the return time is 1.

$$\gcd\{n > 0 : \Pr\left(X_n = i \mid X_0 = i\right) > 0\} = 1$$

for all states i. Otherwise it is *periodic*. A state is *ergodic* if it is aperiodic and positive recurrent. In this case, positive recurrent refers to the expected return time being finite. A Markov chain is ergodic if all states are ergodic. In particular, this implies the existence of a unique stationary distribution.

Definition 1.1.10 (Stationary Distribution)**.** *Let* $\mathbf{P}$ *be the transition matrix of an ergodic Markov Chain* M*. Let* π *be a probability distribution such that,*

$$\pi = \pi\mathbf{P}$$

then π *denotes the stationary distribution.*

See Levin and Perres [72] for an excellent introduction into Markov chains and the involved terminology.

1.1.5 Random Walks

Let $G = (V, E)$ be a connected, undirected graph. A *random walk* is a Markov Chain M_G such that the states of M_G are the vertices of G. The transition matrix $\mathbf{P}$ of M_G is defined as follows.

$$\mathbf{P}_{i,j} = \begin{cases} \frac{1}{deg(u)} & \text{if } i, j \in E \\ 0 & \text{otherwise} \end{cases}$$

When discussing random walks, the following quantities will be of particular interest

Definition 1.1.11 (Hitting Time)**.** *The hitting time of a random walk $H(u, v)$ is the expected number of steps for the random walk starting at u to reach v for the first time*

Definition 1.1.12 (Variation Distance)**.** *For two probability vectors μ and ν let*

$$\|\mu - \nu\| = \frac{1}{2} \sum_{i=1}^{n} |\mu_i - \nu_i|$$

be the variation distance.

Definition 1.1.13 (Mixing Time)**.** *Let x^t denote the probability vector of a random walk after t time steps and let π denote the stationary distribution. The mixing time t_{mix} is defined as follows:*

$$t_{mix}(\varepsilon) := \min\{t : \|\pi - x^t\| \leq \varepsilon\}$$

for $0 \leq \varepsilon < 1$.

Table 1.1 gives examples of these properties for several known graph classes.

1.2 Aims and Outline

In this thesis we consider the effect of different communication patterns on the performance of randomised algorithms for problems relating to the efficient dissem-

Graph Family	Hitting Time	Mixing Time
Cycle	$n^2/2$	$\Theta(n^2)$
2 dimensional grid	$\Theta(n\ logn)$	$\Theta(n)$
d-dimensional grid, d>2	$\Theta(n)$	$\Theta\left(n^{2/d}\right)$
Hypercube	$\Theta(n)$	$\mathcal{O}(\log n \log\log n)$
Complete Graph	$\Theta(n)$	1
ER Random Graphs	$\Theta(n)$	$\mathcal{O}(\log n)$

Table 1.1: Examples of Hitting and Mixing Times based on Graph Family (for any constant $\varepsilon > 0$) [6]

ination of information in large networks.

In Chapter 2 we study a infinite balls into bins allocation process. A fundamental problem in distributed computing is the distribution of requests to a set of uniform servers without a centralized controller. Classically, such problems are modelled as static balls into bins processes, where m balls (tasks) are to be distributed among n bins (servers) where the value of m is fixed. Balls are allocated sequentially. i.e., one after the other. In a seminal work, Azar et al. [12] proposed the GREEDY[d] strategy for $n = m$. Each ball queries the load of d random bins and is allocated to the least loaded of them. Azar et al. showed that $d = 2$ yields an exponential improvement compared to $d = 1$. Berenbrink et al. [21] extended this to $m \gg n$, showing that for $d = 2$ the maximal load difference is independent of m (in contrast to the $d = 1$ case). We propose a new variant of an *infinite* balls-into-bins process. Unlike previous work that assumes that a total system load of n balls at all times, in each round an expected number of λn new balls arrive and are distributed (in parallel) to the bins, and subsequently each non-empty bin deletes one of its balls. This setting models a set of servers processing incoming requests, where clients can query a server's current load but receive no information about parallel requests. We study the GREEDY[d] distribution scheme in this setting and show a strong self-stabilizing property: for *any* arrival rate $\lambda = \lambda(n) < 1$, the system load is time-invariant. Moreover, for *any* (even super-exponential) round t, the maximum system load is (w.h.p.) $\mathcal{O}\left(\frac{1}{1-\lambda}\log\frac{n}{1-\lambda}\right)$ for $d = 1$ and $\mathcal{O}\left(\log\frac{n}{1-\lambda}\right)$ for $d = 2$. Our results show that the so called "power of two choices" carries over to our new infinite

setting. In particular, GREEDY[2] has an exponentially smaller system load for high arrival rates. For example for an arrival rate of $\lambda = 1 - 1/n$ the maximum load under GREEDY[1] is $\mathcal{O}(n \log n)$ (w.h.p) where as under GREEDY[2] the maximum load is $\mathcal{O}(\log n)$ (w.h.p).

Chapter 3 analyses the runtime and space requirements of protocols solving the plurality consensus problem on arbitrary graphs and large range of communication patterns. The plurality consensus problem is defined as follows. 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 neighbours at once) and everything in-between. We propose a framework to design protocols for a multitude of communication patterns. Moreover, we introduce two protocols (BALANCE and SHUFFLE) that solve the plurality consensus problem and are 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. In particular our protocols are able to solve the plurality consensus problem even when the bias between opinions is arbitrarily small.

Chapter 4 considers the performance of the classical PULL algorithm with a restricted communication pattern for the rumour spreading problem. Rumour spreading (broadcast) is a fundamental task in distributed computing. Randomised rumour spreading algorithms disseminate a piece of information (or rumour) from a single source node to all nodes in a graph. Typically two symmetrically defined algorithms are studied. Namely the PUSH and PULL algorithms. The algorithms proceed in synchronous rounds. Under the PUSH algorithm, each node that has the message

chooses a neighbour uniformly at random to send the message to. The PULL algorithm is defined symmetrically. Each node without the message sends a request to a neighbour chosen uniformly at random. Finally the PUSH-PULL algorithm is the combination of both these algorithms. These algorithms have been studied under a range of different assumptions such as different network topologies, graph expansion properties, dynamic graphs, asynchronous execution, and robustness against node failure and noise (See Section 4.1.1). Recent results [39, 55, 69] investigate the discrepancy between PUSH and PULL. The PULL algorithm assumes that a single node is able to inform all nodes it receives requests from in a given round. This might be unrealistic since it immediately implies certain capabilities of the nodes. We study the PULL algorithm with the restriction that just as in the PUSH algorithm, each node can inform at most one new node per round. Our results bound the broadcast time of this algorithm in terms of a certain random walk through its relationship to the PUSH algorithm. In particular, we show that for certain graph classes this restricted algorithm is optimal.

Chapter 2

Infinite, Parallel Balls-into-Bins

In this chapter we study an infinite, parallel balls into bins allocation process. In each round, λn balls (in expectation) are allocated according to the GREEDY[d] distribution scheme in parallel. Subsequently, a ball is removed from each non-empty bin. For $d = 1, 2$ we show that the GREEDY[d] allocation scheme is self stabilising. More we show that the so called "power of two choices" carries over to this new setting. In particular we show that the maximum load (with high probability) under GREEDY[2] is exponentially smaller than under GREEDY[1].

2.1 Introduction

One of the fundamental problems in distributed computing is the distribution of requests, tasks, or data items to a set of uniform servers. In order to simplify this process and to avoid a single point of failure, it is often advisable to use a simple, randomized strategy instead of a complex, centralized controller to allocate the requests to the servers. In the most naïve strategy (*1-choice*), each client sends its request to a server chosen uniformly at random. A more elaborate scheme (*2-choice*) chooses two servers, queries their current loads, and sends the request to a least loaded of them. Both approaches are typically modelled as balls-into-bins processes [2, 12, 13, 21, 61, 86, 94], where requests are represented as balls and servers as bins. While the 2-choice approach leads to considerably better load distributions [12, 21], it loses

some of its power in parallel settings, where requests arrive in parallel and cannot take each other into account [2, 94].

We propose and study a novel infinite batch-based balls-into-bins process to model the client-server scenario. In a round, each server (bin) consumes one of its current tasks (balls). Afterward, expectedly λn tasks arrive and are allocated using a given distribution scheme. The *arrival rate* λ is allowed to be a function of n (e.g., $\lambda = 1 - 1/\operatorname{poly}(n)$). Standard balls-into-bins results imply that, for high arrival rates, with high probability (w.h.p.) in each round there is a bin that receives $\Theta(\log n / \log \log n)$ balls. Most other infinite balls-into-bins-type processes limit the total number of concurrent balls in the system by n [12, 13] and show a fast recovery. Since we do not limit the number of balls, our process can, in principle, result in an arbitrary high system load. In particular, if starting in a high-load situation (e.g., exponentially many balls), we cannot recover in a polynomial number of steps. Instead, we regard the system load as a Markov chain and adapt the following notion of *self-stabilization*: The system is positive recurrent (expected return time to a typical low-load situation is finite), and taking a snapshot of the load situation at an *arbitrary* (even super-exponential large) time step yields (w.h.p.) a time-independent maximum load. Positive recurrence is a standard notion for stability and basically states that the system load is time-invariant. Recall that for irreducible, aperiodic Markov chains it implies the existence of a unique stationary distribution (cf. Section 1.1.4). While this alone does not guarantee a good load in the stationary distribution, together with the snapshot property we can look at an arbitrary time window of polynomial size (even if it is exponentially far away from the start) and give strong load guarantees. In particular, we give the following bounds on the load in addition to showing positive recurrence:

1-choice Process: The maximum load at an arbitrary time is (w.h.p.) bounded by $\mathcal{O}\left(\frac{1}{1-\lambda} \cdot \log \frac{n}{1-\lambda}\right)$. We also provide a lower bound which is asymptotically tight for $\lambda \leq 1 - 1/\operatorname{poly}(n)$. While this implies that already the simple 1-choice process is self-stabilizing, the load properties in a "typical" state are poor: even an arrival

rate of only $\lambda = 1 - 1/n$ yields a superlinear maximum load.

2-choice Process: The maximum load at an arbitrary time is (w.h.p.) bounded by $\mathcal{O}\left(\log \frac{n}{1-\lambda}\right)$. This allows to maintain an exponentially better system load compared to the 1-choice process; for any $\lambda \leq 1 - 1/\operatorname{poly}(n)$ the maximum load remains logarithmic. Note that the resulting processes can be seen as queuing processes.

2.1.1 Related Work

We will continue with an overview of related work. We start with classical results for sequential and finite balls-into-bins processes, go over to parallel settings, and give an overview of infinite and batch-based processes similar to ours. We also briefly mention some results from queuing theory (which is related but studies slightly different quality of service measures and system models).

Sequential Setting There are many strong, well-known results for the classical, sequential balls-into-bins process. In the sequential setting, m balls are thrown one after another and allocated to n bins. For $m = n$, the maximum load of any bin is known to be (w.h.p.) $(1 + \mathcal{O}(1)) \cdot \ln(n)/\ln\ln n$ for the 1-choice process [61, 86] and $\ln\ln(n)/\ln d + \Theta(1)$ for the d-choice process with $d \geq 2$ [12]. If $m \geq n \cdot \ln n$, the maximum load increases to $m/n + \Theta\left(\sqrt{m \cdot \ln(n)/n}\right)$ [86] and $m/n + \ln\ln(n)/\ln d + \Theta(1)$ [21], respectively. In particular, note that the number of balls above the average grows with m for $d = 1$ but is independent of m for $d \geq 2$. This fundamental difference is known as the *power of two choices*. A similar (if slightly weaker) result was shown by Talwar and Wieder [96] using a quite elegant proof technique (which we also employ and generalize for our analysis in Section 2.3). Czumaj and Stemann [37] study adaptive allocation processes where the number of a ball's choices depends on the load of queried bins. The authors subsequently analyze a scenario that allows reallocations.

Berenbrink et al. [26] adapt the threshold protocol from Adler et al. [2] (see below) to a sequential setting and $m \geq n$ bins. Here, ball i randomly chooses bins

until it sees a load smaller than $1 + i/n$. While this is a relatively strong assumption on the balls, this protocol needs only $\mathcal{O}(m)$ choices in total (allocation time) and achieves an almost optimal maximum load of $\lceil m/n \rceil + 1$.

Parallel Setting Several papers (e.g., [2, 94]) investigated parallel settings of multiple-choice games for the case $m = n$. Here, all m balls have to be allocated in parallel, but balls and bins might employ some (limited) communication. Adler et al. [2] consider a trade-off between the maximum load and the number of communication rounds r the balls need to decide for a target bin. Basically, bounds that are close to the classical (sequential) processes can only be achieved if r is close to the maximum load [2]. The authors also give a lower bound on the maximum load if r communication rounds are allowed, and Stemann [94] provides a matching upper bound via a collision-based protocol.

Infinite Processes For infinite processes, the number of balls to be thrown is not fixed. Instead, in each of infinitely many rounds, balls are thrown or reallocated and bins (possibly) delete old balls. Azar et al. [12] consider an infinite, sequential process starting with n balls arbitrarily assigned to n bins. In each round one random ball is reallocated using the d-choice process. For any $t > cn^2 \log\log n$, the maximum load at time t is (w.h.p.) $\ln\ln(n)/\ln d + \mathcal{O}(1)$.

Adler et al. [1] consider a system where in each round $m \leq n/9$ balls are allocated. Each bin has a FIFO queue, and each arriving ball is stored in the queue of two randomly chosen bins. After each round, every non-empty bin deletes its frontmost ball (which automatically removes its copy from the second random bin). It is shown that the expected waiting time is constant and the maximum waiting time is (w.h.p.) $\ln\ln(n)/\ln d + \mathcal{O}(1)$. The restriction $m \leq n/9$ is the major drawback of this process. A further study of this process, based on differential methods and experiments, was conducted by Berenbrink et al. [20]. The balls' arrival times are binomially distributed with parameters n and $\lambda = m/n$. Their results indicate a stable behaviour for $\lambda \leq 0.86$. A similar model was considered by Mitzenmacher

[76], who considers ball arrivals as a Poisson stream of rate λn for $\lambda < 1$. It is shown that the 2-choice process reduces the waiting time exponentially compared to the 1-choice process.

Czumaj [38] presents a framework to study the recovery time of discrete-time dynamic allocation processes. In each round one of n balls is reallocated using the d-choice process. Two models are considered: in the first, the ball to be reallocated is chosen by taking a ball from a random bin. In the second, the ball to be reallocated is chosen by selecting a random ball. From an arbitrary initial assignment, the system is shown to recover to the maximum load from Azar et al. [12] within $\mathcal{O}(n^2 \ln n)$ rounds in the former and $\mathcal{O}(n \ln n)$ rounds in the latter case. Becchetti et al. [13] consider a similar (but parallel) process. In each round one ball is chosen from every non-empty bin and reallocated to a randomly chosen bin (one choice per ball). The authors show that (w.h.p.) starting from an arbitrary configuration, it takes $\mathcal{O}(n)$ rounds to reach a configuration with maximum load $\mathcal{O}(\log n)$. Moreover, if the process starts in a configuration with maximum load $\mathcal{O}(\log n)$, then the maximum load stays in $\mathcal{O}(\log n)$ for poly(n) rounds. An interesting connection to our work is that the analysis of [13] is based on an auxiliary TETRIS-process. This process can be seen a special version of our 1-choice process and is defined as follows: starting from a state with at least $n/4$ empty bins, in each round every non-empty bin deletes one ball. Subsequently, exactly $(3/4)n$ new balls are allocated to the bins (one choice per ball).

Batch-Processes Batch-based processes allocate m balls to n bins in batches of (usually) n balls each, where each batch is allocated in parallel. They lie between (pure) parallel and sequential processes. For $m = \tau \cdot n$, Stemann [94] investigates a scenario with n players each having m/n balls. To allocate a ball, every player independently chooses two bins and allocates copies of the ball to both of them. Every bin has two queues (one for first copies, one for second copies) and processes one ball from each queue per round. When a ball is processed, its copy is removed from the system and the player is allowed to initiate the allocation of the next ball.

If $\tau = \ln n$, all balls are processed in $\mathcal{O}(\ln n)$ rounds and the waiting time is (w.h.p.) $\mathcal{O}(\ln \ln n)$. Berenbrink et al. [19] study the d-choice process in a scenario where m balls are allocated to n bins in batches of size n each. The authors show that the load of every bin is (w.h.p.) $m/n \pm \mathcal{O}(\log n)$. As noted in Lemma 2.3.2, our analysis can be used to derive the same result by easier means.

Queuing Processes Batch arrival processes have also been considered in the context of queuing systems. A key motivation for such models stems from the asynchronous transfer mode (ATM) in telecommunication systems. Tasks arrive in batches, are stored in a FIFO queue and served by a fixed number of servers which remove the tasks from the queue and process them. Several papers [3, 65, 68, 93] consider scenarios where the number of arriving tasks is determined by a finite state Markov chain. Results study steady state properties of the system to determine properties of interest (e.g., waiting times or queue lengths). Sohraby and Zhang [93] use spectral techniques to study a multi-server scenario with an infinite queue. Alfa [3] considers a discrete-time process for n identical servers and tasks with constant service time $s \geq 1$. To ensure a stable system, the arrival rate λ is assumed to be at most n/s and tasks are assigned cyclically, allowing the authors to study an arbitrary server (instead of the complete system). Kamal [65] and Kim et al. [68] study a system with a finite capacity. The tasks which arrive when the buffer is full are lost. The authors study the steady state probability and give empirical results to show the decay of waiting times as n increases.

2.1.2 Model & Preliminaries

We model our load balancing problem as an infinite, parallel balls-into-bins process. Time is divided into discrete, synchronous rounds. There are n bins and n generators, and the initial system is assumed to be empty. At the start of each round, every non-empty bin deletes one ball. Afterward, every generator generates a ball with a probability of $\lambda = \lambda(n) \in [0, 1]$ (the *arrival rate*). This generation scheme allows us

to consider arrival rates that are arbitrarily close to one (such as $1 - 1/\operatorname{poly}(n)$). Generated balls are distributed in the system using a distribution process. In this chapter we analyze two specific distribution processes:

- The 1-choice process (GREEDY[1]) assigns every ball to a random bin.
- The 2-choice process (GREEDY[2]) assigns every ball to a least loaded among two randomly chosen bins.

Notation The random variable $X_i(t)$ denotes the load (number of balls) of the i-th fullest bin at the end of round t. Thus, the load situation (configuration) after round t can be described by the load vector $\boldsymbol{X}(t) = (X_i(t))_{i\in[n]} \in \mathbb{N}^n$. We define $\varnothing(t) \coloneqq \frac{1}{n}\sum_{i=1}^{n} X_i(t)$ as the average load at the end of round t.

Markov Chain Preliminaries Before proceeding with our analysis we first make explicit the Markov chain that we study. For supplementary definitions see Section 1.1.4. We consider the Markov chain on the load vectors. This Markov chain has the Markov property since $\boldsymbol{X}(t)$ depends only on $\boldsymbol{X}(t-1)$ and the random choices during round t. We refer to this Markov chain as $\boldsymbol{X}$. This Markov chains state space includes all vectors with non-increasing entries over $\mathbb{N}^n$. Note that $\boldsymbol{X}$ is time-homogeneous (transition probabilities are time-independent), irreducible (every state is reachable from every other state, and aperiodic (path lengths have no period; in fact, our chain is lazy). Recall that such a Markov chain is positive recurrent (or ergodic) if the probability to return to the start state is 1 and the expected return time is finite. In particular, this implies the existence of a unique stationary distribution. See [72] for an excellent introduction into Markov chains and the involved terminology.

Positive recurrence is a standard formalization of the intuitive concept of stability. A state of a Markov chain is positive recurrent if the expected return time the state is finite. Moreover, if all states in an irreducible Markov chain are positive recurrent then the Markov chain is said to be positive recurrent. In order to show

that the infinite processes studied in the chapter are positive recurrent we will use the following results. The first result is due Fayolle et al. [49] and states two conditions that must hold for a Markov Chain to be positive recurrent. The theorem considers a suitably chosen potential function that is defined over the states of the Markov chain. Informally Condition (a) of the theorem state that in most cases the potential function is decreasing linearly over a given number of steps ($\beta(x)$). For a finite set of states C we do not require that the potential function is decreasing during $\beta(x)$ steps but instead that it remains finite.

Theorem 2.1.1 (Fayolle et al. [49, Theorem 2.2.4])**.** *A time-homogeneous irreducible aperiodic Markov chain ζ with a countable state space Ω is positive recurrent if and only if there exists a positive function $\phi(x), x \in \Omega$, a number $\eta > 0$, a positive integer-valued function $\beta(x), x \in \Omega$, and a finite set $C \subseteq \Omega$ such that the following inequalities hold:*

(a) $\mathbb{E}\left[\phi(\zeta(t+\beta(x))) - \phi(x) \mid \zeta(t) = x\right] \leq -\eta\beta(x),\ x \notin C$

(b) $\mathbb{E}\left[\phi(\zeta(t+\beta(x))) \mid \zeta(t) = x\right] < \infty,\ x \in C$

2.2 1-Choice Process

In this section we present two main results for the 1-choice process. Theorem 2.2.4 states the stability of the system under the 1-choice process for an arbitrary λ, using the standard notion of positive recurrence (cf. Section 2.1). In particular, this implies the existence of a stationary distribution for the 1-choice process. Theorem 2.2.2 strengthens this by giving a high probability bound on the maximum load for an *arbitrary* round $t \in \mathbb{N}$. Together, both results imply that the 1-choice process is self-stabilizing, i.e., the system is positive recurrent, and taking a snapshot of the load situation at an arbitrary time step yields (w.h.p.) a time-independent maximum load.

Theorem 2.2.2 (Maximum Load)**.** *Let $\lambda = \lambda(n) < 1$. Fix an arbitrary round t of the 1-choice process. The maximum load of any bin is (w.h.p.) bounded by $\mathcal{O}\left(\frac{1}{1-\lambda} \cdot \log \frac{n}{1-\lambda}\right)$.*

Theorem 2.2.4 (Stability)**.** *Let $\lambda = \lambda(n) < 1$. The Markov chain $\boldsymbol{X}$ of the 1-choice process is positive recurrent.*

Theorem 2.2.2 implies that for high arrival rates such as $\lambda(n) = 1 - 1/n$ the maximal load is $\mathcal{O}(n \log n)$. Theorem 2.2.6 shows that this dependence is unavoidable i.e., the bound given in Theorem 2.2.2 is tight for large values of λ.

Theorem 2.2.6 (Lower Bound)**.** *Let $\lambda = \lambda(n) \geq 0.75$ and consider step $t \coloneqq \lambda \log(n)/(8(1-\lambda)^2)$. With probability $1 - o(1)$ there is a bin i in step t with load $\Omega\left(\frac{1}{1-\lambda} \cdot \log n\right)$.*

We first prove a bound on the maximum load (Theorem 2.2.2), afterward we prove stability of the system (Theorem 2.2.4), and finally we prove the lower bound (Theorem 2.2.6).

2.2.1 Maximum Load

In this section we prove Theorem 2.2.2 that bounds the maximum load w.h.p. To show this result we first bound the load of a fixed bin i at time t using Theorem 2.2.1 (Hajek) and, subsequently, use this result along with a union bound to bound the maximum load over all bins.

Theorem 2.2.1 (Simplified version of Hajek [63, Theorem 2.3])**.** *Let $(Y(t))_{t\geq 0}$ be a sequence of random variables on a probability space $(\Omega, \mathcal{F}, P)$ with respect to the filtration $(\mathcal{F}(t))_{t\geq 0}$. Assume the following two conditions hold:*

(i) (Majorization) There exists a random variable Z and a constant $\lambda' > 0$, such that $\mathbb{E}[e^{\lambda' Z}] \leq D$ for some finite D, and $(|Y(t+1) - Y(t)| \big| \mathcal{F}(t)) \prec Z$ for all $t \geq 0$; and

(ii) (Negative Bias) There exist $a, \varepsilon_0 > 0$, such for all t we have

$$\mathbb{E}[Y(t+1) - Y(t) \mid \mathcal{F}(t), Y(t) > a] \leq -\varepsilon_0.$$

Let $\eta = \min\{\lambda', \varepsilon_0 \cdot \lambda'^2/(2D), 1/(2\varepsilon_0)\}$. Then, for all b and t we have

$$\Pr(Y(t) \geq b \mid \mathcal{F}(0)) \leq e^{\eta(Y(0)-b)} + \frac{2D}{\varepsilon_0 \cdot \eta} \cdot e^{\eta(a-b)}.$$

Proof. The statement of the theorem provided in [63] requires besides (i) and (ii) to choose constants η, and ρ such that $0 < \rho \leq \lambda'$, $\eta < \varepsilon_0/c$ and $\rho = 1 - \varepsilon_0 \cdot \eta + c\eta^2$ where $c = \frac{\mathbb{E}\left[e^{\lambda' Z}\right] - (1+\lambda'\mathbb{E}[Z])}{\lambda'^2} = \sum_{k=2}^{\infty} \frac{\lambda'^{k-2}}{k!} \mathbb{E}\left[Z^k\right]$. With these requirements it then holds that for all b and t

$$\Pr(Y(t) \geq b \mid \mathcal{F}(0)) \leq \rho^t e^{\eta(Y(0)-b)} + \frac{1-\rho^t}{1-\rho} \cdot D \cdot e^{\eta(a-b)}. \tag{2.1}$$

In the following we bound (2.1) by setting $\eta = \min\{\lambda', \varepsilon_0 \cdot \lambda'^2/(2D), 1/(2\varepsilon_0)\}$. The following upper and lower bound on ρ follow.

- $\rho = 1 - \varepsilon_0 \cdot \eta + c\eta^2 \leq 1 - \varepsilon_0 \cdot \eta + \varepsilon_0 \cdot \eta \cdot c \cdot \lambda'^2/(2D) \leq 1 - \varepsilon_0 \cdot \eta + \varepsilon_0 \cdot \eta/2 = 1 - \varepsilon_0 \cdot \eta/2$, where we used $c \leq D/\lambda'^2$.
- $\rho = 1 - \varepsilon_0 \cdot \eta + c\eta^2 \geq 1 - \varepsilon_0/(2\varepsilon_0) \geq 0$.

We derive, from (2.1) using that for any $t \geq 0$ we have $0 \leq \rho^t \leq 1$

$$\begin{aligned}
\Pr(Y(t) \geq b \mid \mathcal{F}(0)) &\leq \rho^t e^{\eta(Y(0)-b)} + \frac{1-\rho^t}{1-\rho} \cdot D \cdot e^{\eta(a-b)} \\
&\leq e^{\eta(Y(0)-b)} + \frac{1}{1-\rho} \cdot D \cdot e^{\eta(a-b)} \\
&\leq e^{\eta(Y(0)-b)} + \frac{2D}{\varepsilon_0 \cdot \eta} \cdot e^{\eta(a-b)},
\end{aligned} \tag{2.2}$$

since $\frac{1}{(1-\rho)} \leq \frac{2}{\varepsilon_0 \cdot \eta}$. This yields the claim. □

In order to apply Theorem 2.2.1 (Hajek), we have to prove that the maximum load difference of bin i between two rounds is exponentially bounded (Condition (i) Majorization) and that, given a load high enough, the total system decreases in expectation (Condition (ii): Negative Bias).

Theorem 2.2.2 (Maximum Load)**.** *Let $\lambda = \lambda(n) < 1$. Fix an arbitrary round t of the 1-choice process. The maximum load of any bin is (w.h.p.) bounded by $\mathcal{O}\left(\frac{1}{1-\lambda} \cdot \log \frac{n}{1-\lambda}\right)$.*

Proof. In the following, we prove Theorem 2.2.2 using a (slightly simplified) drift theorem (Theorem 2.2.1 cf Hajek [63]). Remember that, as mentioned in Section 2.1.2, our process is a Markov chain. As such we only need to condition only on the previous state (instead of the full filtration from Theorem 2.2.1 (Hajek)).

We start with the Condition (i) (Majorization).

Condition (i) (Majorization):

The load difference for a bin i between round t and round $t+1$ is defined as $|X_i(t+1) - X_i(t)|$. In a given round the load difference is bounded by $\max(1, B_i(t)) \leq 1 + B_i(t)$, where $B_i(t)$ is the number of tokens resource i receives during round $t+1$. The statement holds since in a round any bin either deletes a ball without receiving additional balls, receives additional balls with (or without) deleting a ball, or remains empty. In each of these cases the change in the load of bin i is 1,$B_i(t)$, and 0 respectively. This follows from the definition of our process.

From this observation we obtain the following stochastic domination

$$(|X_i(t+1) - X_i(t)| \,|\, \boldsymbol{X}(t)) \prec 1 + B_i(t).$$

Note that $B_i(t)$ is binomially distributed random variable with parameters n and λ/n. This follows from the definition of our model. Each generator creates a ball with probability λ and is allocated to a given bin with probability $1/n$. It follows that each ball is allocated to bin i with probability $\lambda \cdot 1/n$.

Using standard inequalities we bound

$$\Pr\left(B_i(t) = k\right) = \binom{n}{k}\left(\frac{\lambda}{n}\right)^k\left(\frac{1-\lambda}{n}\right)^{n-k}$$

Since $\frac{1-\lambda}{n} < 1$

$$\leq \binom{n}{k}\cdot\left(\frac{\lambda}{n}\right)^k \overset{eq.\ 1.5}{\leq} \left(\frac{e\cdot n}{k}\right)^k\cdot\left(\frac{1}{n}\right)^k = \frac{e^k}{k^k}$$

Where the second inequality follows from Equation 1.5 and $\lambda < 1$.

Let $Z := B_i(t) + 1$, we now calculate $\mathbb{E}\left[e^{\lambda' Z}\right]$.

$$\begin{aligned}\mathbb{E}\left[e^{B_i(t)+1}\right] = e\cdot\sum_{k=0}^{n} e^k\cdot\frac{e^k}{k^k} &\leq e\cdot\sum_{k=0}^{\lceil e^3-1\rceil}\frac{e^{2k}}{k^k} + e\cdot\sum_{k=e^3}^{\infty}\frac{e^{2k}}{k^k}\\ &\leq \Theta(1) + \sum_{k=1}^{\infty} e^{-k} = \Theta(1).\end{aligned} \tag{2.3}$$

This shows that the Majorization condition from Theorem 2.2.1 holds (with $\lambda' = 1$ and $D = \Theta(1)$).

Condition (ii) (Negative Bias)

To see that the Negative Bias condition is given, note that if bin i has non-zero load, it is guaranteed to delete one ball and receives in expectation $n\cdot\lambda/n = \lambda$ balls. We get

$$\mathbb{E}\left[X_i(t+1) - X_i(t) \mid X_i(t) > 0\right] \leq \lambda - 1 < 0$$

establishing the Negative Bias condition (with $\varepsilon_0 = 1-\lambda$).

We finally can apply Theorem 2.2.1 with

$$\eta \coloneqq \min\left\{1, (1-\lambda)/(2D), 1/(2-2\lambda)\right\} = (1-\lambda)/(2D)$$

and for $b \geq 1$ obtain the following

$$\begin{aligned}\Pr\left(X_i(t) \geq b\right) &\leq e^{-b\cdot\eta} + \frac{2D}{\eta\cdot(1-\lambda)} \cdot e^{\eta\cdot(1-b)} \leq \frac{2\cdot(2D)^2}{(1-\lambda)^2} \cdot e^{\frac{(1-\lambda)\cdot(1-b)}{2D}} \\ &\leq \frac{2\cdot(2D)^2}{(1-\lambda)^2} \cdot e^{\frac{(1-\lambda)\cdot(1-b)}{(4D)^2}} \leq \frac{(4D)^2}{(1-\lambda)^2} \cdot e^{\frac{-b\cdot(1-\lambda)}{(4D)^2}} \\ &\leq \frac{c}{(1-\lambda)^2} \cdot e^{-\frac{b\cdot(1-\lambda)}{c}}\end{aligned} \tag{2.4}$$

where $c \geq (4D)^2$ denotes a constant.

Applying a union bound to all n bins and choosing $b \coloneqq \frac{c}{1-\lambda} \cdot \ln\left(\frac{c\cdot n^{h+1}}{(1-\lambda)^2}\right)$, where $h > 0$ is a constant, yields that $\Pr\left(\max_{i\in[n]} X_i(t) \geq b\right) \leq n^{-h}$.

The theorem's statement now follows from

$$\begin{aligned}b &= \frac{c}{1-\lambda} \cdot \ln\left(\frac{c\cdot n^{h+1}}{(1-\lambda)^2}\right) \\ &\leq \frac{c\cdot(h+1)+1}{1-\lambda} \cdot \ln\left(\frac{n}{1-\lambda}\right) \\ &= \mathcal{O}\left(\frac{1}{1-\lambda} \cdot \ln\left(\frac{n}{1-\lambda}\right)\right)\end{aligned} \tag{2.5}$$

□

2.2.2 Stability

In this section we show the stability of the 1-choice process. To do this show that the Markov chain for the 1-choice process is positive recurrent (cf. Section 2.1). We prove this using Theorem 2.1.1 (cf. Fayolle et al. [49]). Before showing the requisite conditions to be able to apply Theorem 2.1.1 we show an upper bound on the expected load of a bin under the 1-choice process.

Lemma 2.2.3. *Let $\lambda = \lambda(n) < 1$. Fix an arbitrary round t of the 1-choice process and a bin i. Let $X_i(t)$ denote the load of bin i in round t. There is a constant $c > 0$ such that,*

$$\mathbb{E}\left[X_i(t)\right] \leq \frac{6c}{1-\lambda} \cdot \ln\left(\frac{n}{1-\lambda}\right)$$

Proof. Let $\gamma \coloneqq \frac{c}{1-\lambda} \cdot \ln\left(\frac{n}{(1-\lambda)^2}\right)$ where c is the constant from the proof of Theorem 2.2.2. Considering time windows of γ rounds each and using Equation (2.4), we calculate

$$\begin{aligned}
\mathbb{E}\left[X_i(t)\right] &= \sum_{b=1}^{\infty} b \cdot \Pr\left(X_i(t) = b\right) = \sum_{k=1}^{\infty} \sum_{b=k\cdot\gamma}^{(k+1)\gamma} b \cdot \Pr\left(X_i(t) = b\right) \\
&\leq \sum_{b=1}^{\gamma} b \cdot \Pr\left(X_i(t) = b\right) + \sum_{k=1}^{\infty} \sum_{b=k\cdot\gamma}^{(k+1)\gamma} b \cdot \Pr\left(X_i(t) = b\right) \\
&\leq \gamma + \sum_{k=1}^{\infty} (k+1) \cdot \gamma \cdot \Pr\left(X_i(t) \geq k \cdot \gamma\right) \\
&\leq \gamma + \sum_{k=1}^{\infty} (k+1) \cdot \gamma \cdot e^{-k} \\
&\leq 3\gamma \leq \frac{6c}{1-\lambda} \cdot \ln\left(\frac{n}{(1-\lambda)}\right)
\end{aligned} \tag{2.6}$$

This finishes the proof. □

The following theorem states that the Markov chain for the 1-choice process is self-stabilizing. i.e., the Markov chain is positive recurrent.

Theorem 2.2.4 (Stability)**.** *Let $\lambda = \lambda(n) < 1$. The Markov chain $\boldsymbol{X}$ of the 1-choice process is positive recurrent.*

Proof. We prove Theorem 2.2.4 by applying a result from Fayolle et al. [49] (cf. Theorem 2.1.1). We define the parameters of Theorem 2.1.1. Note that $\boldsymbol{X}$ is a time-homogeneous irreducible Markov chain with a countable state space. For a configuration $\boldsymbol{x}$, we define the potential $\Psi(\boldsymbol{x}) \coloneqq \sum_{i=1}^{n} x_i$ as the total system load of configuration $\boldsymbol{x}$.

In the following, let $\Delta \coloneqq \frac{12cn^2}{(1-\lambda)^3}$ where c is the constant in Lemma 2.2.3. Define the (finite) set $C \coloneqq \{\, \boldsymbol{x} \mid \Psi(\boldsymbol{x}) \leq \Delta \cdot n \,\}$ of all configurations where the total system load is not "too" high. To prove positive recurrence, it remains to show that Condition (a) (expected potential drop if in a high-load configuration) and Condition (b) (finite potential) of Theorem 2.1.1 hold.

Let us start with Condition (a).

Condition (a)

Fix a round t. Condition (a) of theorem 2.1.1 considers states not in the set C. Let $\boldsymbol{X}(t) = \boldsymbol{x} \notin C$. By definition of C, we have $\Psi(\boldsymbol{x}) > \Delta \cdot n$. It follows using a pigeonhole argument that there is at least one bin i with load $x_i \geq \Psi(\boldsymbol{x})/n > \Delta$.

By the definition of our model, bin i deletes exactly one ball during each of the next Δ rounds. On the other hand, bin i receives in expectation $\Delta \cdot \lambda n \cdot \frac{1}{n} = \lambda\Delta$ balls during the next Δ rounds. Combining the above observations we obtain the following,

$$\mathbb{E}\left[X_i(t+\Delta) - x_i \mid \boldsymbol{X}(t) = \boldsymbol{x}\right] = \lambda\Delta - \Delta = -(1-\lambda)\cdot\Delta$$

For any bin $j \neq i$, we assume pessimistically that no ball is deleted. Note that the expected load increase of each of these bins can be majorized by the load increase in an empty system running for Δ rounds. Thus, we can use Lemma 2.2.3 to bound the expected load increase in each of these bins by $\frac{6c}{1-\lambda} \cdot \ln\left(\frac{n}{1-\lambda}\right) \leq \frac{6 \cdot cn}{(1-\lambda)^2} = \frac{(1-\lambda)\Delta}{2n}$. We get

$$\begin{aligned}
\mathbb{E}\left[\Psi(\boldsymbol{X}(t+\Delta)) \mid \boldsymbol{X}(t) = \boldsymbol{x}\right] &\leq -(1-\lambda)\cdot\Delta + (n-1)\cdot\frac{(1-\lambda)\Delta}{2n} \\
&\leq -(1-\lambda)\cdot\Delta + \frac{(1-\lambda)\Delta}{2} \\
&= -\frac{(1-\lambda)}{2}\cdot\Delta
\end{aligned} \tag{2.7}$$

This proves Condition (a) of Theorem 2.1.1 with $\beta(x) = \Delta$ and $\eta = \frac{(1-\lambda)}{2}$ and $\phi(x) = \Psi(x)$.

Condition (b)

Assume $\boldsymbol{x} = \boldsymbol{X}(t) \in C$. We bound the system load after Δ rounds trivially by

$$\mathbb{E}\left[\Psi(\boldsymbol{X}(t+\Delta)) \mid \boldsymbol{X}(t) = \boldsymbol{x}\right] \leq \Psi(\boldsymbol{x}) + \Delta \cdot n \leq \Delta \cdot n + \Delta \cdot n < \infty \tag{2.8}$$

(note that the finiteness in Theorem 2.1.1 is with respect to time, not n). This finishes the proof. □

2.2.3 Lower Bound on Maximum Load

In this section we show a lower bound for the maximum load. This lower bound shows that we are unable to avoid the dependence on $\frac{1}{1-\lambda}$. This implies that for high arrival rates e.g., $\lambda = 1 - \frac{1}{n}$, the bound given in in the previous section (Theorem 2.2.2) is tight.

To show a lower bound for the maximum load, we will use the following result by Raab and Steger [86, Theorem 1] which lower-bounds the maximum number of balls a bin receives when m balls are allocated into n bins. We first state Raab and Steger [86, Theorem 1] before presenting our lower bound.

Theorem 2.2.5 (Raab and Steger [86, Theorem 1])**.** *Let M be the random variable that counts the maximum number of balls in any bin, if we throw m balls independently and uniformly at random into n bins. Then $\Pr(M > k_\alpha) = o(1)$ if $\alpha > 1$ and $\Pr(M > k_\alpha) = 1 - o(1)$ if $0 < \alpha < 1$, where*

$$k_\alpha = \begin{cases} \frac{\log n}{\log \frac{n \log n}{m}} \left(1 + \alpha \frac{\log\log \frac{n \log n}{m}}{\log \frac{n \log n}{m}}\right) & if\ \frac{n}{\text{polylog}(n)} \leq m \ll n \log n \\ (d_c - 1 + \alpha) \log n & if\ m = c \cdot n \log n \text{ for some constant } c \\ \frac{m}{n} + \alpha \sqrt{2 \frac{m}{n} \log n} & if\ n \log n \ll m \leq n \,\text{polylog}(n) \\ \frac{m}{n} + \sqrt{2 \frac{m}{n} \log n \left(1 - \frac{1}{\alpha} \frac{\log\log n}{2 \log n}\right)} & if\ m \gg n (\log n)^3, \end{cases}$$

where d_c is largest solution of $1 + x(\log c - \log x + 1) - c = 0$. We have $d_1 = e$ and $d_{1.00001} = 2.7183$.

We will now prove the following theorem that lower bounds the maximum load.

Theorem 2.2.6. *Let $\lambda = \lambda(n) \geq \frac{3}{4}$ and consider step $t := \lambda \log(n) / (8(1-\lambda)^2)$. With probability $1 - o(1)$ there is a bin i in step t with load $\Omega\left(\frac{1}{1-\lambda} \cdot \log n\right)$.*

Proof. The idea of the proof is as follows:

Assume that we start at an empty system and apply Theorem 2.2.5 (cf.[86, Theorem 1]) to $m = \lambda t n$ many balls. The theorem states that one of the bins is likely to get more than λt many balls, which allows us to show that the load of this bin is large, even if the bin was able to delete a ball during each of the t observed time steps.

We begin by bounding the number of balls that is allocated during the first t' rounds. Let $M(t')$ be the number of balls allocated during the first t' rounds. By the definition of our process it follows that $\mathbb{E}[M(t')] = \lambda t' n$. By choosing t' to be an appropriate number of rounds we are able to apply Chernoff bounds to show that w.h.p at least $(1-\epsilon) \cdot \mathbb{E}[M(t')]$ balls are generated for small ϵ.

Set $t \coloneqq \lambda \log(n) / (8(1-\lambda)^2)$ and $\epsilon \coloneqq (1-\lambda)/\lambda$.

Using the Chernoff bound in Lemma 1.1.9 we can show that w.h.p. $M(t) \geq (1-\epsilon)\mathbb{E}[M(t)] = (1-\epsilon) \cdot t \cdot \lambda n$. Note that to apply the Chernoff bound in Lemma 1.1.9 we choose $0 < \epsilon < 1$. For our choice of ϵ this implies that,

$$\frac{1-\lambda}{\lambda} < 1 \implies \lambda > \frac{1}{2}$$

Applying the Chernoff bound we obtain,

$$\begin{aligned}
\Pr(M(t) \leq (1-\varepsilon) \cdot \mathbb{E}[M(t)]) &\leq \exp\left(-\frac{\varepsilon^2}{2} \cdot \mathbb{E}[M(t)]\right) \\
&= \exp\left(-\frac{\varepsilon^2}{2} \cdot \frac{\lambda \log n}{8(1-\lambda)^2} \cdot \lambda n\right) \\
&= \exp\left(-\frac{1}{16} \cdot n \log n\right) = n^{-\frac{n}{16}} \leq n^{-2}
\end{aligned}$$

where the last inequality holds for $n \geq 32$.

It follows that $M(t) \geq (1-\epsilon) \cdot \mathbb{E}[M(t)]$ with high probability for our choice of ϵ and $n \geq 32$.

Using that $M(t) \geq (1-\epsilon) \cdot \mathbb{E}[M(t)]$, we use Theorem 2.2.5 to lower bound the maximum load with probability $1 - o(1)$. Let $Y_{max}(t')$ be the maximum number

of balls allocated to a bin during t' rounds. We now apply Theorem 2.2.5 to lower bound the maximum number of balls that a bin receives. Since our lower bound on the number of balls generated depends on our choice of t,λ and ϵ we apply either case (3) or (4) of Theorem 2.2.5 to obtain the following lower bound,

$$\begin{aligned} Y_{max}(t) &\geq \frac{M(t)}{n} + \sqrt{2\frac{M(t)}{n} \cdot \log(n)} \cdot \min\left\{\alpha, \sqrt{1 - \frac{\log\log n}{2\alpha \log n}}\right\} \\ &= \frac{M(t)}{n} + \sqrt{2\frac{M(t)}{n} \cdot \log(n)} \cdot \alpha \\ &\geq (1-\epsilon)\lambda t + \sqrt{2(1-\epsilon)\lambda t \cdot \log(n)} \cdot \alpha \end{aligned}$$

where the first inequality holds for a suitable choice of α. Using this lower bound on the maximum number of balls allocated to any bin during the first t rounds, we now show that even if this bin is able to delete a ball in each of these t rounds then the maximum load is still lower bounded by $\Omega\left(\frac{\lambda \log n}{1-\lambda}\right)$.

Let $X_{\max}(t)$ denote the load of the bin with the maximum load. In the following let $\alpha \coloneqq \sqrt{\frac{9}{16}}$ and $\lambda \geq \frac{3}{4}$

$$\begin{aligned} X_{\max}(t) &\geq Y_{\max}(t) - t \\ &\geq (1-\epsilon)\lambda t + \sqrt{2(1-\epsilon)\lambda t \cdot \log(n)} \cdot \alpha - t \\ &= (1-\epsilon)\lambda t + \sqrt{(1-\epsilon)\frac{18}{16} \cdot \lambda t \cdot \log(n)} - t \\ &= (1-\epsilon)\lambda t + \sqrt{(1-\epsilon)\frac{9}{64}} \cdot \frac{\lambda \log n}{(1-\lambda)} - t \\ &= \sqrt{(1-\epsilon)\frac{9}{64}} \cdot \frac{\lambda \log n}{(1-\lambda)} - 2(1-\lambda)t \\ &= \left(\sqrt{\left(1 - \frac{1-\lambda}{\lambda}\right)\frac{9}{64}} - \frac{1}{4}\right) \cdot \frac{\lambda \log n}{(1-\lambda)} \\ &\geq \left(\sqrt{\frac{2}{3} \cdot \frac{9}{64}} - \frac{1}{4}\right) \cdot \frac{\lambda \log n}{(1-\lambda)} \\ &= \Omega\left(\frac{\lambda \log n}{1-\lambda}\right) \end{aligned}$$

The claim follows since $1 > \lambda \geq \frac{3}{4}$ □

2.3 The 2-Choice Process

We continue with the study of the 2-choice process. Here, new balls are distributed according to GREEDY[2] (cf. description in Section 2.1.2). Our main results are the following theorems. These theorems are analogous to the corresponding theorems that were showing in the previous section for the 1-choice process.

Theorem 2.3.18 (Maximum Load)**.** *Let $\lambda = \lambda(n) \in [1/4, 1)$. Fix an arbitrary round t of the 2-choice process. The maximum load of any bin is (w.h.p.) bounded by $\mathcal{O}\left(\log \frac{n}{1-\lambda}\right)$.*

Theorem 2.3.20 (Stability)**.** *Let $\lambda = \lambda(n) \in [1/4, 1)$. The Markov chain $\boldsymbol{X}$ of the 2-choice process is positive recurrent.*

Theorem 2.3.18 implies a much better behaved system than for the 1-choice process (See Theorem 2.2.2). In particular, it allows for an exponentially higher arrival rate. For $\lambda(n) = 1 - 1/\operatorname{poly}(n)$ the 2-choice process maintains a maximal load of $\mathcal{O}(\log n)$. In contrast, for the same arrival rate the 1-choice process results in a system with maximal load $\Omega(\operatorname{poly}(n))$.

Results in the sequential setting have been obtain through majorising GREEDY[2] by GREEDY[1](e.g.,[12, 21]). However it is not clear that such an argument holds in our setting. We therefore follow the approach used by Peres et al. [83] and Talwar and Wieder [96]. To prove these results, we combine three different potential functions:

For a configuration $\boldsymbol{x}$ with average load $\varnothing$ and for a suitable constant $\alpha < 1$ (to

be fixed later), we define

$$\begin{aligned}\Phi(\boldsymbol{x}) &\coloneqq \sum_{i\in[n]} e^{\alpha\cdot(x_i-\varnothing)} + \sum_{i\in[n]} e^{\alpha\cdot(\varnothing-x_i)} \\ \Psi(\boldsymbol{x}) &\coloneqq \sum_{i\in[n]} x_i \\ \Gamma(\boldsymbol{x}) &\coloneqq \Phi(\boldsymbol{x}) + \tfrac{n}{1-\lambda}\cdot\Psi(\boldsymbol{x}).\end{aligned} \tag{2.9}$$

Our analysis of the 2-choice process relies to a large part on a good bound on the *smoothness*. We define the smoothness of an allocation to be the maximum load difference between any two bins. The potential Φ measures the *smoothness* of a configuration and is used to prove Lemma 2.3.4 (Section 2.3.1). The proof is based on the observation that whenever the load of a bin is far from the average load then in expectation the potential function Φ decreases.

The potential Ψ measures the *total load* of a configuration and is used, in combination with our results on the smoothness, to prove Theorem 2.3.18 (Maximum Load). For example, if the total load of a configuration is bounded by $\mathcal{O}(n\ln(n))$, it follows that the average load is $\mathcal{O}(\ln n)$. Using our bound on the smoothness of a configuration (Lemma 2.3.4) we are able to bound the maximum load.

Finally, the potential Γ entangles the smoothness and total load, allowing us to prove Theorem 2.3.20 (Stability). The proof is based on the fact that whenever Γ is large (i.e., the configuration is not smooth or it has a large total load), it decreases in expectation. This allows us to apply Theorem 2.1.1 to show that the Markov chain for the 2-choice process is positive recurrent.

2.3.1 Smoothness

In this section we consider the smoothness of the allocation in an arbitrary round t. The smoothness of an allocation is defined as the maximum difference between any two bins. Our main result in this section (Lemma 2.3.4) states the smoothness of a configuration in an arbitrary round is bounded by $\mathcal{O}(\ln(n))$ (w.h.p). We begin by proving this lemma. To do so we will state some of the required results without

proof. The rest of this section will then be dedicated to proving these results.

Recall that the potential function Φ measures the smoothness of an allocation. Lemma 2.3.1 bounds the expected change in Φ due to a single round. We state this result without proof. The proof of the Lemma follows the lines of Peres et al. [83] and Talwar and Wieder [96], who used the same potential function to analyze variants of the sequential d-choice process without deletions. While the basic idea of showing a relative drop when the potential is high combined with a bounded absolute increase in the general case is the same, our analysis turns out to be much more involved. In particular, not only do we have to handle deletions and allocating balls in batches, but the size of each batch is also a random variable.

Lemma 2.3.1. *Consider an arbitrary round $t+1$ of the 2-choice process and the constants ε (from Proposition 2.3.10) and $\alpha \leq \min(\ln(10/9), \varepsilon/8)$. For $\lambda \in [1/4, 1]$ we have*

$$\mathbb{E}\left[\Phi(\boldsymbol{X}(t+1)) \mid \boldsymbol{X}(t)\right] \leq \left(1 - \frac{\varepsilon\alpha\lambda}{4}\right) \cdot \Phi(\boldsymbol{X}(t)) + \varepsilon^{-8} \cdot \mathcal{O}\left(n\right). \tag{2.10}$$

Using Lemma 2.3.1 we are able to prove an upper bound on the expected value of Φ in an arbitrary round t. This bound will be used to obtain an upper bound the value of Φ in an arbitrary round t (w.h.p).

Lemma 2.3.2. *Let $\lambda \in [1/4, 1]$. Fix an arbitrary round t of the 2-choice process. There is a constant $\varepsilon > 0$ such that*

$$\mathbb{E}\left[\Phi(\boldsymbol{X}(t))\right] \leq \frac{n}{\varepsilon} \tag{2.11}$$

Proof. Using Lemma 2.3.1 we obtain that for all rounds $t \geq 0$,

$$\mathbb{E}\left[\Phi(\boldsymbol{X}(t+1)) \mid \boldsymbol{X}(t)\right] \leq \gamma \cdot \Phi(\boldsymbol{X}(t)) + c$$

where $\gamma < 1$ and $c > 0$ are values given by Lemma 2.3.1.

Taking the expected value on both sides yields,

$$\mathbb{E}\left[\Phi(\boldsymbol{X}(t+1))\right] \leq \gamma \cdot \mathbb{E}\left[\Phi(\boldsymbol{X}(t))\right] + c$$

$\mathbb{E}\left[\Phi(\boldsymbol{X}(t))\right] \leq \frac{c}{1-\gamma}$ solves this recursion.

Using the values from Lemma 2.3.1 for γ and c (substituting ε' for ε) we get

$$\mathbb{E}\left[\Phi(\boldsymbol{X}(t))\right] \leq \frac{4\varepsilon'^{-8}}{\varepsilon'\alpha\lambda} \cdot \mathcal{O}\left(n\right)$$

The lemma's statement follows for the constant $\varepsilon = \mathcal{O}\left(\varepsilon'^{-9}/(\alpha\lambda)\right)$. □

Before proving Lemma 2.3.4 we first make the following observation. Observation 2.3.3 states that for any configuration $\boldsymbol{x}$ and value $b \geq 0$, the inequality $\Phi(\boldsymbol{x}) \leq e^{\alpha \cdot b}$ implies that $\max_i |x_i - \varnothing| \leq b$. That is, the load difference of any bin to the average is at most b and, thus, the load difference between any two bins is at most $2b$.

Observation 2.3.3. *Let $b \geq 0$ and consider a configuration $\boldsymbol{x}$ with average load $\varnothing$. If $\Phi(\boldsymbol{x}) \leq e^{\alpha \cdot b}$, then $|x_i - \varnothing| \leq b$ for all $i \in [n]$. In particular, $\max_i(x_i) - \min_i(x_i) \leq 2b$.*

Proof. The statement can be proved by contraposition. Assume for some bin i, $|x_i - \varnothing| > b$ it follows that there is a term in $\Phi(\boldsymbol{x})$ such that either $e^{\alpha(X_i - \varnothing)} > e^{\alpha \cdot b}$ or $e^{\alpha(\varnothing - X_i)} > e^{\alpha \cdot b}$. Since all terms in $\Phi(\boldsymbol{x})$ are greater than zero, it follows that $\Phi(\boldsymbol{x}) > e^{\alpha \cdot b}$. This shows the contrapositive and the statement follows. □

With this observation we now show the main result in this section.

Lemma 2.3.4 (Smoothness)**.** *Let $\lambda = \lambda(n) \in [1/4, 1]$. Fix an arbitrary round t of the 2-choice process. The load difference of all bins is (w.h.p.) bounded by $\mathcal{O}\left(\ln n\right)$.*

Proof. Using Markov's inequality and Lemma 2.3.2 we obtain

$$\Pr\left(\Phi(\boldsymbol{X}(t)) \geq \frac{n^2}{\varepsilon^2}\right) \leq \frac{1}{n^2}$$

It follows that w.h.p.

$$\Phi(\boldsymbol{X}(t)) \leq \frac{n^2}{\varepsilon^2}.$$

Rewriting we obtain,

$$e^{\alpha b} = \frac{n^2}{\varepsilon^2} \implies b = \frac{2}{\alpha} \ln\left(\frac{n}{\varepsilon}\right).$$

Using this value of b and Observation 2.3.3 it follows that,

$$\max_i(X_i) - \min_i(X_i) \leq \frac{4}{\alpha} \ln\left(\frac{n}{\varepsilon}\right) = \mathcal{O}\left(\ln(n)\right).$$

This gives the desired bound on the smoothness. □

This proves the main result in this section (Lemma 2.3.4) that bounds the smoothness of a configuration in an arbitrary round t w.h.p. The proof of the lemma uses Lemma 2.3.1 that bounds the potential change in a single round. Since we stated this result without proof it remains to show Lemma 2.3.1.

Bounding the one step potential change

The rest of this section is dedicated to proving Lemma 2.3.1 that bounds the expected change of Φ due to a single round. The proof of Lemma 2.3.1 is by case analysis. The case analysis follows the same line of argument used by Peres et al. [83] and Talwar and Wieder [96]. To show the case analysis, we show lemmas that bound the change in Φ due to a single round in different situations. e.g., reasonable balanced load allocations or very unbalanced load allocations.

We begin with some additional notation and auxiliary definitions that will be used in this section. The value $\nu(t)$ denotes the fraction of non-empty bins after round t and $\eta(t) \coloneqq 1 - \nu(t)$ the fraction of empty bins after round t. It will be useful to define $Z_i(t) \coloneqq \min\big(1, X_i(t)\big)$ and $\eta_i(t) \coloneqq Z_i(t) - \nu(t)$ (which equals $\eta(t)$ if i is a non-empty bin and $-\nu(t)$ otherwise).

The potential function Φ can be rewritten as follows,

$$\Phi(\boldsymbol{x}) \coloneqq \Phi_+(\boldsymbol{x}) + \Phi_-(\boldsymbol{x})$$

Where $\Phi_+(\boldsymbol{x})$ denotes the *upper potential* of $\boldsymbol{x}$ and $\Phi_-(\boldsymbol{x})$ denotes the *lower potential* of $\boldsymbol{x}$. For a fixed bin i, we use $\Phi_{i,+}(\boldsymbol{x}) \coloneqq e^{\alpha\cdot(x_i-\varnothing)}$ and $\Phi_{i,-}(\boldsymbol{x}) \coloneqq e^{\alpha\cdot(\varnothing-x_i)}$ to denote i's contribution to the upper and lower potential, respectively. It follows that

$$\begin{aligned}\Phi_+(\boldsymbol{x}) &\coloneqq \sum_{i=1}^{n} \Phi_{i,+}(\boldsymbol{x}) \\ \Phi_-(\boldsymbol{x}) &\coloneqq \sum_{i=1}^{n} \Phi_{i,-}(\boldsymbol{x})\end{aligned} \tag{2.12}$$

We define the potential change due to a fixed bin i during a single round $t+1$ for the upper and lower potential respectively as follows

$$\begin{aligned}\Delta_{i,+}(t+1) &\coloneqq \Phi_{i,+}(\boldsymbol{X}(t+1)) - \Phi_{i,+}(\boldsymbol{X}(t)) \\ \Delta_{i,-}(t+1) &\coloneqq \Phi_{i,-}(\boldsymbol{X}(t+1)) - \Phi_{i,-}(\boldsymbol{X}(t))\end{aligned} \tag{2.13}$$

Similarly let $\Delta_+(t+1) \coloneqq \sum_{i=1}^{n} \Delta_{i,+}$ and $\Delta_-(t+1) \coloneqq \sum_{i=1}^{n} \Delta_{i,-}$ denote the change in the upper and lower potential during round $t+1$.

We define the following variables to ease the presentation of the subsequent results. Let p_i denote the probability that a ball thrown with Greedy[2] falls into the i-th fullest bin.

$$p_i \coloneqq \left(\frac{i}{n}\right)^2 - \left(\frac{i-1}{n}\right)^2 = \frac{2i-1}{n^2} \tag{2.14}$$

We also define

$$\begin{aligned}\hat{\alpha} &\coloneqq e^{\alpha} - 1 \\ \check{\alpha} &\coloneqq 1 - e^{-\alpha}\end{aligned} \tag{2.15}$$

where $\hat{\alpha} \in (\alpha, \alpha+\alpha^2)$ and $\check{\alpha} \in (\alpha-\alpha^2, \alpha)$ for $\alpha \in (0, 1.7)$. This follows from the Taylor approximation $e^x \leq 1 + x + x^2$, which holds for any $x \in (-\infty, 1.7]$.

Finally, let

$$\begin{aligned}\hat{\delta}_i &\coloneqq \lambda n \cdot \left(\frac{1}{n} \cdot \ell_\perp(\alpha) - p_i \cdot \frac{\hat{\alpha}}{\alpha}\right) \\ \check{\delta}_i &\coloneqq \lambda n \cdot \left(\frac{1}{n} \cdot \ell_\top(\alpha) - p_i \cdot \frac{\check{\alpha}}{\alpha}\right)\end{aligned} \tag{2.16}$$

where $\ell_\perp(\alpha) \coloneqq 1 - \alpha/n < 1 < \ell_\top(\alpha) \coloneqq 1 + \alpha/n$. We will omit the parameter α when it is clear from context. Note that $\ell_\top(\alpha)$, $\ell_\perp(\alpha)$, $\hat{\alpha}/\alpha$, and $\check{\alpha}/\alpha$ are all close to 1.

We start with two useful identities regarding the potential change $\Delta_{i,+}(t+1)$ (and $\Delta_{i,-}(t+1)$) due to a fixed bin i during round $t+1$ that will used in our subsequent analysis. These identities state the change in the upper and lower potential for a bin i when both the number of balls allocated to the bin and the total number of balls allocated in a round are known.

Observation 2.3.5. *Fix a bin i, let K denote the number of balls that are placed during round $t+1$ and let $k \leq K$ be the number of these balls that fall into bin i. Then*

(a) $\Delta_{i,+}(t+1) = \Phi_{i,+}(\boldsymbol{X}(t)) \cdot \left(e^{\alpha \cdot (k - \eta_i(t) - K/n)} - 1\right)$ *and*

(b) $\Delta_{i,-}(t+1) = \Phi_{i,-}(\boldsymbol{X}(t)) \cdot \left(e^{-\alpha \cdot (k - \eta_i(t) - K/n)} - 1\right)$.

Proof. Recall that $Z_i(t) \coloneqq \min\big(1, X_i(t)\big)$ is an indicator value which equals 1 if and only if the i-th bin is non-empty in configuration $\boldsymbol{X}$. Bin i deletes exactly Z_i balls and receives exactly k balls, such that $X_i(t+1) - X_i(t) = -Z_i + k$. Similarly, we have that the change of the average load is $\varnothing(t+1) - \varnothing(t) = -\nu + K/n$. With the identity $\eta_i = Z_i - \nu$, this yields

$$\begin{aligned}\Delta_{i,+}(t) &= e^{\alpha \cdot \left(X_i' - \varnothing'\right)} - e^{\alpha \cdot \left(X_i - \varnothing\right)} \\ &= e^{\alpha \cdot \left(X_i - \varnothing\right)} \cdot \left(e^{\alpha \cdot \left(-Z_i + k + \nu - K/n\right)} - 1\right) \\ &= \Phi_{i,+} \cdot \left(e^{\alpha \cdot (k - \eta_i - K/n)} - 1\right)\end{aligned} \tag{2.17}$$

proving the first statement. The second statement follows similarly. □

With these definitions, we now show Lemma 2.3.6 that bounds the expected change in the upper and lower potential due to a single round for a single bin i.

Lemma 2.3.6. *Consider a bin i after round t and a constant $\alpha \leq 1$.*

(a) For the expected change of i's upper potential during round $t+1$ we have

$$\frac{\mathbb{E}\left[\Delta_{i,+}(t+1) \mid \boldsymbol{X}(t)\right]}{\Phi_{i,+}(\boldsymbol{X}(t))} \leq -\alpha \cdot \left(\eta_i + \hat{\delta}_i\right) + \alpha^2 \cdot \left(\eta_i + \hat{\delta}_i\right)^2. \tag{2.18}$$

(b) For the expected change of i's lower potential during round $t+1$ we have

$$\frac{\mathbb{E}\left[\Delta_{i,-}(t+1) \mid \boldsymbol{X}(t)\right]}{\Phi_{i,-}(\boldsymbol{X}(t))} \leq \alpha \cdot \left(\eta_i + \check{\delta}_i\right) + \alpha^2 \cdot \left(\eta_i + \check{\delta}_i\right)^2. \tag{2.19}$$

Proof. Consider an arbitrary fixed round t. Let $\mathcal{A}_K(t)$ be the event that K balls are allocated in round t and $B_i(t)$ be the number of balls are allocated to bin i. We omit the time parameter since the round t is fixed. Observation 2.3.5 derives an expression for $\Delta_{i,-}$ given that k balls are allocated to bin i. Using this we obtain,

$$\mathbb{E}\left[\Delta_{i,+}(t) \mid \boldsymbol{X}, \mathcal{A}_K\right] = \sum_{k=0}^{K} \Phi_{i,+}(\boldsymbol{X}(t)) \cdot \left(e^{\alpha \cdot (k - \eta_i(t) - K/n)} - 1\right) \cdot \Pr\left(B_i = k\right) \tag{2.20}$$

That is, the expected change of the upper potential given that K balls are allocated during the round.

We use the law of total expectation to calculate $\mathbb{E}\left[\Delta_{i,+}(t) \mid \boldsymbol{X}\right]$. Since $0 \leq K \leq n$, we can use the events $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$ to partition the outcome space. By the law of total expectation we obtain

$$\mathbb{E}\left[\Delta_{i,+}(t) \mid \boldsymbol{X}\right] = \sum_{K=0}^{n} \mathbb{E}\left[\Delta_{i,+}(t) \mid \boldsymbol{X}, \mathcal{A}_K\right] \cdot \Pr\left(\mathcal{A}_K\right) \tag{2.21}$$

Combining Equation 2.20 and Equation 2.21 by substituting for $\mathbb{E}\left[\Delta_{i,+}(t) \mid \boldsymbol{X}, \mathcal{A}_K\right]$ in Equation 2.21,

$$\mathbb{E}\left[\Delta_{i,+}(t+1) \mid \boldsymbol{X}\right] = \sum_{K=0}^{n}\sum_{k=0}^{K} \Phi_{i,+}(\boldsymbol{X}(t)) \cdot \left(e^{\alpha\cdot(k-\eta_i(t)-K/n)} - 1\right) \cdot \Pr\left(B_i = k\right) \cdot \Pr\left(\mathcal{A}_K\right)$$

Dividing both sides by $\Phi_{i,+}$,

$$\begin{aligned}
&\mathbb{E}\left[\Delta_{i,+}(t+1) \mid \boldsymbol{X}\right] / \Phi_{i,+} \\
&= \sum_{K=0}^{n}\sum_{k=0}^{K} \left(e^{\alpha\cdot(k-\eta_i(t)-K/n)} - 1\right) \cdot \Pr\left(B_i = k\right) \cdot \Pr\left(\mathcal{A}_K\right) \\
&= \sum_{K=0}^{n}\sum_{k=0}^{K} \binom{n}{K}\binom{K}{k} (p_i\lambda)^k \cdot \left((1-p_i)\lambda\right)^{K-k} \cdot (1-\lambda)^{n-K} \cdot \left(e^{\alpha\cdot(k-\eta_i-K/n)} - 1\right) \\
&= \sum_{K=0}^{n} \binom{n}{K} (1-\lambda)^{n-K}\lambda^K \sum_{k=0}^{K} \binom{K}{k} \cdot p_i^k \cdot (1-p_i)^{K-k} \cdot \left(e^{\alpha\cdot(k-\eta_i-K/n)} - 1\right)
\end{aligned}$$

Expanding the final term and since $e^{\alpha\cdot(k-\eta_i-K/n)} = e^{-\alpha(\eta_i+K/n)} \cdot e^{\alpha\cdot k}$, and using $\sum_{k=0}^{K}\binom{K}{k}p^k(1-p)^k = 1$,

$$= \sum_{K=0}^{n} \binom{n}{K} (1-\lambda)^{n-K}\lambda^K \cdot \left(e^{-\alpha(\eta_i+K/n)} \sum_{k=0}^{K} \binom{K}{k} (e^\alpha \cdot p_i)^k (1-p_i)^{K-k} - 1\right)$$

Applying the binomial theorem (Definition 1.1.2) and $e^\alpha := \hat{\alpha} + 1$

$$= \sum_{K=0}^{n} \binom{n}{K} (1-\lambda)^{n-K}\lambda^K \cdot \left(e^{-\alpha(\eta_i+K/n)} \cdot (1+\hat{\alpha}\cdot p_i)^K - 1\right),$$

Applying the binomial theorem again,

$$\begin{aligned}
&= e^{-\alpha\eta_i} \cdot \left(1 - \lambda + \lambda e^{-\alpha/n} \cdot (1 + \hat{\alpha}\cdot p_i)\right)^n - 1 \\
&\leq e^{-\alpha\eta_i} \cdot \left(1 - \lambda(1 - e^{-\alpha/n}) + \lambda\cdot\hat{\alpha}\cdot p_i\right)^n - 1
\end{aligned}$$

Using the Taylor approximation, $e^x \leq 1 + x + x^2$ which holds for any $x \in (-\infty, 1.7]$

$$\leq e^{-\alpha\eta_i} \cdot \left(1 - \frac{\lambda \cdot \alpha}{n} \cdot (1 - \alpha/n) + \lambda \cdot \hat{\alpha} \cdot p_i\right)^n - 1$$
$$= e^{-\alpha\eta_i} \cdot \left(1 - \frac{\alpha}{n}\left(\lambda(1 - \alpha/n) - \lambda \cdot n \cdot \frac{\hat{\alpha}}{\alpha} \cdot p_i\right)\right)^n - 1$$

Using the definition of $\hat{\delta}_i$ that $\ell_\perp := 1 - \frac{\alpha}{n}$,

$$= e^{-\alpha\eta_i} \cdot \left(1 - \frac{\alpha}{n}\left(\lambda n\left(1/n \cdot \ell_\perp - \frac{\hat{\alpha}}{\alpha} \cdot p_i\right)\right)\right)^n - 1$$

Using the property of the exponential function (Lemma 1.1.5),

$$\leq e^{-\alpha\eta_i} \cdot \left(1 - \frac{\alpha}{n} \cdot \hat{\delta}_i\right)^n - 1$$
$$\leq e^{-\alpha \cdot \left(\eta_i + \hat{\delta}_i\right)} - 1.$$

Now, the claim follows by another application of the Taylor approximation. The second statement follows similarly. □

Lemma 2.3.6 will be used to derive the bounds on the potential drop in different situations that will used in the case analysis.

We provide two auxiliary claims (Proposition 2.3.7 and Proposition 2.3.8). These propositions bound key quantities and will be used when deriving different bounds on the potential drop.

Proposition 2.3.7. *Consider a bin i and the values $\hat{\delta}_i$ and $\check{\delta}_i$ as defined in Equation 2.16. If $\alpha \leq \ln(10/9)$, then $\max(|\hat{\delta}_i|, |\check{\delta}_i|) \leq \frac{5}{4}\lambda$.*

Proof. Remember that $\hat{\delta}_i \coloneqq \lambda n \cdot (1/n \cdot \ell_\perp - p_i \cdot \hat{\alpha}/\alpha)$ and $\check{\delta}_i \coloneqq \lambda n \cdot (1/n \cdot \ell_\top - p_i \cdot \check{\alpha}/\alpha)$, where $\ell_\perp = 1 - \alpha/n < 1 < 1 + \alpha/n = \ell_\top$.

To show the statement of the lemma we prove the following inequalities,

$$-\frac{5}{4}n \overset{(a)}{\leq} \check{\delta}_i \overset{(b)}{\leq} \frac{5}{4}n \qquad \text{and,} \qquad -\frac{5}{4}n \overset{(c)}{\leq} \hat{\delta}_i \overset{(d)}{\leq} \frac{5}{4}n \tag{2.22}$$

Since p_i are non-decreasing we consider the two extremes of $i = 1$ and $i = n$ and show that the claim holds for both.

For inequalities (a) and (c) we consider $p_n \leq \frac{2}{n}$. First consider inequality (a):

$$\hat{\delta}_i \geq \lambda n \left(\frac{1}{n} \cdot \ell_\perp - \frac{2}{n} \cdot (\alpha + 1) \right) \geq \lambda \left(1 - \frac{1}{4} \right) \tag{2.23}$$

where the first inequality uses that $\hat{\alpha} \in (\alpha, \alpha + \alpha^2)$. For inequality (c),

$$\check{\delta}_i \geq \lambda n \left(\frac{1}{n} \cdot \ell_\top - \frac{2}{n} \right) \geq -\lambda \tag{2.24}$$

where the first inequality follows using that $\check{\alpha}(\alpha - \alpha^2, \alpha)$.

For inequalities (b) and (d) we consider $p_1 = \frac{1}{n^2}$. For inequality (b),

$$\hat{\delta}_i \leq \lambda n \left(\frac{1}{n} \cdot \ell_\perp - \frac{1}{n^2} \right) \leq \lambda \tag{2.25}$$

Finally for inequality (d) we obtain the following using the definitions of $\ell_\top$ and $\check{\alpha}$

$$\check{\delta}_i = \lambda n \left(\frac{1}{n} \cdot \ell_\top - p_i \cdot \frac{\check{\alpha}}{\alpha} \right) \leq \lambda \left(1 + \frac{\alpha}{n} - \frac{1}{n\alpha}(1 - e^{-\alpha}) \right) \tag{2.26}$$

The claim holds where,

$$\begin{aligned} 1 + \frac{\alpha}{n} - \frac{1}{n\alpha}(1 - e^{-\alpha}) \leq \frac{5}{4} &\implies \alpha - \frac{1}{\alpha}(1 - e^{-\alpha}) \leq \frac{1}{4}n \\ &\implies \alpha - \frac{1}{\alpha} + \frac{1}{\alpha} \cdot e^{-\alpha} \leq \frac{1}{4}n \end{aligned}$$

For $\alpha \leq \ln(10/9)$ the LHS can be upper bounded by α and the claim follows.

□

Proposition 2.3.8. *Consider a round t and a constant $1 \geq \alpha \geq 0$. The following inequalities hold:*

(a) $\sum_{i \in [n]} \alpha\eta_i(\alpha\eta_i - 1) \cdot \Phi_{i,+}(\boldsymbol{X}(t)) \leq \alpha^2\eta\nu \cdot \min\big(n, \Phi_+(\boldsymbol{X}(t))\big)$.

(b) $\sum_{i \in [n]} \alpha\eta_i(\alpha\eta_i + 1) \cdot \Phi_{i,-}(\boldsymbol{X}(t)) \leq \alpha^2\eta\nu \cdot \Phi_-(\boldsymbol{X}(t))$.

Proof. For the first statement, we calculate $\sum_{i\in[n]} \alpha\eta_i(\alpha\eta_i - 1)\cdot\Phi_{i,+}(\boldsymbol{X}(t))$

$$\begin{aligned}
&= \sum_{i\le\nu n} \alpha\eta_i(\alpha\eta_i - 1)\cdot\Phi_{i,+}(\boldsymbol{X}(t)) + \sum_{i>\nu n} \alpha\eta_i(\alpha\eta_i - 1)\cdot\Phi_{i,+}(\boldsymbol{X}(t))\\
&= \alpha\eta(\alpha\eta - 1)\cdot\sum_{i\le\nu n}\Phi_{i,+}(\boldsymbol{X}(t)) + \alpha\nu(1+\alpha\nu)\cdot\sum_{i>\nu n}\Phi_{i,+}(\boldsymbol{X}(t))\\
&\le \alpha\eta(\alpha\eta - 1)\cdot\nu\cdot\Phi_{+}(\boldsymbol{X}(t)) + \alpha\nu(1+\alpha\nu)\cdot\eta\cdot\min\big(n,\Phi_{+}(\boldsymbol{X}(t))\big)\\
&\le \alpha^2\eta\nu\cdot\min\big(n,\Phi_{+}(\boldsymbol{X}(t))\big),
\end{aligned} \tag{2.27}$$

where the first inequality uses that $\Phi_{i,+}(\boldsymbol{X}(t))$ is non-increasing in i and that $\Phi_{i,+}(\boldsymbol{X}(t)) \le 1$ for all $i > \nu n$. The claim's second statement follows by a similar calculation, using that $\Phi_{i,-}(\boldsymbol{X}(t))$ is non-decreasing in i (note that we cannot apply the same trick as above to get $\min\big(n,\Phi_{-}(\boldsymbol{X}(t))\big)$ instead of $\Phi_{-}(\boldsymbol{X}(t))$). □

With these tools we are able to derive bounds on the potential change in different situations. These bounds are used in the case analysis for Lemma 2.3.1. We start with a relative bound on the upper and lower potential change.

Lemma 2.3.9. *Consider a round t and a constant $\alpha \le \ln(10/9) < 1/8$). Let $R \in \{+,-\}$ and $\lambda \in [1/4,1]$. For the expected upper and lower potential drop during round $t+1$ we have*

$$\mathbb{E}\left[\Delta_R(t+1) \mid \boldsymbol{X}(t)\right] < 2\alpha\lambda\cdot\Phi_R(\boldsymbol{X}(t)). \tag{2.28}$$

Proof. We prove the statement for $R = +$. The case $R = -$ follows similarly. Using Lemma 2.3.6 and summing up over all $i \in [n]$ we get

$$\begin{aligned}
\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] &\le \sum_{i\in[n]}\Big(-\alpha\cdot(\eta_i+\hat{\delta}_i) + \alpha^2\cdot(\eta_i+\hat{\delta}_i)^2\Big)\cdot\Phi_{i,+}\\
&= \sum_{i\in[n]}\Big(\eta_i\alpha(\eta_i\alpha - 1) + \alpha^2\cdot(2\eta_i\hat{\delta}_i + \hat{\delta}_i^2) - \alpha\cdot\hat{\delta}_i\Big)\cdot\Phi_{i,+}\\
&\le \sum_{i\in[n]}\left(\eta_i\alpha(\eta_i\alpha - 1) + 5\alpha^2\lambda + \frac{5}{4}\alpha\lambda\right)\cdot\Phi_{i,+}.
\end{aligned} \tag{2.29}$$

Here, the last inequality uses $\lambda \le 1$ and $|\hat{\delta}_i| \le \frac{5}{4}\lambda$ (Proposition 2.3.7). We now apply

Proposition 2.3.8, $\nu\eta \leq 1/4 \leq \lambda$, and $\alpha < 1/8$ to get

$$\mathbb{E}\left[\Delta_+(t) \mid \boldsymbol{X}\right] \leq \left(\alpha^2\lambda + 5\alpha^2\lambda + \frac{5}{4}\alpha\lambda\right) \cdot \Phi_+ < 2\alpha\lambda \cdot \Phi_+. \tag{2.30}$$

□

The following proposition is used to derive bounds on the potential change when the system is in a reasonably balanced configuration.

Proposition 2.3.10. *There is a constant $\varepsilon > 0$ such that*

$$\sum_{i \leq \frac{3}{4}n} p_i \cdot \Phi_{i,+} \leq (1 - 2\varepsilon) \cdot \frac{\Phi_+}{n} \tag{2.31}$$

and

$$\sum_{i \in [n]} p_i \cdot \Phi_{i,-} \geq (1 + 2\varepsilon) \cdot \frac{\Phi_- - \sum_{i \leq \frac{n}{4}} \Phi_{i,-}}{n} \tag{2.32}$$

Proof. For Equation (2.31), note that $\Phi_{i,+}$ and p_i for $i = 1, \ldots, n$ are non-increasing in i and that by definition $\Phi_+ = \sum_{i=1}^{n} \Phi_{i,+}$. The LHS of Eq. (2.31) is maximized where all $\Phi_{i,+} = \frac{4\Phi_+}{3n}$. Using the following observation from Talwar and Wieder [95, Appendix A] there is a constant $\varepsilon' > 0$ such that

$$\begin{aligned} \sum_{i \geq 3n/4} p_i \geq \frac{1}{4} + \varepsilon' \implies \sum_{i < 3n/4} p_i &\leq 1 - \frac{1}{4} - \varepsilon' = \frac{3}{4} - \varepsilon' \\ \implies \sum_{i \leq 3n/4} p_i &\leq 1 - \frac{1}{4} - \varepsilon' = \frac{3}{4} - \varepsilon'' \end{aligned} \tag{2.33}$$

where $\varepsilon'' = \varepsilon' - o(1)$. The result follows by

$$\sum_{i \leq \frac{3}{4}n} p_i \cdot \Phi_{i,+} \leq \left(\frac{3}{4} - \varepsilon''\right) \frac{4\Phi_+}{3n} = \left(1 - \frac{4\varepsilon''}{3n}\right) \cdot \Phi_+ \leq (1 - 2\varepsilon) \cdot \frac{\Phi_+}{n} \tag{2.34}$$

where the last inequality holds for a suitable choice of $\varepsilon > 0$.

Equation (2.32) follows similarly using the observation from Talwar and Wieder [95, Appendix A] that there exists a constant $\varepsilon' > 0$ such that $\sum_{i \leq n/4} p_i \leq \frac{1}{4} - \varepsilon'$. □

The next two lemmas derive bounds that are used to bound the upper/lower potential change in reasonably balanced configurations.

Lemma 2.3.11. *Consider a round t and the constants ε (from Proposition 2.3.10) and $\alpha \leq \min(\ln(10/9), \varepsilon/8)$. Let $\lambda \in [1/4, 1]$ and assume $X_{\frac{3}{4}n}(t) \leq \varnothing(t)$. For the expected upper potential drop during round $t+1$ we have*

$$\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] \leq -\varepsilon\alpha\lambda \cdot \Phi_+(\boldsymbol{X}(t)) + 2\alpha\lambda n. \tag{2.35}$$

Proof. To calculate the expected upper potential change, we use Lemma 2.3.6 and sum up over all $i \in [n]$ (using similar inequalities as in the proof of Lemma 2.3.9 and the definition of $\hat{\delta}_i$):

$$\begin{aligned}\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] &\leq 6\alpha^2\lambda \cdot \Phi_+ - \sum_{i\in[n]} \alpha \cdot \hat{\delta}_i \cdot \Phi_{i,+} \\ &= \left(6\alpha^2\lambda - \alpha\lambda \cdot \ell_\perp\right) \cdot \Phi_+ + \hat{\alpha}\lambda n \sum_{i\in[n]} p_i \cdot \Phi_{i,+}.\end{aligned} \tag{2.36}$$

We now use that $\Phi_{i,+} = e^{\alpha\cdot(X_i - \varnothing)} \leq 1$ for all $i > \frac{3}{4}n$ (by our assumption on $X_{\frac{3}{4}n}$). This yields

$$\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] \leq \left(6\alpha^2\lambda - \alpha\lambda \cdot \ell_\perp\right) \cdot \Phi_+ + \hat{\alpha}\lambda n \sum_{i\leq\frac{3}{4}n} p_i \cdot \Phi_{i,+} + 2\alpha\lambda n. \tag{2.37}$$

Finally, we apply Proposition 2.3.10 and the definition of $\ell_\perp$ and $\hat{\alpha}$ to get

$$\begin{aligned}\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] &\leq \left(6\alpha^2\lambda - \alpha\lambda \cdot \ell_\perp + (1-2\varepsilon) \cdot \hat{\alpha}\lambda\right) \cdot \Phi_+ + 2\alpha\lambda n \\ &\leq \left(8\alpha^2\lambda - 2\varepsilon \cdot \alpha\lambda\right) \cdot \Phi_+ + 2\alpha\lambda n.\end{aligned} \tag{2.38}$$

Using $\alpha \leq \varepsilon/8$ yields the desired result. □

Lemma 2.3.12. *Consider a round t and the constants ε (from Proposition 2.3.10) and $\alpha \leq \min(\ln(10/9), \varepsilon/8)$. Let $\lambda \in [1/4, 1]$ and assume $X_{\frac{n}{4}}(t) \geq \varnothing(t)$. For the expected lower potential drop during round t we have*

$$\mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}(t)\right] \leq -\varepsilon\alpha\lambda \cdot \Phi_-(\boldsymbol{X}(t)) + \frac{\alpha\lambda n}{2}. \tag{2.39}$$

Proof. To calculate the expected lower potential change, we use Lemma 2.3.6 and sum up over all $i \in [n]$ (as in the proof of Lemma 2.3.11):

$$\begin{aligned}\mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}\right] &\leq 6\alpha^2\lambda \cdot \Phi_- + \sum_{i\in[n]} \alpha \cdot \check{\delta}_i \cdot \Phi_{i,-} \\ &= \left(6\alpha^2\lambda + \alpha\lambda \cdot \ell_\top\right) \cdot \Phi_- - \check{\alpha}\lambda n \sum_{i\in[n]} p_i \cdot \Phi_{i,-}.\end{aligned} \tag{2.40}$$

We now use that $\Phi_{i,-} = e^{\alpha\cdot(\varnothing - X_i)} \leq 1$ for all $i \leq \frac{n}{4}$ (by our assumption on $X_{\frac{n}{4}}$) and apply Proposition 2.3.10 to get

$$\begin{aligned}\mathbb{E}\left[\Delta_-(t) \mid \boldsymbol{X}\right] &\leq \left(6\alpha^2\lambda + \alpha\lambda \cdot \ell_\top\right) \cdot \Phi_- - (1+2\varepsilon) \cdot \check{\alpha}\lambda n \cdot \frac{\Phi_- - \frac{n}{4}}{n} \\ &= \left(6\alpha^2\lambda + \alpha\lambda \cdot \ell_\top - (1+2\varepsilon) \cdot \check{\alpha}\lambda\right) \cdot \Phi_- + (1+2\varepsilon) \cdot \frac{\check{\alpha}\lambda n}{4} \\ &\leq \left(8\alpha^2\lambda - 2\varepsilon \cdot \alpha\lambda\right) \cdot \Phi_- + \frac{\alpha\lambda n}{2},\end{aligned} \tag{2.41}$$

where the last inequality used the definitions of $\ell_\top$, $\check{\alpha}$, as well as $\check{\alpha} > \alpha - \alpha^2$. Using $\alpha \leq \varepsilon/8$ yields the desired result. □

The following two lemmas bound the potential drop in configurations with many balls far below the average to the right (Lemma 2.3.13) and with many balls far above the average to the left (Lemma 2.3.14).

Lemma 2.3.13. *Consider a round t and constants $\alpha \leq 1/46 (< \ln(10/9))$ and $\varepsilon \leq 1/3$. Let $\lambda \in [1/4, 1]$ and assume $X_{\frac{3}{4}n}(t) \geq \varnothing(t)$ and $\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] \geq -\frac{\varepsilon\alpha\lambda}{4} \cdot \Phi_+(\boldsymbol{X}(t))$. Then $\Phi_+(\boldsymbol{X}(t)) \leq \frac{\varepsilon}{4} \cdot \Phi_-(\boldsymbol{X}(t))$ or $\Phi(\boldsymbol{X}(t)) = \varepsilon^{-8} \cdot \mathcal{O}(n)$.*

Proof. Let

$$L \coloneqq \sum_{i\in[n]} \max(X_i - \varnothing, 0) = \sum_{i\in[n]} \max(\varnothing - X_i, 0)$$

be the "excess load" above and below the average.

We begin with the following observation. Since all terms of Φ_- are positive, it follows that $\Phi_- = \sum_{i=1}^{n} \Phi_{i,-} \geq \sum_{i=3n/4}^{n} \Phi_{i,-}$. Using Jensen's inequality, the assumption that $X_{\frac{3}{4}n} \geq \varnothing$ implies $\Phi_- \geq \frac{n}{4} \cdot \exp(\frac{\alpha L}{n/4})$.

On the other hand, we can use the assumption $\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] \geq -\frac{\varepsilon\alpha\lambda}{4} \cdot \Phi_+$ to show an upper bound on Φ_+.

To this end, we use Lemma 2.3.6 and sum up over all $i \in [n]$ (as in the proof of Lemma 2.3.11):

$$\begin{aligned}\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] &\leq 6\alpha^2\lambda \cdot \Phi_+ - \sum_{i\in[n]} \alpha \cdot \hat{\delta}_i \cdot \Phi_{i,+} \\ &= 6\alpha^2\lambda \cdot \Phi_+ - \sum_{i\leq\frac{n}{3}} \alpha \cdot \hat{\delta}_i \cdot \Phi_{i,+} - \sum_{i>\frac{n}{3}} \alpha \cdot \hat{\delta}_i \cdot \Phi_{i,+}.\end{aligned} \tag{2.42}$$

For $i \leq n/3$ we have $p_i = \frac{2i-1}{n^2} \leq \frac{2}{3n}$ and, using the definition of $\ell_\perp$ and $\hat{\alpha}$, $\hat{\delta}_i = \lambda n \cdot \left(1/n \cdot \ell_\perp - p_i \cdot \hat{\alpha}/\alpha\right) \geq (1-5\alpha)\lambda/3$. Setting $\Phi_{\leq n/3,+} \coloneqq \sum_{i\leq n/3} \Phi_{i,+}$ and $\Phi_{>n/3,+} \coloneqq \sum_{i>n/3} \Phi_{i,+}$, together with Proposition 2.3.7 this yields

$$\begin{aligned}&\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] \\ &\leq 6\alpha^2\lambda \cdot \Phi_+ - \frac{\alpha(1-5\alpha)\lambda}{3} \cdot \Phi_{\leq n/3,+} + \frac{5}{4}\alpha\lambda \cdot \Phi_{>n/3,+} \\ &= \left(6\alpha^2\lambda - \frac{\alpha(1-5\alpha)\lambda}{3}\right) \cdot \Phi_+ + \left(\frac{5}{4}\alpha\lambda + \frac{\alpha(1-5\alpha)\lambda}{3}\right) \cdot \Phi_{>n/3,+} \\ &\leq -\frac{\varepsilon\alpha\lambda}{2} \cdot \Phi_+ + 2\alpha\lambda \cdot \Phi_{>n/3,+},\end{aligned} \tag{2.43}$$

where the last inequality uses $\alpha \leq 1/46 \leq \frac{1}{23} - \frac{3}{46}\varepsilon$. With this, the assumption $\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}\right] \geq -\frac{\varepsilon\alpha\lambda}{4} \cdot \Phi_+$ implies $\Phi_+ \leq \frac{8}{\varepsilon} \cdot \Phi_{>n/3,+} \leq \frac{8}{\varepsilon} \cdot \frac{2n}{3} e^{\frac{\alpha L}{n/3}} = \frac{16n}{3\varepsilon} e^{\frac{3\alpha L}{n}}$ (the last inequality uses that none of the $2n/3$ remaining bins can have a load higher than $L/(n/3)$). To finish the proof, assume $\Phi_+ > \frac{\varepsilon}{4} \cdot \Phi_-$ (otherwise the lemma holds). Combining this with the upper bound on Φ_+ and with the lower bound on Φ_-, we

get

$$\frac{16n}{3\varepsilon}e^{\frac{3\alpha L}{n}} \geq \Phi_+ > \frac{\varepsilon}{4}\cdot\Phi_- \geq \frac{\varepsilon n}{16}\cdot e^{\frac{4\alpha L}{n}}. \tag{2.44}$$

Thus, the excess load can be bounded by $L < \frac{n}{\alpha}\cdot\ln\left(\frac{256}{3\varepsilon^2}\right)$. Now, the lemma's statement follows from $\Phi = \Phi_+ + \Phi_- < \frac{5}{\varepsilon}\cdot\Phi_+ \leq \frac{80n}{3\varepsilon^2}e^{\frac{3\alpha L}{n}} = \varepsilon^{-8}\cdot\mathcal{O}(n)$. □

Lemma 2.3.14. *Consider a round t and constants $\alpha \leq 1/32 (< \ln(10/9))$ and $\varepsilon \leq 1$. Let $\lambda \in [1/4, 1]$ and assume $X_{\frac{n}{4}}(t) \leq \varnothing(t)$ and $\mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}(t)\right] \geq -\frac{\varepsilon\alpha\lambda}{4}\cdot\Phi_-(\boldsymbol{X}(t))$. Then $\Phi_-(\boldsymbol{X}(t)) \leq \frac{\varepsilon}{4}\cdot\Phi_+(\boldsymbol{X}(t))$ or $\Phi(\boldsymbol{X}(t)) = \varepsilon^{-8}\cdot\mathcal{O}(n)$.*

Proof. Let $L \coloneqq \sum_{i\in[n]}\max(X_i - \varnothing, 0) = \sum_{i\in[n]}\max(\varnothing - X_i, 0)$ be the "excess load" above and below the average.

Using a similar observation to the one in the proof of the previous lemma, the assumption $X_{\frac{n}{4}} \leq \varnothing$ implies $\Phi_+ \geq \frac{n}{4}\cdot e^{\frac{\alpha L}{n/4}}$ using Jensen's inequality.

On the other hand, we can use the assumption $\mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}\right] \geq -\frac{\varepsilon\alpha\lambda}{4}\cdot\Phi_-$ to show an upper bound on Φ_-. To this end, we use Lemma 2.3.6 and sum up over all $i \in [n]$ (as in the proof of Lemma 2.3.12):

$$\begin{aligned}\mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}\right] &\leq 6\alpha^2\lambda\cdot\Phi_- + \sum_{i\in[n]}\alpha\cdot\check{\delta}_i\cdot\Phi_{i,-}\\ &= 6\alpha^2\lambda\cdot\Phi_- + \sum_{i\leq\frac{2n}{3}}\alpha\cdot\check{\delta}_i\cdot\Phi_{i,-} + \sum_{i>\frac{2n}{3}}\alpha\cdot\check{\delta}_i\cdot\Phi_{i,-}.\end{aligned} \tag{2.45}$$

For $i \geq 2n/3$ we have $p_i = \frac{2i-1}{n^2} \geq \frac{4}{3n} - \frac{1}{n^2}$. Using this with $p_i \leq p_n \leq 2/n$ and $\check{\alpha} \geq \alpha - \alpha^2$, we can bound $\check{\delta}_i = \lambda n\cdot\left(1/n\cdot\ell_\top - p_i\cdot\check{\alpha}/\alpha\right) \leq \lambda\cdot(-1/3 + \frac{1+\alpha}{n}) + 2\alpha\lambda \leq -\lambda/6 + 2\alpha\lambda$. Setting $\Phi_{\leq 2n/3,-} \coloneqq \sum_{i\leq 2n/3}\Phi_{i,-}$ and $\Phi_{>2n/3,-} \coloneqq \sum_{i>2n/3}\Phi_{i,-}$, together with Proposition 2.3.7 this yields

$$\begin{aligned}&\mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}\right]\\ &\leq 6\alpha^2\lambda\cdot\Phi_- + \frac{5}{4}\alpha\lambda\cdot\Phi_{\leq 2n/3,-} - \frac{\alpha\lambda}{6}\cdot\Phi_{>2n/3,-} + 2\alpha^2\lambda\cdot\Phi_{>2n/3,-}\\ &\leq \left(8\alpha^2\lambda - \alpha\lambda/6\right)\cdot\Phi_- + \left(\frac{5}{4}\alpha\lambda + \alpha\lambda/6\right)\cdot\Phi_{\leq 2n/3,-}\\ &\leq -\frac{\varepsilon\alpha\lambda}{2}\cdot\Phi_- + 2\alpha\lambda\cdot\Phi_{\leq 2n/3,-},\end{aligned} \tag{2.46}$$

where the last inequality uses $\alpha \leq 1/32 \leq \frac{1}{16} - \frac{1}{48}\varepsilon$. With this, the assumption $\mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}\right] \geq -\frac{\varepsilon\alpha\lambda}{4} \cdot \Phi_-$ implies that $\Phi_- \leq \frac{8}{\varepsilon} \cdot \Phi_{\leq 2n/3,-} \leq \frac{8}{\varepsilon} \cdot \frac{2n}{3} e^{\frac{\alpha L}{n/3}} = \frac{16n}{3\varepsilon} e^{\frac{3\alpha L}{n}}$ (the last inequality uses that none of the $2n/3$ remaining bins can have a load higher than $L/(n/3)$). To finish the proof, assume $\Phi_- > \frac{\varepsilon}{4} \cdot \Phi_+$ (otherwise the lemma holds). Combining this with the upper bound on Φ_- and with the lower bound on Φ_+, we get

$$\frac{16n}{3\varepsilon} e^{\frac{3\alpha L}{n}} \geq \Phi_- > \frac{\varepsilon}{4} \cdot \Phi_+ \geq \frac{\varepsilon n}{16} \cdot e^{\frac{4\alpha L}{n}}. \tag{2.47}$$

Thus, the excess load can be bounded by $L < \frac{n}{\alpha} \cdot \ln\left(\frac{256}{3\varepsilon^2}\right)$. Now, the lemma's statement follows from $\Phi = \Phi_+ + \Phi_- < \frac{5}{\varepsilon} \cdot \Phi_- \leq \frac{80n}{3\varepsilon^2} e^{\frac{3\alpha L}{n}} = \varepsilon^{-8} \cdot \mathcal{O}(n)$. □

This completes the proof of the lemmas used in the proof of Lemma 2.3.1. The proof of the result above uses Lemma 2.3.1 to bound the expected drop of the potential function Φ in a single round. The proof of Lemma 2.3.1 is by case analysis. The case analysis follows the same line of argument used by Peres et al. [83] and Talwar and Wieder [96].

Lemma 2.3.1. *Consider an arbitrary round $t+1$ of the 2-choice process and the constants ε (from Proposition 2.3.10) and $\alpha \leq \min(\ln(10/9), \varepsilon/8)$. For $\lambda \in [1/4, 1]$ we have*

$$\mathbb{E}\left[\Phi(\boldsymbol{X}(t+1)) \mid \boldsymbol{X}(t)\right] \leq \left(1 - \frac{\varepsilon\alpha\lambda}{4}\right) \cdot \Phi(\boldsymbol{X}(t)) + \varepsilon^{-8} \cdot \mathcal{O}(n). \tag{2.48}$$

Proof. The proof is via case analysis.

Case 1: $x_{\frac{n}{4}} \geq \varnothing$ and $x_{\frac{3n}{4}} \leq \varnothing$:

This case considers reasonably balanced allocations. The desired bound follows from Lemma 2.3.11 and Lemma 2.3.12.

Case 2: $x_{\frac{n}{4}} \geq x_{\frac{3n}{4}} \geq \varnothing$:

Recall that by definition

$$\mathbb{E}\left[\Delta(t+1) \mid \boldsymbol{X}(t)\right] = \mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] + \mathbb{E}\left[\Delta_-(t+1) \mid \boldsymbol{X}(t)\right]$$

By Lemma 2.3.12 we derive

$$\mathbb{E}\left[\Delta(t+1) \mid \boldsymbol{X}(t)\right] \leq \mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] - \varepsilon\alpha\lambda \cdot \Phi_-(\boldsymbol{X}(t)) + \frac{\alpha\lambda n}{2}$$

The desired result holds when,

$$\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] - \varepsilon\alpha\lambda \cdot \Phi_-(\boldsymbol{X}(t)) + \frac{\alpha\lambda n}{2} \leq -\frac{\varepsilon\alpha\lambda}{4}\Phi(\boldsymbol{X}(t)) + \varepsilon^{-8} \cdot \mathcal{O}\left(n\right)$$

Rearranging we obtain

$$\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] \leq -\frac{\varepsilon\alpha\lambda}{4}\Phi(\boldsymbol{X}(t)) + \varepsilon^{-8} \cdot \mathcal{O}\left(n\right) + \varepsilon\alpha\lambda \cdot \Phi_-(\boldsymbol{X}(t)) - \frac{\alpha\lambda n}{2}$$

Since the RHS is lower bounded by $\frac{-\varepsilon\alpha}{4}\Phi_+$ it follows that the desired bound holds when $\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] \leq \frac{-\varepsilon\alpha}{4}\Phi_+$.

It remains to show that the desired bound holds when $\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] \geq \frac{-\varepsilon\alpha}{4}\Phi_+$. Lemma 2.3.13 states that when $X_{\frac{3n}{4}} \geq \varnothing$ and $\mathbb{E}\left[\Delta_+(t+1) \mid \boldsymbol{X}(t)\right] \geq \frac{-\varepsilon\alpha}{4}\Phi_+$ then either $\Phi_+(\boldsymbol{X}(t)) \leq \frac{\varepsilon}{4} \cdot \Phi_-(\boldsymbol{X}(t))$ or $\Phi(\boldsymbol{X}(t)) = \varepsilon^{-8} \cdot \mathcal{O}\left(n\right)$. This gives two subcases that we now consider.

Case 2.1: $\Phi_+(\boldsymbol{X}(t)) \leq \frac{\varepsilon}{4} \cdot \Phi_-(\boldsymbol{X}(t))$**:**

Using Lemma 2.3.9 and Lemma 2.3.12 we obtain

$$\begin{aligned}
\mathbb{E}\left[\Delta(t+1) \mid \boldsymbol{X}(t)\right] &\leq 2\alpha\lambda \cdot \Phi_+(\boldsymbol{X}(t)) - \varepsilon\alpha\lambda \cdot \Phi_-(\boldsymbol{X}(t)) + \frac{\alpha\lambda n}{2} \\
&\leq \frac{2\alpha\lambda\varepsilon}{4} \cdot \Phi_-(\boldsymbol{X}(t)) - \varepsilon\alpha\lambda \cdot \Phi_-(\boldsymbol{X}(t)) + \frac{\alpha\lambda n}{2} \\
&= -\frac{\varepsilon\alpha\lambda}{2} \cdot \Phi_-(\boldsymbol{X}(t)) + \frac{\alpha\lambda n}{2} \\
&\leq -\frac{\varepsilon\alpha\lambda}{4} \cdot \Phi(\boldsymbol{X}(t)) + \varepsilon^{-8} \cdot \mathcal{O}\left(n\right).
\end{aligned} \tag{2.49}$$

where the last inequality follows from the case assumption.

Case 2.2: $\Phi(\boldsymbol{X}(t)) = \varepsilon^{-8} \cdot \mathcal{O}(n)$**:**

Using Lemma 2.3.9 we obtain that $\mathbb{E}[\Delta(t+1) \mid \boldsymbol{X}(t)] \leq 2\alpha\lambda\varepsilon^{-8} \cdot \mathcal{O}(n)$ It remains to show that

$$2\alpha\lambda\varepsilon^{-8} \cdot \mathcal{O}(n) \leq -\frac{\varepsilon\alpha\lambda}{4} \cdot \Phi(\boldsymbol{X}(t)) + \varepsilon^{-8} \cdot \mathcal{O}(n). \tag{2.50}$$

Since $\Phi(\boldsymbol{X}(t)) = \varepsilon^{-8} \cdot \mathcal{O}(n)$,

$$2\alpha\lambda\varepsilon^{-8} \cdot \mathcal{O}(n) \leq \left(1 - \frac{\varepsilon\alpha\lambda}{4}\right) \cdot \varepsilon^{-8} \cdot \mathcal{O}(n). \tag{2.51}$$

This holds where $2\alpha\lambda \leq \left(1 - \frac{\varepsilon\alpha\lambda}{4}\right)$. By definition $\alpha \leq 1/8$ and $\lambda < 1$. The result follows.

Case 3: $x_{\frac{3n}{4}} \leq x_{\frac{n}{4}} \leq \varnothing$**:**

This case is similar to case 2. For $\mathbb{E}[\Delta_-(t+1) \mid \boldsymbol{X}(t)] \leq \frac{-\varepsilon\alpha n}{4}\Phi_-$ the results follows from Lemma 2.3.12. For $\mathbb{E}[\Delta_-(t+1) \mid \boldsymbol{X}(t)] \geq \frac{-\varepsilon\alpha n}{4}\Phi_-$ two sub-cases are given by Lemma 2.3.14.

Case 3.1: $\Phi_-(\boldsymbol{X}(t)) \leq \frac{\varepsilon}{4} \cdot \Phi_+(\boldsymbol{X}(t))$**:**

The result follows from applying Lemma 2.3.14 and Lemma 2.3.9.

Case 3.2: $\Phi(\boldsymbol{X}(t)) = \varepsilon^{-8} \cdot \mathcal{O}(n)$**:**

This result follows from Lemma 2.3.9. □

2.3.2 Maximum Load

The goal of this section is to prove Theorem 2.3.18 that upper bounds the maximum load in an arbitrary round w.h.p.

Theorem 2.3.18 (Maximum Load)**.** *Let $\lambda = \lambda(n) \in [1/4, 1)$. Fix an arbitrary round t of the 2-choice process. The maximum load of any bin is (w.h.p.) bounded by $\mathcal{O}\left(\log \frac{n}{1-\lambda}\right)$.*

We begin by restating $\Phi(\boldsymbol{x})$ and $\Psi(\boldsymbol{x})$ from Equation (2.9).

$$\Phi(\boldsymbol{x}) \coloneqq \sum_{i\in[n]} e^{\alpha\cdot(x_i-\varnothing)} + \sum_{i\in[n]} e^{\alpha\cdot(\varnothing-x_i)}$$
$$\Psi(\boldsymbol{x}) \coloneqq \sum_{i\in[n]} x_i$$

The potential function $\Psi(\boldsymbol{X}(t))$ describes the total load of the system at time t.

Warm Up

To get an intuition for our analysis, consider the case $t = \text{poly}(n)$ and assume that the number of balls allocated in each round is fixed and exactly $\lambda \cdot n \leq n$. Lemma 2.3.4 states the load difference between any two bins is upper bounded by $\mathcal{O}(\ln n)$ w.h.p. Combining this with a union bound over all $t = \text{poly}(n)$ rounds we obtain that the load difference between any pair of bins and for all $t' < t$ is bounded by $\mathcal{O}(\ln n)$ w.h.p. Using the combinatorial observation that, while the load distance to the average is bounded by some $b \geq 0$, the bound $\Psi \leq 2b \cdot n$ is invariant under the 2-choice process (Lemma 2.3.15), it follows that we get for $b = \mathcal{O}(\ln n)$ that $\Psi(\boldsymbol{X}(t)) \leq 2b \cdot n = \mathcal{O}(n \cdot \ln n)$, as required.

Lemma 2.3.15. *For an arbitrary round t, let $b \geq 0$ and consider a configuration $\boldsymbol{x}(t)$ with $\Psi(\boldsymbol{x}(t)) \leq 2b \cdot n$ and $\Phi(\boldsymbol{x}(t)) \leq e^{\alpha\cdot b}$. Let $\boldsymbol{x}(t+1)$ denote the configuration after one step of the 2-choice process. Then $\Psi(\boldsymbol{x}(t+1)) \leq 2b \cdot n$.*

Proof. We distinguish two cases: if there is no empty bin, then all n bins delete one ball. Since the maximum number of new balls is n, the number of balls cannot increase. That is, we have $\Psi(\boldsymbol{x}(t+1)) \leq \Psi(\boldsymbol{x}(t)) \leq 2b \cdot n$. Now consider the case that there is at least one empty bin. Let $\eta \in (0, 1]$ denote the fraction of empty bins (i.e., there are exactly $\eta \cdot n > 0$ empty bins). Since the minimal load is zero, Observation 2.3.3 implies $\max_i x_i(t) \leq 2b$. Thus, the total number of balls in configuration $\boldsymbol{x}(t)$ is at most $(1-\eta)n \cdot 2b$. Exactly $(1-\eta)n$ balls are deleted (one from each non-empty bin) and at most n new balls enter the system. We get $\Psi(\boldsymbol{x}(t+1)) \leq (1-\eta)n \cdot 2b - (1-\eta)n + n = (1-\eta)n \cdot (2b-1) + n \leq 2b \cdot n$. □

However, the case for $t = \omega(\text{poly}(n))$ is considerably more involved. We would like to be able to use the standard technique of designing a suitable potential function that drops when it is high. However, the fact that the number of balls in the system is only guaranteed to decrease when the total load is high *and* the load distance to the average is low makes it challenging to design a suitable potential function that drops fast enough when it is high. For this reason we deviate from this standard technique and elaborate on the idea of the $t = \text{poly}(n)$ case. In the $t = \text{poly}(n)$ case we used lemma 2.3.4 to bound the load difference between any pair of bins w.h.p for all $t' < t$ by $\mathcal{O}(\ln n)$. Since this is not possible for $t \gg \text{poly}(n)$), we prove (w.h.p.) an *adaptive bound* of $\mathcal{O}(\ln(t - t') \cdot f(\lambda))$ for all $t' < t$, where f is a suitable function (Lemma 2.3.16).

We then consider the last round $t'' < t$ with an empty bin. Note that since the total system load only increases when there is an empty bin that $t'' < t$ is the last round where the total system load increases. Lemma 2.3.15 yields a bound of $\Psi(\boldsymbol{X}(t'')) = 2 \cdot \mathcal{O}(\ln(t - t'') \cdot f(\lambda)) \cdot n$ on the total load at time t''. Using the same combinatorial observation as in the $t = \text{poly}(n)$ case, we get that (w.h.p.) $\Psi(\boldsymbol{X}(t)) \leq \Psi(\boldsymbol{X}(t'')) = 2 \cdot \mathcal{O}(\ln(t - t'') \cdot f(\lambda)) \cdot n$.

The final step is to show that the load at time t'' (which is logarithmic in $t - t''$) decreases *linearly* in $t - t''$, showing that the time interval $t - t''$ cannot be too large (or we would get a negative load at time t).

The following lemma bounds the probability of two events. Statement (a) of the lemma bounds Φ over an arbitrarily large interval $[0, t)$ using a union bound over all rounds $t' < t$. Since the interval can be arbitrarily large the bound must be adaptive and allow for larger errors that occur with larger intervals. Statement (b) of the lemma bounds the number of balls allocated w.h.p.

Lemma 2.3.16. *Let $\lambda \in [1/4, 1)$. Fix a round t. For $i \in \mathbb{N}$ with $t - i \cdot \frac{8 \log n}{1-\lambda} \geq 0$ define $\mathcal{I}_i \coloneqq [t - i \cdot \frac{8 \ln n}{1-\lambda}, t]$. Let Y_i be the number of balls which spawn in $\mathcal{I}_i$.*

(a) Define the (good) smooth event $\mathcal{S}_t \coloneqq \bigcap_{t'<t} \big(\Phi(\boldsymbol{X}(t')) \leq |t - t'|^2 \cdot n^2\big)$. Then $\Pr(\mathcal{S}_t) = 1 - \mathcal{O}(n^{-1})$.

(b) Define the (good) bounded balls event $\mathcal{B}_t \coloneqq \bigcap_i \big(Y_i \leq \frac{1+\lambda}{2} \cdot |\mathcal{I}_i| \cdot n\big)$. Then $\Pr(\mathcal{B}_t) = 1 - \mathcal{O}(n^{-1})$.

Proof. Consider an arbitrary time $t' < t$. By Lemma 2.3.2 we have $\mathbb{E}[\Phi(t')] \leq n/\varepsilon$. Using Markov's inequality, this implies

$$\Pr(\Phi(t') \geq |t - t'|^2 \cdot n^2) \leq 1/(\varepsilon \cdot |t - t'|^2 \cdot n).$$

Using the union bound over all $t' < t$ we calculate

$$\Pr\left(\bar{\mathcal{S}}_t\right) \leq \sum_{t'<t} \Pr\left(\Phi(t') \geq |t - t'|^2 \cdot n^2\right) \leq \frac{1}{\varepsilon n} \cdot \sum_{t'<t} \frac{1}{|t - t'|^2} \leq \frac{\pi^2}{6\varepsilon \cdot n} = \mathcal{O}\left(n^{-1}\right), \tag{2.52}$$

where the last inequality uses the solution to the Basel problem. [1] This proves the first statement.

For the second statement, let $Z_i \coloneqq |\mathcal{I}_i| \cdot n - Y_i$ be the number of balls that did not spawn during $\mathcal{I}_i$. Note that Z_i is a sum of $|\mathcal{I}_i| \cdot n$ independent indicator variables with $\mathbb{E}[Z_i] = (1 - \lambda) \cdot |\mathcal{I}_i| \cdot n = 8i \cdot \ln n$. Chernoff yields $\Pr(Z_i \leq (1 - \lambda) \cdot |\mathcal{I}_i| \cdot n/2) \leq e^{-8i \cdot \ln n/8} = n^{-i}$. The desired statement follows from applying the identity $Z_i = |\mathcal{I}_i| \cdot n - Y_i$ and taking the union bound. □

The next lemma assumes that both events $\mathcal{S}_t$ and $\mathcal{B}_t$ hold and bounds the total number of balls in the system. As previously stated, the total number of balls in the system can only increase when there is an empty bin. Lemma 2.3.17 upper bounds the potential Ψ.

[1] The solution to the Basel problem states that $\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$

Lemma 2.3.17. *Fix a round t and assume that both $\mathcal{S}_t$ and $\mathcal{B}_t$ hold. Then,*

$$\Psi(\boldsymbol{X}(t)) \leq \mathcal{O}\left(n \cdot \ln\left(\frac{n}{1-\lambda}\right)\right).$$

Proof. Let $t' < t$ be the last time when there was an empty bin and set $\Delta \coloneqq t - t'$. Note that t' is well defined, as we have $X_i(0) = 0$ for all $i \in [n]$.

Since $\mathcal{S}_t$ holds, we have $\Phi(\boldsymbol{X}(t')) \leq \Delta^2 \cdot n^2 = \exp\left(\ln(\Delta^2 \cdot n^2)\right)$. By choice of t' we have $\min_i X_i(t') = 0$. Together with Observation 2.3.3 we get $\max_i X_i(t') \leq 2\ln\big(\Delta^2 \cdot n^2\big)/\alpha$. Summing up over all bins (and pulling out the square), this implies $\Psi(\boldsymbol{X}(t')) \leq 4n \cdot \ln\big(\Delta \cdot n\big)/\alpha$. Applying Lemma 2.3.15 yields

$$\Psi(\boldsymbol{X}(t'+1)) \leq 4n \cdot \ln\big(\Delta \cdot n\big)/\alpha \tag{2.53}$$

By choice of t', there is no empty bin in $\boldsymbol{X}(t'')$ for all $t'' \in \{\, t'+1, t'+2, \ldots, t-1 \,\}$. Thus, during each of these rounds exactly n balls are deleted.

To bound the number of balls generated, let i be maximal with $\mathcal{I}_i \subseteq [t', t]$ (as defined in Lemma 2.3.16). That is, the largest interval that is a multiple of $\frac{8\ln(n)}{(1-\lambda)}$ rounds. Since $\mathcal{B}_t$ holds and using the maximality of i, the number of balls that are generated during $[t', t]$ can be bounded as follows.

$$(1+\lambda)|\mathcal{I}_i| \cdot n/2 + \frac{8\ln n}{1-\lambda} \cdot n \leq (1+\lambda)\Delta \cdot n/2 + \frac{8\ln n}{1-\lambda} \cdot n \tag{2.54}$$

Since the number of balls generated during the interval $\mathcal{I}_i$ is at most the bound from statement (b) in lemma 2.3.16. Note there are at most $\frac{8\ln n}{1-\lambda}$ rounds not in $\mathcal{I}_i$ through the maximality of i. We assume pessimistically that n balls are generated in each of these rounds. The second inequality follows since $|\mathcal{I}_i| \leq \Delta$ by definition.

We calculate

$$\begin{aligned}\Psi(\boldsymbol{X}(t)) &\leq \Psi(\boldsymbol{X}(t'+1)) - \Delta \cdot n + Y \\ &\leq \frac{4n}{\alpha}\ln(\Delta \cdot n) - \frac{1-\lambda}{2}\Delta \cdot n + \frac{8\ln n}{1-\lambda} \cdot n \\ &= \frac{1-\lambda}{2} \cdot n \cdot \left(\frac{8}{\alpha(1-\lambda)} \cdot \ln(\Delta \cdot n) - \Delta + \frac{16\ln n}{(1-\lambda)^2}\right) \\ &\leq \frac{1-\lambda}{2} \cdot \Delta \cdot n \cdot \left(\frac{24}{\alpha(1-\lambda)^2} \cdot \frac{\ln(\Delta \cdot n)}{\Delta} - 1\right)\end{aligned} \tag{2.55}$$

First inequality substitutes Equation 2.53 and using the bound in Equation 2.54. With $f = f(\lambda) \coloneqq 24/\big(\alpha(1-\lambda)^2\big)$ the last factor becomes $f \cdot \ln(\Delta \cdot n)/\Delta - 1$. It is negative if and only if $\Delta > f \cdot \ln(\Delta \cdot n)$. This inequality holds for any $\Delta > -f \cdot W_{-1}(-\frac{1}{f \cdot n})$, where W_{-1} denotes the lower branch of the Lambert W function[2]. This implies that $\Delta \leq -f \cdot W_{-1}(-1/fn)$, since otherwise we would have $\Psi(\boldsymbol{X}(t)) < 0$ i.e., negative load, which is clearly a contradiction. Using the Taylor approximation $W_{-1}(x) = \ln(-x) - \ln\big(\ln(-1/x)\big) - \mathcal{O}(1)$ as $x \to -0$, we get

$$\Delta \leq -f \cdot W_{-1}\left(-\frac{1}{f \cdot n}\right) \leq f \cdot \ln(f \cdot n) + f \cdot \ln\big(\ln(f \cdot n)\big) + f \leq 2f \cdot \ln(f \cdot n). \tag{2.56}$$

Finally, we use this bound on Δ to get

$$\begin{aligned}\Psi(\boldsymbol{X}(t)) &\leq \Psi(\boldsymbol{X}(t'+1)) \leq \frac{4n}{\alpha} \cdot \ln(\Delta \cdot n) \leq \frac{4n}{\alpha} \cdot \ln\big(2fn \cdot \ln(fn)\big) \\ &\leq \frac{4n}{\alpha} \cdot \ln\left(\frac{48n}{\alpha(1-\lambda)^2} \cdot \ln\left(\frac{24n}{\alpha(1-\lambda)^2}\right)\right) \leq \mathcal{O}\left(n \cdot \ln\left(\frac{n}{1-\lambda}\right)\right).\end{aligned}$$

□

Theorem 2.3.18 (Maximum Load)**.** *Let $\lambda = \lambda(n) \in [1/4, 1)$. Fix an arbitrary round t of the 2-choice process. The maximum load of all bins is (w.h.p.) bounded by $\mathcal{O}\left(\log \frac{n}{1-\lambda}\right)$.*

Proof. Combining Lemma 2.3.17 with the fact that the events $\mathcal{S}_t$ and $\mathcal{B}_t$ hold with high probability (Lemma 2.3.16), we immediately get that (w.h.p.) $\Psi(\boldsymbol{X}(t)) = \mathcal{O}\left(n \cdot \ln\left(\frac{n}{1-\lambda}\right)\right)$. As described at the beginning of this section, combining this with Lemma 2.3.4 proves Theorem 2.3.18. □

[2]Note that $-\frac{1}{f \cdot n} \geq -1/e$, so that $W_{-1}(-\frac{1}{f \cdot n})$ is well defined.

2.3.3 Stability

This section proves Theorem 2.3.20 and shows that the 2-choice process is stable. As with the 1-choice process we show that the Markov chain for the 2-choice process is positive recurrent. In this section we will consider the potential function Γ. Recall

$$\Gamma(\boldsymbol{x}) \coloneqq \Phi(\boldsymbol{x}) + \tfrac{n}{1-\lambda} \cdot \Psi(\boldsymbol{x})$$

where,

$$\Phi(\boldsymbol{x}) \coloneqq \sum_{i\in[n]} e^{\alpha\cdot(x_i-\varnothing)} + \sum_{i\in[n]} e^{\alpha\cdot(\varnothing-x_i)}$$
$$\Psi(\boldsymbol{x}) \coloneqq \sum_{i\in[n]} x_i$$

The first lemma states that for a sufficiently high value of Γ, this potential function decreases.

Lemma 2.3.19. *Let $\lambda \in [1/4, 1)$. If $\Gamma(\boldsymbol{X}(t)) \geq 2\frac{n^4}{(1-\lambda)^2\lambda}$, then*

$$\mathbb{E}\left[\Gamma(\boldsymbol{X}(t+1)) - \Gamma(\boldsymbol{X}(t)) \mid \boldsymbol{X}(t)\right] \leq -1.$$

Proof. Assume $\boldsymbol{X}(t) = x$ is fixed. By definition of Γ, we have $\Phi(x) \geq \Gamma(x)/2$ or $\Psi(x) \geq \Gamma(x)/2$. We now show that the result follows in both cases.

If $\Phi(x) \geq \Gamma(x)/2$, then we have, by Lemma 2.3.1, a potential drop of

$$\begin{aligned}\mathbb{E}\left[\Phi(\boldsymbol{X}(t+1)) - \Phi(x) \mid \boldsymbol{X}(t) = x\right] &\leq -(\varepsilon\alpha\lambda/4)\cdot\Phi(x) + n\log n \\ &\leq -(\varepsilon\alpha\lambda/8)\cdot\Gamma(x) + n\log n.\end{aligned}$$

Note that, by definition of Ψ, $\Psi(\boldsymbol{X}(t+1)) - \Psi(x) \leq n$. Together with $\Gamma(x) \geq$

$\frac{8(n\log n+n^2/(1-\lambda)+1)}{e\alpha\lambda}$,

$$\mathbb{E}\left[\Gamma(\boldsymbol{X}(t+1)) - \Gamma(x) \mid \boldsymbol{X}(t) = x\right] \leq -\frac{\varepsilon\alpha\lambda}{8}\Gamma(x) + n\log n + (n/(1-\lambda))\cdot n$$
$$\leq -1.$$

Otherwise, i.e., if $\Phi(x) < \Gamma(x)/2$, we have that

(i) the load difference is, by Observation 2.3.3, bounded by $2\ln(\Gamma(x)/2)/\alpha$, and

(ii) $\Psi(x) \geq \Gamma(x)/2$ must hold. This implies that $\varnothing \geq \frac{1}{n}\left(\frac{\Gamma(x)/2}{\frac{n}{1-\lambda}}\right) = \frac{(1-\lambda)\cdot\Gamma(x)}{2n^2}$.

From (i) and (ii) we have that the minimum load is at least $\frac{(1-\lambda)\cdot\Gamma(x)}{2n^2} - 2\ln(\Gamma(x)/2)/\alpha$.

Since $\Gamma(x) \geq 2\frac{n^4}{(1-\lambda)^2\lambda}$,

$$\frac{(1-\lambda)\cdot\Gamma(x)}{2n^2} - \frac{2\ln(\Gamma(x)/2)}{\alpha} \geq \frac{n^2}{(1-\lambda)^2\lambda} - \frac{2}{\alpha}\ln\left(\frac{n^4}{(1-\lambda)^2\lambda}\right) \tag{2.57}$$
$$\geq 1$$

where the last inequality follows for large enough n since α is a constant.

Differentiating the left hand side of equation 2.57 with respect to $\Gamma(x)$ we obtain

$$\frac{1-\lambda}{n^2} - \frac{2}{\alpha\Gamma(x)} > 0$$

Hence the left hand side of Equation 2.57 is increasing in $\Gamma(x)$.

It follows that for all $\Gamma(x) \geq 2\frac{n^4}{(1-\lambda)^2\lambda}$, every bin has load of at least 1. Thus each bin will delete one ball and the number of balls arriving is λn in expectation. Hence,

$$\mathbb{E}\left[\Psi(\boldsymbol{X}(t+1)) - \Psi(x) \mid \boldsymbol{X}(t) = x\right] = -\frac{n}{1-\lambda}(1-\lambda)n$$

Now,

$$\begin{aligned}
&\mathbb{E}\left[\Gamma(\boldsymbol{X}(t+1)) - \Gamma(x) \mid \boldsymbol{X}(t) = x\right] \\
&= \mathbb{E}\left[\Phi(\boldsymbol{X}(t+1)) - \Phi(x) \mid \boldsymbol{X}(t) = x\right] - \frac{n}{1-\lambda}(1-\lambda)n \qquad (2.58)\\
&\leq n \log n - \frac{n}{1-\lambda}(1-\lambda)n \leq -1.
\end{aligned}$$

Thus, $\mathbb{E}\left[\Gamma(\boldsymbol{X}(t+1)) - \Gamma(x) \mid \boldsymbol{X}(t) = x\right] \leq -1$, which yields the claim. □

This shows that when Γ is sufficiently large then the potential function decreases in expectation.

We are ready to prove that the Markov chain for the 2-choice process is positive recurrent (Theorem 2.3.20).

Theorem 2.3.20. *Let $\lambda = \lambda(n) \in [1/4, 1)$. The Markov chain $\boldsymbol{X}$ of the 2-choice process is positive recurrent.*

Proof. The proof proceeds by applying Theorem 2.1.1.

We now define the parameters of Theorem 2.1.1. Let $\zeta(t) = \boldsymbol{X}(t)$ and hence Ω is the state space of X. First we observe that Ω is countable since there are a constant number of bins (n is consider a constant in this matter) each having a load which is a natural number. We define $\phi(\boldsymbol{X}(t))$ to be $\Gamma(\boldsymbol{X}(t))$. We define $C = \{\, x \mid \Gamma(x) \leq 2\frac{n^4}{(1-\lambda)^2\lambda} \,\}$. Define $\beta(x) = 1$ and $\eta = 1$. We now show that the preconditions (a) and (b) of Theorem 2.1.1 are fulfilled. Let $x \notin C$. By definition of C and $\phi(\boldsymbol{X}(t))$, and from Lemma 2.3.19 we have

$$\mathbb{E}\left[\phi(X(t+1)) - \phi(x) \mid \boldsymbol{X}(t) = x\right] \leq \mathbb{E}\left[\Gamma(\boldsymbol{X}(t+1)) - \Gamma(x) \mid \boldsymbol{X}(t) = x\right] \leq -1. \qquad (2.59)$$

Let $x \in C$. Recall that $\Gamma(\boldsymbol{X}(t)) = \Phi(\boldsymbol{X}(t)) + \frac{n}{1-\lambda}\Psi(\boldsymbol{X}(t))$. By Lemma 2.3.14 and the fact that the number of balls arriving in one round is bounded by n, we

derive,

$$\begin{aligned}\mathbb{E}\left[\phi(X(t+1)) \mid \boldsymbol{X}(t) = x\right] &= \mathbb{E}\left[\Phi(\boldsymbol{X}(t+1)) \mid \boldsymbol{X}(t) = x\right] + \mathbb{E}\left[\Psi(\boldsymbol{X}(t+1)) \mid \boldsymbol{X}(t) = x\right] \\ &\leq \left(\left(1 - \frac{\varepsilon\alpha\lambda}{4}\right) 2^{\frac{n^4}{(1-\lambda)^2\lambda}}\right) + \frac{n}{1-\lambda} n < \infty. \end{aligned} \tag{2.60}$$

The claim follows by applying Theorem 2.1.1 with Equations (2.59) and (2.60).

□

2.4 Conclusion

Our results show that the power of two choices carries over to generalized setting with deletion. Similar to the classic setting without deletions, the maximum load under Greedy[2] is exponentially smaller than the load under Greedy[1]. Moreover, Greedy[2] can handle much larger arrival rates w.r.t. the maximum load difference.

One might assume that our (upper) bounds for Greedy[1] carry over to Greedy[2] (and, in general, to Greedy[d]) via a simple coupling (similar to Azar et al. [12]). However, we are not aware of such a coupling in the *parallel* setting. In fact, for naïve approaches to such a coupling, it is not hard to come up with situations where Greedy[2] behaves worse than Greedy[1] (in one step). It would be interesting to find arguments that, for example, for any $d \in \mathbb{N}$ Greedy[$d+1$] behaves "better" than Greedy[d].

Another open questions is concerned with arrival rates $\lambda \geq 1$ (this would require a slight reformulation of our model, which currently assumes the existence of n generators that generate balls with a probability of λ). As mentioned in Section 2.3.1, our assumptions on λ for proving bounds on the smoothness (Lemma 2.3.4) are merely for convenience. The corresponding proofs carry over (with minor modifications) to any constant λ, no matter whether $\lambda < 1$ or $\lambda > 1$. Thus, for Greedy[2] we know that the load difference between any two bins is still logarithmic, even for arrival rates > 1. Still, the maximum load obviously diverges for $\lambda \geq 1$. It would be interesting to quantify this divergence in terms of λ.

Chapter 3

Plurality Consensus

In this chapter we study the plurality consensus problem. We present two protocols that solve the plurality consensus problem. Our protocols are inspired by a relationship between the plurality consensus problem and distributed load balancing. This connection allows us to design protocols that are able to solve the plurality consensus problem for a multitude of communication patterns and an arbitrarily small bias.

3.1 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 [42, 82, 84], social networks [34, 79, 80], as well as the modelling of biological interactions [28, 31]. All these areas typically demand both very simple and space-efficient protocols. However the communication models considered can vary from anything between simple sequential communication with a single neighbour (often used in biological settings as a simple variant of asynchronous communication [9]) to fully parallel communication where all nodes communicate with all their neighbours simultaneously (e.g. broadcasting models in distributed computing). This diversity turns out to be a major obstacle

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

In this chapter we present two simple protocols for the plurality consensus problem 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 Avin et al. [11] and Sauerwald and Sun [91] 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 3.2). These smoothing properties are inspired by similar smoothing properties used by Sauerwald and Sun [91] 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 labelled 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 where 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 3.2). For both protocols we show a strong correlation between their running

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

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 [91].

To give some more concrete examples of our results, let $T \coloneqq \mathcal{O}(\log n/(1-\lambda_2))$, 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 neighbours at once).

To the best of our knowledge, these match the best known runtime 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 3.1.2.

Reading vs. Amplification Protocols. In addition to solving the plurality consensus problem for a wide range of communication patterns and arbitrarily small bias, both our Balance and Shuffle protocols allow nodes to estimate the total number tokens of a given opinion. In fact, with minor adaptations both protocols allow nodes to estimate the frequency of all opinions.

Using the terminology of Ghaffari and Parter [56], protocols such as ours can be classified as *reading protocols*. Ghaffari and Parter [56] roughly divide protocols for the plurality consensus problem in to two main classes namely, reading and plurality amplification. Reading protocols are a natural approach for solving the plurality consensus problem. Nodes (collectively or individually) estimate the frequencies of the opinions and choose the most frequent. On the other hand, plurality amplification protocols do not attempt to estimate the initial distribution but instead seek to

modify the distribution such that the bias towards the plurality opinion increases. These protocols seek to apply this idea repeatedly until all nodes learn the plurality opinion.

Ghaffari and Parter [56] suggest that it seems implausible that a reading protocol might solve the plurality consensus problem in polylogarithmic time using only polylogarithmic space. Both our protocols are able to achieve this goal for a range of parameters. In particular our protocols work for graphs other than the complete graph. On the other hand, to the best of our knowledge, there is no plurality amplification protocol that solves the plurality consensus problem for an arbitrary bias in polylogarithmic time on arbitrary graphs.

3.1.1 Related Work

There is a diverse body of literature that analyzes consensus problems under various models and assumptions. Results differ in the considered network 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, or spanning tree). To capture this diverse spectrum, we classify results into *population protocols*, *sensor networks*, and *pull voting*. This classification is neither unique nor injective but merely an attempt to make the overview more accessible and highlights the extensive interest in consensus problems. A condensed form of this discussion is given in Table 3.1. We will not discuss work whose focus is too far away from our work, e.g., consensus on some arbitrary opinion, leader election, robustness concerns, or Byzantine models.

Population Protocols. The first area of work we consider comes from *population protocols*. Population protocols model interactions between large populations of very simple entities (such as molecules). Entities are modelled as finite state machines with a small state space and communicate asynchronously. In each step, an edge is

[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 3.1: 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	$T \cdot t_{\text{mix}}$ $T \cdot \log(n)/(1-\lambda_2)$ for (d-regular graph)	sync & async	Thm.3.4.10
BALANCE	✓	arbitrary	arbitrary	τ $\log(n)/(1-\lambda_2)$ for (d-regular graph)	sync & async	$k \cdot \log(n)$
[70]	✓	arbitrary	arbitrary	$D + \frac{F_2}{n_1^2} \cdot \log(k)$	broadcast	–
[74]	✓	2	arbitrary	n^5	async	1
[44]	✓	2	arbitrary	$\log n/\delta(G, n_1/n)$	async	1
[36]	expander	2	$vol(1) - vol(2) \geq 4\lambda_2^2 \cdot \lvert E\rvert$	$\log(n)$	sync	1
[35]	random d-reg	2	$\sqrt{1/d + d/n}$	$\log(n)$	sync	1
[14]	✗	$\leq n$	$\sqrt{\min\left\{k, \sqrt[3]{\frac{n}{\log(n)}}\right\} \cdot \frac{\log(n)}{n}}$	$\min\left\{k, \sqrt[3]{\frac{n}{\log(n)}}\right\} \cdot \log n$	sync	$\log(k)$
[15]	✗	$O((\frac{n}{\log(n)})^{1/3})$	$\epsilon \cdot n_2/n$	$md(c) \cdot \log(n)$	sync	$\log(k)$
[47]	✗	$O(n^\epsilon)$	$\sqrt{\log n/n}$	$k + \log(n)$	sync	$\log(k)$
[23]	✗	$o(\sqrt{n/\log(n)})$	$\gg \sqrt{\log n/n}$	$\log(n) \cdot \log\log(n)$	sync	$\log(k)$
[56]	✗	$\sqrt{n/\log(n)}$	$\sqrt{\log n/n}$	$\log(k) \cdot \log(n)$	sync	$\log(k)$
[5]	✗	2	arbitrary	$\frac{\log^2(n)}{s\alpha} + \log^2(n)$	async	$s = O(n)$ states
[7]	✗	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. [70] 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 [44], hold w.p. $1 - o(1)$. [5] also gives an expected time of $o(\log(n)/(s\alpha) + \log(n) \cdot \log(s))$.

chosen uniformly at random and only the two connected nodes communicate. We refer to this communication model as the *sequential model*. For detailed introductions to population protocols see Angluin et al. [8] and Aspnes and Ruppert [9].

Angluin et al. [7] 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 $\mathcal{O}(n \cdot \log n)$ steps. Mertzios et al. [74] suggest a 4-state protocol for *exact* majority voting, which always returns the majority opinion (independent of α) in time $\mathcal{O}(n^6)$ in arbitrary graphs and in time $\mathcal{O}(n^2 \cdot \log(n)/\alpha)$ in the clique. This is optimal in that no population protocol for exact majority can have fewer than four states [74].

Alistarh et al. [5] presents 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*[3] $\mathcal{O}\left(\frac{\log n}{s \cdot \alpha} + \log n \cdot \log s\right)$ and (w.h.p.) $\mathcal{O}\left(\frac{\log^2 n}{s \cdot \alpha} + \log^2 n\right)$. Here, s is the number of states and must be in the range $s = \mathcal{O}(n)$ and $s = \Omega(\log n \cdot \log\log n)$. Alistarh et al. [4] explore further the trade off between time and space. The authors show that any protocol solving the majority problem ($k = 2$) using $\mathcal{O}(\log\log n)$ states requires $\Omega(n/\operatorname{polylog} n)$ rounds in expectation. Their results show a separation in time complexity between protocols using $\mathcal{O}(\log\log n)$ and $\Theta\left(\log^2 n\right)$ states.

In contrast to these results, our protocols consider the case of arbitrary number $k \geq 2$ of opinions. Also, with the notable exception of Mertzios et al. [74], 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 on arbitrary, even dynamic graphs. Balance can be seen as a slightly simplified and generalized version of the protocol presented by Alistarh et al. [5], and Shuffle uses a similar idea for a speed-memory trade-off.

Sensor Networks. Another line of work has its background in sensor networks. *Quantized interval consensus* draws its motivation from signal processing. Initially,

[3] 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.

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 the sequential model.

Bénézit et al. [16] propose a protocol that is equivalent to the 4-state population protocol of Mertzios et al. [74] and prove that with probability 1 it converges in finite time, but without bounds on that convergence time.

A more recent result by Draief and Vojnovic [44] shows that this protocol (and thus Mertzios et al. [74]) needs $\mathcal{O}\left(\frac{\log n}{\delta(Q_S,\alpha)}\right)$ steps in expectation. Here, $\delta(Q_S, \alpha)$ depends on the bias α and on the spectrum of a set of matrices 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[4] $\mathcal{O}(\log n/\alpha)$). The only related result for $k > 2$ we are aware of is Bénézit et al. [17] which again proves only convergence in finite time.

Another consensus variant is *mode computation*. For example, Kuhn et al. [70] 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 $\mathcal{O}(D + F_2/n_1^2 \cdot \log k)$. 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. Kuhn et al. [70] can also be used for aggregate computation as done by Kempe et al. [67] (where the authors provide an elegant protocol to compute sums or averages in complete graphs).

Overall,the work of Draief and Vojnovic [44] and Kuhn et al. [70] are the most closely related to our work since they consider arbitrary graphs. However, our work considers more general communication models, including dynamic graphs. Similarly to Draief and Vojnovic [44], our results for $k = 2$ rely on spectral properties of the

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

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 we consider has its roots in gossiping and rumour spreading. Here, communication is typically restricted to synchronous pull requests (nodes query other nodes' opinions and use simple rules to update their own). We refer the reader to Peleg [82] for a slightly dated but thorough survey.

Cooper et al. [35] consider a voting process for $k = 2$ opinions on d-regular graphs. Nodes pull from two random neighbours and if the queried opinions are the same then the node adopts the queried opinion. For random d-regular graphs and $\alpha = \Omega\left(\sqrt{1/d + d/n}\right)$, all nodes agree (w.h.p.) in $\mathcal{O}(\log n)$ rounds on the plurality opinion. For an arbitrary d-regular graph G, they need $\alpha = \Omega(\lambda_2)$ (where $1 - \lambda_2$ is the spectral gap of G). In the follow up paper Cooper et al. [36] extend these results to expander graphs. Cooper et al. [36] show that the run time is $\mathcal{O}(\log n)$ 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.

Becchetti et al. [14] consider a similar update rule on the clique for k opinions. Here, each node pulls the opinion of three random neighbours and adopts the majority among those. The protocol requires $\mathcal{O}(\log k)$ memory bits and is shown (w.h.p.) to have a tight running time of $\Theta(k \cdot \log n)$ (given a sufficiently high bias α).

Becchetti et al. [15] build upon the idea of the 3-state population protocol from Angluin et al. [7] (but in the gossip model) and generalize it to k opinions. Nodes pull the opinion of a random neighbour in each round. If $n_1 \geq (1+\varepsilon) \cdot n_2$ for a constant $\varepsilon > 0$ and if $k = \mathcal{O}\left((n/\log n)^{1/3}\right)$, they agree (w.h.p.) on the plurality opinion in time $\mathcal{O}(\mathrm{md} \cdot \log n)$ 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 Kuhn et al. [70]).

Recent work address an open question left by Becchetti et al. [15]. Becchetti et al. [15] leave as an open question whether or not there exists a protocol solving the plurality consensus problem in polylogarithmic time using only polylogarithmic (local) memory. Berenbrink et al. [23] build upon Angluin et al. [7] and design a protocol that reaches plurality consensus (w.h.p.) in time $\mathcal{O}(\log n \cdot \log\log n)$ and uses $\log k + 4$ bits. Ghaffari and Parter [56] give an algorithm that solves the the plurality consensus problem in $\mathcal{O}(\log k \cdot \log n)$ rounds with message and memory size $\log k + \mathcal{O}(1)$ bits where the initial bias $\alpha = \Omega\left(\sqrt{\log n/n}\right)$.

The running times of gossip protocols are relatively good when compared to other protocols, like population protocols or those introduced here (cf. Table 3.1). In particular, these results do typically not show a linear dependency on the bias, as our Shuffle protocol or Alistarh et al. [5], Draief and Vojnovic [44], and Mertzios et al. [74] do. This efficiency however comes at the expense of parameter constraints. In particular, results like Becchetti et al. [15] and Berenbrink et al. [23] 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.

3.1.2 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 labelled 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.

For example, consider communication models where the maximum number of communication partners for each node is small e.g., the sequential model and balancing circuits. Let the number of tokens be $\gamma = \mathcal{O}(\log n/(\alpha^2 \cdot T))$, where T is a parameter to control the trade-off between memory and running time. The protocol does not need to know the initial bias α. The protocol works for any integral choice of γ that is suitably large. Our choice of γ fixes the trade-off parameter T. Moreover, let t_{mix} be such that any time interval $[t, t + t_{\text{mix}}]$ is ε-*smoothing* (cf. Section 3.2). 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.

For our choice of γ and the *mixing time* t_{mix} of the underlying communication pattern, SHUFFLE lets all nodes agree on the plurality opinion in $\mathcal{O}(T \cdot t_{\text{mix}})$ rounds (w.h.p.), using

$$\mathcal{O}(\gamma \cdot \log(k) + \log(T \cdot t_{\text{mix}}) + \log(\gamma \cdot T)) = \mathcal{O}\left(\frac{\log n}{\alpha^2 \cdot T} \cdot \log k + \log(T \cdot t_{\text{mix}})\right)$$

memory bits per node.

Our result shows, for example, that under the SHUFFLE protocol nodes reach plurality consensus on expanders in the sequential model in $\mathcal{O}(T \cdot n \log n)$ time steps and with $\mathcal{O}\left(\frac{\log n \cdot \log k}{T} + \log(Tn)\right)$ memory bits (assuming a constant initial bias). For arbitrary graphs, arbitrary bias, and many natural communication patterns (e.g., communicating with all neighbours 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 3.4.1).

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 located 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 3.4.4 and Lemma 3.4.5, which generalize a result by Sauerwald and Sun [91] (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 labelled 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 relatively small, our much simpler BALANCE protocol gives better results.

In BALANCE, each node u maintains a k-dimensional load vector. Where 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 $\mathcal{O}(n \cdot \log n)$ time steps with $\mathcal{O}(k)$ memory bits per node (assuming a constant initial bias).

BALANCE can be thought of as a (slightly simplified) version of Alistarh et al. [5] or Kempe et al. [67] that generalizes naturally to $k \geq 2$ and arbitrary (even dynamic) graphs. In the setting of Alistarh et al. [5] (but as opposed to Alistarh et al. [5] for arbitrary k), it achieves plurality consensus with probability $1 - \mathcal{O}(1)$ in parallel time $\mathcal{O}(\log n)$ and uses $\mathcal{O}(k \cdot \log(1/\alpha)) = \mathcal{O}(k \cdot \log n)$ bits per node (Corollary 3.3.2), an improvement by a $\log(n)$ factor.

3.2 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 where λ_2 is the second largest eigenvalue. 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 $\alpha \coloneqq \frac{n_1 - n_2}{n} \in [1/n, 1]$ denotes the *initial bias* towards the plurality opinion.

The objective of *plurality consensus problem*, 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. Further to the running time we also 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 without loss of generality (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 $\boldsymbol{M} = (\boldsymbol{M}_t)_{t \leq N}$ 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:* In every round t , all edges of the graph are 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. Note that there are several simple, distributed protocols to obtain such matchings Boyd et al. [27] and Ghosh and Muthukr-

ishnan [57]. Results for the random matching model dependent on $p_{\min} \coloneqq \min_{t\in\mathbb{N},\{u,v\}\in E} \Pr(\boldsymbol{M}_t[u,v]=1)$.

- *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 $t \geq d$ we have $\boldsymbol{M}_t = \boldsymbol{M}_{t \bmod d}$.

- *Sequential Model:* In each round t an edge $\{u,v\} \in E$ is activated uniformly random.

Notation. Before we proceed, we restate the following notation that will be used in subsequent sections. 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 $\operatorname{disc}(\boldsymbol{x}) \coloneqq \max_i x(i) - \min_i x(i)$. For $i \in \mathbb{N}$, we define $[i] \coloneqq \{1,2,\ldots,i\}$ 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 $d_t(u) \coloneqq \sum_v \boldsymbol{M}_t[u,v]$ 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) neighbourhood respectively. Moreover, $\Delta \coloneqq \max_{t,u} d_t(u)$ is the maximum active degree of any node. Recall, We say an event happens with high probability (w.h.p.) if its probability is at least $1 - 1/n^c$ for $c \in \mathbb{N}$ (See Definition 1.1.7).

Smoothing Property. The running time of our protocols is closely related to the running 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 Avin et al. [11] and Sauerwald and Sun [91]). More exactly, we consider a random walk on G that is restricted to the active edges in each time step. As indicated in Section 3.1.2, 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 $\boldsymbol{M}_t$:

$$\boldsymbol{P}_t[u,v] \coloneqq \begin{cases} \frac{1}{2\Delta} & \text{if } \boldsymbol{M}_t[u,v] = 1 \text{ and } u \neq v, \\ 1 - \frac{d_t(u)}{2\Delta} & \text{if } \boldsymbol{M}_t[u,v] = 1 \text{ and } u = v, \\ 0 & \text{if } \boldsymbol{M}_t[u,v] = 0. \end{cases} \tag{3.1}$$

$\boldsymbol{P}_t$ 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 the diffusion model on a d-regular graph. We are now ready to define the required smoothing property.

Definition 3.2.1 (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* $\operatorname{disc}(\boldsymbol{x} \cdot \prod_{t=t_1}^{t_2} \boldsymbol{P}_t) \leq \varepsilon$*. Moreover, we define the* mixing time $t_{mix}(\varepsilon)$ *as the smallest number of steps such that any time window of length $t_{mix}(\varepsilon)$ is ε-smoothing. That is, $t_{mix}(\varepsilon) \coloneqq \min\{ t' \mid \forall t \in \mathbb{N} \colon [t, t+t'] \text{ is } \varepsilon\text{-smoothing} \}$.*

The mixing time can be seen as the worst-case time required by a random walk to get "close" to the uniform stationary distribution. If the parameter ε is not explicitly stated, we consider $t_{\text{mix}} \coloneqq t_{\text{mix}}(n^{-5})$.

3.3 Protocol BALANCE

We begin by analysing our Balance protocol. Our Balance protocol is inspired by load balancing and can be thought of as a (slightly simplified) version of Alistarh et al. [5] or Kempe et al. [67] that generalizes naturally to $k \geq 2$ and arbitrary (even dynamic) graphs.

```
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 3.1: Protocol BALANCE as executed by node u at time t. At time zero, each node initializes $\ell_{o_u,0}(u) \coloneqq \gamma$ and $\ell_{j,0}(u) \coloneqq 0$ for all $j \neq o_u$.

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 $\boldsymbol{M} = (\boldsymbol{M}_t)_{t \leq N}$ and the corresponding transition matrices $\boldsymbol{P}_t$ (cf. Section 3.2). 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, our protocol does not send fractional loads over active edges. This avoids the overheads that are necessary for nodes to handle floating point operations. To this end, we use a rounding scheme from Berenbrink et al. [18] and Sauerwald and Sun [91] to restrict the protocol to integer loads. The round scheme 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 neighbours 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 neighbour v receives a token with probability $\boldsymbol{P}_t[u, v]$. In the following we refer to the balancing algorithm using this rounding scheme as the VERTEX-BASED BALANCER algorithm.

The formal description of protocol BALANCE is given in Listing 3.1.

Analysis of Balance.

Consider initial load vectors $\boldsymbol{\ell_0}$ with $\|\boldsymbol{\ell_0}\|_\infty \leq n^5$. Let $\tau \coloneqq \tau(g, \boldsymbol{M})$ 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. More formally,

$$\tau \coloneqq \tau(g, \boldsymbol{M}) \coloneqq \min_t \{\mathrm{disc}(\boldsymbol{\ell}_t) \leq g\}$$

With this, we show:

Theorem 3.3.1. *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. By our definition, $\gamma \geq 3\frac{g}{\alpha} = 3g \cdot \frac{n}{n_1 - n_2}$ and for $i \in [k]$ let $\bar{\ell}_i \coloneqq n_i \cdot \gamma / n$. $\bar{\ell}_i$ denotes the average number of tokens for opinion i. The definition of $\tau(g, \boldsymbol{M})$ implies $\ell_{1,\tau}(u) \geq \bar{\ell}_1 - g$ and $\ell_{i,\tau}(u) \leq \bar{\ell}_i + g$ for all nodes u and $i \geq 2$.

Consequently, we get

$$\ell_{1,t}(u) - \ell_{i,t}(u) \geq \bar{\ell}_1 - \bar{\ell}_i - 2g = 3g \cdot \frac{n_1 - n_i}{n_1 - n_2} - 2g > 0. \tag{3.2}$$

Thus, every node u has the correct plurality guess at time τ. □

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 3.3.2. *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 $\mathcal{O}(k \cdot \log n)$ *bits of memory in time* $\mathcal{O}\left(\frac{\log n}{1-\lambda_2}\right)$ *in the diffusion model,*

(b) using $\mathcal{O}(k \cdot \log n)$ *bits of memory in time* $\mathcal{O}\left(\frac{1}{d \cdot p_{min}} \cdot \frac{\log n}{1-\lambda_2}\right)$ *in the random matching model,*

(c) using $\mathcal{O}(k \cdot \log(\alpha^{-1}))$ *bits of memory in time* $\mathcal{O}\left(d \cdot \frac{\log n}{1-\lambda_2}\right)$ *in the balancing circuit model, and*

(d) using $\mathcal{O}(k \cdot \log(\alpha^{-1}))$ *bits of memory in time* $\mathcal{O}\left(n \cdot \frac{\log n}{1-\lambda_2}\right)$ *in the sequential model.*

Proof. Part (a) follows directly from [92, Theorem 6.6] and Part (c) follows directly from [92, 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 [92, Theorem 1.1]. □

3.4 Protocol SHUFFLE

Given that the memory requirements of the Balance protocol become impractical for a large number of opinions we now present a different protocol that we will refer to as Shuffle. Our main result is the following theorem, stating the correctness as well as the time and space-efficiency of Shuffle. In particular, our result shows that our Shuffle protocol allows for a trade off between time and space. The protocol is described in Section 3.4.1, followed by its analysis in Section 3.4.2.

Theorem 3.4.10. *Let* $\alpha = \frac{n_1 - n_2}{n} \in [1/n, 1]$ *denote 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* $\mathcal{O}(T \cdot t_{mix})$ *rounds (w.h.p.) and requires* $\mathcal{O}(\gamma \cdot \log(k) + \log(T \cdot t_{mix}) + \log(\gamma \cdot T))$ *memory bits per node for* $\gamma \geq \frac{c \cdot \log n}{\alpha^2 T} + 2c'\Delta \geq 2\Delta$.

The parameter T in the statement serves as a lever to trade running time for memory. Since the number of tokens γ depends on the maximum number of commu-

nication partners Δ, for communication patterns where Δ is "large" e.g., diffusion, there can be a negative impact on the space requirements for our SHUFFLE protocol. This additional requirement is due to the number of tokens on each node being invariant. In these cases, our BALANCE protocol might be more appropriate.

Since t_{mix} depends on the graph and communication pattern, Theorem 3.4.10 might look a bit unwieldy. The following corollary gives a few concrete examples for common communication patterns on general graphs.

Corollary 3.4.1. *Let G be an arbitrary d-regular graph.* SHUFFLE *ensures that all nodes agree on the plurality opinion (w.h.p.) using*

$$\mathcal{O}\left(\frac{\log(n)}{\alpha^2 T}\cdot\log(k)+\log(T\cdot t_{mix})+\log\left(T\cdot\frac{\log(n)}{\alpha^2 T}\right)\right)$$

bits of memory in time

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

3.4.1 Protocol Description

We continue to explain the SHUFFLE protocol given in Listing 3.2.

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 labelled with u's opinion o_u. Our protocol sends $\frac{\gamma}{2\Delta}$ tokens chosen uniformly at random (without replacement) over each edge $\{u, v\} \in E$ where $\boldsymbol{M}_t[u, v] = 1$ and $\Delta \coloneqq \max_{t,u} d_t(u)$ is the maximum active degree of any node. Here, $\gamma \geq 2\Delta$ is a parameter depending on T and α to be fixed during the analysis [5] such that 2Δ divides γ. SHUFFLE maintains the invariant that, at any time, all nodes have exactly γ tokens. It is this invariance that

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

Listing 3.2: Protocol SHUFFLE as executed by node u at time t. At time zero, each node u creates γ tokens labelled o_u and sets $c_u := 0$ and $(dom_u, e_u) := (o_u, c_u)$.

introduces the dependence on the communication pattern. More specifically on the maximum number of communication partners Δ. For static graphs, the maximum number of communication partners Δ is the maximal degree which can be easily computed in a distributed way, see for example Boyd et al. [27].

In addition to storing the tokens, each node maintains a set of auxiliary variables. The variable c_u is increased during the update part of the protocol and counts tokens labelled o_u. The variable pair (dom_u, e_u) is a temporary guess of the plurality opinion and its frequency. During the broadcast part of the protocol, 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 of the protocol 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. 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 $\tau \leq t_{\text{mix}}$. Consequently, at time t' all nodes have the correct majority opinion. 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 14), 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.

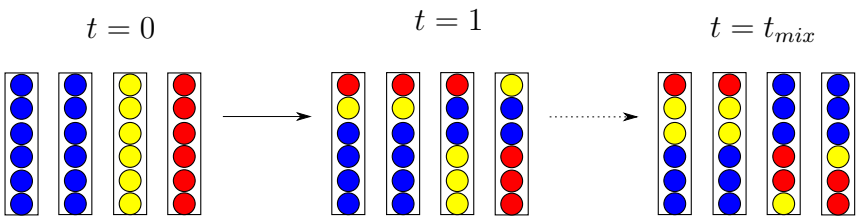

Figure 3.1: Plurality Consensus: Example of the SHUFFLE protocol for $G = K_4$, $\gamma = 6$, and $k = 3$ using Diffusion

In order to emphasise the intuition behind our SHUFFLE protocol we consider the following example for $n = 4$. Assume that the underlying network is a complete graph and that nodes communicate with all neighbours in a given round i.e., Diffusion. Figure 3.1 shows possible configurations after $t = 1$ and $t = t_{mix}$ time steps. At $t = 0$, all nodes generate $\gamma = 2\Delta = 6$ tokens labelled their initial opinion (red, blue, or yellow). In each subsequent round $t > 0$, each node sends $\frac{\gamma}{2\Delta} = 1$ tokens chosen u.a.r to each of its neighbours. Note the number of tokens on each node is invariant.

Since the communication pattern satisfies the smoothing property, after $t = t_{mix}$ time steps the configuration resembles the uniform stationary distribution. However, we must be aware of the discrepancy that occurs due to the integrality of tokens.

After t_{mix} time steps, each node u updates its counter (c_u) for the number of tokens that it has seen of its original opinion (o_u) by the number of tokens currently on the node labelled with that opinion. The counter c_u is then broadcast in subsequent steps so that all nodes are informed of the maximum counter amongst all of the nodes. By only updating after t_{mix} time steps, this value is an estimate of $\gamma n_i/n$ where n_i is the number of nodes that initially held opinion i.

3.4.2 Analysis of Shuffle

Fix a communication pattern $(\boldsymbol{M}_t)_{t\leq N}$ and an arbitrary parameter $T \in \mathbb{N}$. Remember that $t_{\text{mix}} \coloneqq t_{\text{mix}}(n^{-5})$ denotes the smallest number such that any time window of length t_{mix} is n^{-5}-smoothing under $(\boldsymbol{M}_t)_{t\leq N}$. The number of tokens γ stored on each node will be set during the analysis.

The analysis of Shuffle is largely based on Lemma 3.4.9, which states that, after $\mathcal{O}(T \cdot t_{\text{mix}})$ 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 dependency between the tokens' movements, rendering standard Chernoff bounds inapplicable. Instead, we show that certain random variables satisfy the negative regression condition (Lemma 3.4.4), which allows us to majorize the token distribution by a random walk (Lemma 3.4.5) and to derive the Chernoff type bound (Lemma 3.4.8). This Chernoff type bound can be used to show that all counter values are concentrated which is the main pillar of the proof of our main result (Theorem 3.4.10).

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 of our protocol at time t. To ease the discussion, we consider u as a neighbour of itself and speak of $d_t(u)+1$ neighbours. For $i \in [d_t(u)+1]$, let $N_t(u,i) \in V$ denote the i-th neighbour 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 neighbour. 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 neighbours token j is sent. Our key technical lemma (Lemma 3.4.4) establishes the *negative regression condition* for these $(X_j)_{j\in[\gamma n]}$ variables. Negative regression is defined as follows:

Definition 3.4.2 (Neg. Regression Dubhashi and Ranjan [46, Def. 21])**.** *A vector $(X_1, X_2, \ldots, X_n)$ of random variables is said -to satisfy the* negative regression *condition if*

$$\mathbb{E}\left[f(X_l, l \in \mathscr{L}) \mid X_r = x_r, r \in \mathscr{R}\right]$$

is non-increasing in each x_r for any disjoint $\mathscr{L}, \mathscr{R} \subseteq [n]$ and for any non-decreasing function f.

The following lemma states a useful property for random variables that satisfy the negative regression condition (Definition 3.4.2).

Lemma 3.4.3 (Dubhashi and Ranjan [46, Lemma 26])**.** *Let $(X_1, X_2, \ldots, X_n)$ 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*

$$\mathbb{E}\left[\prod_{i\in I} f_i(X_i)\right] \leq \prod_{i\in I} \mathbb{E}\left[f_i(X_i)\right] \tag{3.3}$$

We now show that the random variables $(X_j)_{j\in[\gamma n]}$ that denote the movement of the tokens in a single round satisfy the negative regression condition.

Lemma 3.4.4 (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 the neighbour of u that token j is sent in the next step. We show for any $u \in V$ that $(X_j)_{j:\, u_j=u}$ satisfies the negative regression condition (NRC). The lemma's statement follows since the permutation of tokens on each node (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 $\mathscr{L}, \mathscr{R} \subseteq \{\, j \in [\gamma n] \mid u_j = u \,\}$ of tokens on u. Define $d \coloneqq d(u)$ to be the degree of node u and let $f\colon [d+1]^{|\mathscr{L}|} \to \mathbb{R}$ be an arbitrary non-decreasing function. It remains to show that $\mathbb{E}\left[f(X_l, l \in \mathscr{L}) \mid X_r = x_r, r \in \mathscr{R}\right]$ is non-increasing in each x_r (cf. Definition 3.4.2). That is, we need

$$\mathbb{E}\left[f(X_l, l \in \mathscr{L}) \mid X_r = x_r, r \in \mathscr{R}\right] \leq \mathbb{E}\left[f(X_l, l \in \mathscr{L}) \mid X_r = \tilde{x}_r, r \in \mathscr{R}\right], \tag{3.4}$$

where $x_r = \tilde{x}_r$ holds for all $r \in \mathscr{R} \setminus \{\, \hat{r} \,\}$ and $x_{\hat{r}} > \tilde{x}_{\hat{r}}$ for a fixed index $\hat{r} \in \mathscr{R}$.

We prove Inequality (3.4) via a coupling of the processes on the left-hand side (LHS process) and right-hand side (RHS process) of that inequality. The two processes differ for a token $\hat{r}$ that is sent to different neighbours in the next step. 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., $j \in [\gamma n] \setminus \mathscr{R}$) one by one in order of increasing indices. To ease the description assume (w.l.o.g.) that the tokens from $\mathscr{R}$ are numbered from 1 to $|\mathscr{R}|$. When we reveal the j-th variable (which indicates 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 $T_{j,i}(\boldsymbol{x}) \coloneqq \sum_{i' \leq i} p_{j,i'}(x_{i'})$ for each $i \in [d+1]$. The vector $\boldsymbol{x}$ describes the number of tokens allocated to each of the $d+1$ slots. To define our coupling, let $\beta_{j,i} \coloneqq |\{\, j' < j \mid X_{j'} = i \,\}|$ denote the number

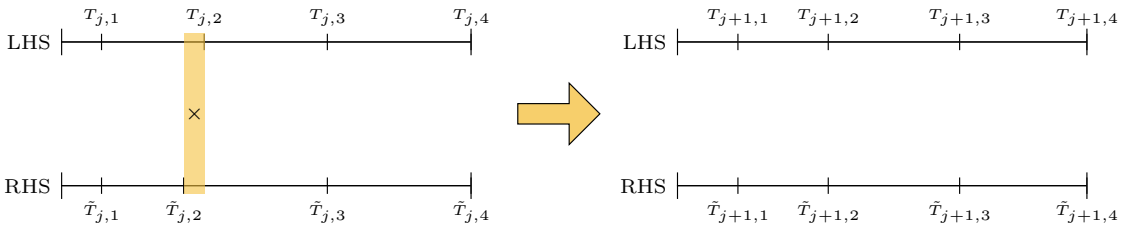

Figure 3.2: Illustration showing the $d+1=4$ different slots for the LHS and RHS process and how they change. In this example, $x_{\hat{r}}=3$ and $\tilde{x}_{\hat{r}}=2$. On the left, the uniform random variable U_j falls into slot $[T_1,T_2)$ for the LHS process (causing j to be sent to node $N(u,2)$) and into slot $[\tilde{T}_2,\tilde{T}_3)$ for the RHS process (causing j to be sent to node $N(u,3)$).

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. In other words, $\beta_{j,i}$ and $\tilde{\beta}_{j,i}$ describe the number of tokens sent to the i'th neighbour prior to j. We use $\boldsymbol{\beta_j}, \tilde{\boldsymbol{\beta}}_{\boldsymbol{j}} \in \mathbb{N}^{d+1}$ 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 $\tilde{T}_{j,i} \coloneqq T_{j,i}(\tilde{\boldsymbol{\beta}}_{\boldsymbol{j}})$ 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 3.2 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 $r \in \mathscr{R}$ are set, and these differ in the LHS and RHS process only for the index $\hat{r} \in \mathscr{R}$, 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 3.2). 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 neighbour in the LHS

process). This implies $\boldsymbol{\beta_{j+1}} = \tilde{\boldsymbol{\beta}}_{\boldsymbol{j+1}}$ and both processes behave identical from now on. Otherwise, if $U_i \notin [\tilde{T}_{j,\iota-1}, T_{j,\iota-1})$, we have $\boldsymbol{\beta_{j+1}} - \tilde{\boldsymbol{\beta}}_{\boldsymbol{j+1}} = \boldsymbol{\beta_j} - \tilde{\boldsymbol{\beta}}_{\boldsymbol{j}}$ and we can repeat the above argument. Thus, after all X_j are revealed, there is at most one $j \in \mathscr{L}$ 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 (3.4). 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 neighbouring. Formally, this merely changes in which order we consider the neighbours 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 3.4.5) we require further notation. Let $\mathscr{S}$ denote our random SHUFFLE process, and $\mathscr{W}$ 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 $\mathbf{P}_t[u,\cdot]$ for the transition probabilities). We use $w_j^{\mathscr{P}}(t)$ to denote the position of token j after t steps of a process $\mathscr{P}$. We assume (w.l.o.g.) $w_j^{\mathscr{S}}(0) = w_j^{\mathscr{W}}(0)$ for all j. While there are strong correlations between the tokens' movements in $\mathscr{S}$ (e.g., not all tokens can move to the same neighbour), Lemma 3.4.5 shows that these correlations are negative.

Lemma 3.4.5 (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:*

(a) $\Pr\left(w_j^{\mathscr{S}}(t) = v\right) = \Pr\left(w_j^{\mathscr{W}}(t) = v\right)$ *and*

(b) $\Pr\left(\bigcap_{j\in B}\left(w_j^{\mathscr{S}}(t) \in D\right)\right) \leq \Pr\left(\bigcap_{j\in B}\left(w_j^{\mathscr{W}}(t) \in D\right)\right) = \prod_{j\in B}\Pr\left(w_j^{\mathscr{W}}(t) \in D\right)$.

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 $\mathscr{W}$. To show the inequality, we define the intermediate process $\mathscr{SW}(t')$ $(t' \leq t)$ that performs t' steps of $\mathscr{S}$ followed by $t - t'$ steps of $\mathscr{W}$. By this definition, $\mathscr{SW}(0)$ is identical to $\mathscr{W}$

restricted to t steps and, similar, $\mathscr{SW}(t)$ is identical to $\mathscr{S}$ restricted to t steps. Define

$$\mathscr{E}_{t'} \coloneqq \bigcap_{j \in B} \left(w_j^{\mathscr{SW}(t')}(t) \in D \right) \tag{3.5}$$

(the event that all tokens from B end up at nodes from D under process $\mathscr{SW}(t')$). The lemma's statement is equivalent to $\Pr(\mathscr{E}_t) \leq \Pr(\mathscr{E}_0)$. To prove this, we show $\Pr(\mathscr{E}_{t'+1}) \leq \Pr(\mathscr{E}_{t'})$ 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 $\mathscr{SW}(t')$ and $\mathscr{SW}(t'+1)$ 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 neighbour ends up in a node from D. Formally,

$$h_j(i) \coloneqq \Pr\left(w_j^{\mathscr{W}}(t) \in D \;\middle|\; w_j^{\mathscr{W}}(t'+1) = N(u_j, i) \right). \tag{3.6}$$

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

Now, by Lemma 3.4.4 the variables $(X_j)_{j \in B}$ satisfy the negative regression condition. Thus, we can apply Lemma 3.4.3 (a well-known characterization of negative regression) to the functions h_j. Using another simple auxiliary result (Claim 3.4.6) we can relate the (conditioned) probabilities of the events $\mathscr{E}_{t'}$ and $\mathscr{E}_{t'+1}$ to the expectations over the different $h_j(X_j)$. That is, for $p \coloneqq \Pr(\mathscr{E}_{t'+1} \mid c(t') = c)$ we compute

$$\begin{aligned} p &\overset{\textit{Clm. } 3.4.6.(a)}{=} \mathbb{E}\left[\prod_{j \in B} h_j(X_j) \;\middle|\; c(t') = c \right] \\ &\overset{\textit{Lem. } 3.4.3}{\leq} \prod_{j \in B} \mathbb{E}\left[h_j(X_j) \mid c(t') = c \right] \\ &\overset{\textit{Clm. } 3.4.6.(b)}{=} \Pr(\mathscr{E}_{t'} \mid c(t') = c). \end{aligned}$$

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

Claim 3.4.6. *Fix a time $t' \in \{0, 1, \ldots, t-1\}$ and consider an arbitrary configuration c. Then the following identities hold:*

(a) $\Pr(\mathscr{E}_{t'+1} \mid c(t') = c) = \mathbb{E}\left[\prod_{j \in B} h_j(X_j) \;\middle|\; c(t') = c\right]$, *and*

(b) $\Pr(\mathscr{E}_{t'} \mid c(t') = c) = \prod_{j \in B} \mathbb{E}[h_j(X_j) \mid c(t') = c]$.

Proof. Remember the definitions from Lemma 3.4.5 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 $d(u_j)+1$ neighbours of u_j (where u_j is considered a neighbour 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 $\Pr(c(t'+1) = c_{\boldsymbol{x}} \mid c(t') = c) = \Pr(\boldsymbol{X} = \boldsymbol{x} \mid c(t') = c)$, and conditioning on $c(t'+1)$ is equivalent to conditioning on $\boldsymbol{X}$ and $c(t')$. For the claim's first statement, we calculate

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

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 $\mathscr{SW}(t'+1)$ 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 neighbour reaches a node from D.

For the claim's second statement, we do a similar calculation for the process $\mathscr{SW}(t')$. 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

With Lemma 3.4.5 we are ready derive a Chernoff bound. In particular we show that we can apply a Chernoff bound that will allow us to show in Lemma 3.4.9 that with high probability nodes are able to distinguish the plurality opinion.

The following standard lemma is used in the proof of Lemma 3.4.8(Token Concentration).

Lemma 3.4.7 (Azar et al. [12, 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 $\Pr(Y_i = 1 \mid X_1, \ldots, X_{i-1}) \leq p$, then*

$$\Pr\left(\sum Y_i \geq \ell\right) \leq \Pr\left(\mathrm{Bin}(n,p) \geq \ell\right) \tag{3.7}$$

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

$$\Pr\left(\sum Y_i \leq \ell\right) \leq \Pr\left(\mathrm{Bin}(n,p) \leq \ell\right). \tag{3.8}$$

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 3.4.8 (Token Concentration)**.** *Consider any subset B of tokens, a node $u \in V$, and an integer T. Let $X \coloneqq \sum_{t \leq T} \sum_{j \in B} X_{j,t}$, 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 $\Pr(X \geq (1+\delta)\cdot\mu) \leq e^{\delta^2 \mu/3}$.*

Proof. Let $v_{j,t}$ denote the location of token j at time $(t-1) \cdot t_{\text{mix}}$. For all $t \leq T$ 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 3.4.5 we have for each $B' \subseteq B$ and $t \leq T$ that

$$\Pr\left(\bigcap_{i \in B'} X_{j,t} = 1\right) \leq \prod_{j \in B'} \Pr\left(Y_{j,t} = 1\right). \tag{3.9}$$

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

$$\Pr\left(X \geq \ell\right) = \Pr\left(\sum_{t \leq T} \sum_{j \in B} X_{j,t} \geq \ell\right) \leq \Pr\left(\sum_{t \leq T} \sum_{j \in B} Y_{j,t} \geq \ell\right). \tag{3.10}$$

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

$$\Pr\left(Y_{j,t} = 1 \;\middle|\; Y_{1,1}, Y_{2,1}, \ldots, Y_{|B|,1}, Y_{1,2}, \ldots, Y_{j-1,t}\right) \leq p. \tag{3.11}$$

Combining our observations with Lemma 3.4.7 (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 (Lemma 1.1.9) we get

$$\Pr\left(X \geq (1+\delta)\mu\right) \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \leq e^{-\delta^2 \mu/3}, \tag{3.12}$$

which yields the desired statement. □

Together, these lemmas generalize a result given in Sauerwald and Sun [91] to a setting with considerably more dependencies. Equipped with this Chernoff bound, we prove concentration of the counter values.

Lemma 3.4.9 (Counter Separation)**.** *Let $c \geq 16$. For every time $t \geq c \cdot T \cdot t_{mix}$ there exist values $\ell_\top > \ell_\perp$ such that*

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

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

for $\gamma \geq \frac{c \cdot \log n}{\alpha^2 T}$

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

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

We set $\ell_\perp \coloneqq \ell_\perp(2)$. We first confirm that for a suitable choice of γ that $\ell_\top > \ell_\perp$

$$\begin{aligned}
\ell_\top - \ell_\perp &= cT\gamma - \mu' - \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \frac{n - n_1}{n}} - \mu_2 - \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \frac{n_2}{n}} \\
&= cT\gamma \left(\frac{n_1 - n_2}{n} - \frac{1}{n^4} + \frac{n_1 - n_2}{n^5} \right) - \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma} \left(\sqrt{\frac{n - n_1}{n}} + \sqrt{\frac{n_2}{n}} \right)
\end{aligned}$$

Since $\alpha \coloneqq \frac{n_1 - n_2}{n} \in [1/n, 1]$,

$$\geq cT\gamma \left(\alpha - \frac{1}{n^4} \right) - \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma} \left(\sqrt{\frac{n - n_1}{n}} + \sqrt{\frac{n_2}{n}} \right)$$

For $n \geq 2$,

$$\begin{aligned}
&\geq cT\gamma \left(\frac{\alpha}{2} \right) - \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma} \left(\sqrt{\frac{n - n_1}{n}} + \sqrt{\frac{n_2}{n}} \right) \\
&\geq \sqrt{cT\gamma} \left(\sqrt{cT\gamma} \frac{\alpha}{2} - 2\sqrt{c \cdot \log n} \right)
\end{aligned}$$

For the claim to hold we require that the second term must be positive,

$$\implies \sqrt{cT\gamma} \frac{\alpha}{2} - 2\sqrt{c \cdot \log n} > 0$$

Finally solving for γ,

$$\implies \sqrt{\gamma} > 4\sqrt{\frac{\log n}{T\alpha^2}} \implies \gamma > \frac{16 \cdot \log n}{T \cdot \alpha^2}$$

It follows that $\ell_\top > \ell_\perp$ for a suitable choice of $\gamma > \frac{16 \cdot \log n}{T\alpha^2}$.

We now show the two statements of the lemma. Let all γn tokens be labelled from 1 to γn.

We begin by showing statement (a). Consider a node w with $o_w \geq 2$ and set $\lambda(o_w) \coloneqq \ell_\perp(o_w) - \mu_{o_w} = \sqrt{c^2 \cdot \log n \cdot T \cdot \gamma \cdot n_{o_w}/n}$. 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 $c_w = \sum_{j=1}^{c \cdot T} \sum_{i \in B} X_{i,j \cdot t_{\text{mix}}}$ where B is the subset of tokens with opinion o_w We compute

$$\begin{aligned} \Pr\left(c_w \geq \ell_\perp\right) &\leq \Pr\left(c_w \geq \mu_{o_w} + \lambda(o_w)\right) = \Pr\left(c_w \geq \left(1 + \frac{\lambda(o_w)}{\mu_{o_w}}\right) \cdot \mu_{o_w}\right) \\ &\leq \exp\left(-\frac{\lambda^2(o_w)}{3\mu_{o_w}}\right) \leq \exp\left(-\frac{c}{6}\log n\right), \end{aligned} \tag{3.13}$$

where the last line follows by Lemma 3.4.8 applied to $c_w = \sum_{j=1}^{c \cdot T} \sum_{i \in B} X_{i,j \cdot t_{\text{mix}}}$ 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 statement (b), consider a node v with $o_v = 1$ and set $\lambda' \coloneqq \mu' - \ell_\top$. 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 $Y = \sum_{j=1}^{c \cdot T} \sum_{i \in B'} Y_{i,j \cdot t_{\text{mix}}}$ where B' is the subset of tokens with an opinion other than 1. Note that $c_v = cT\gamma - Y$. We compute

$$\begin{aligned} \Pr\left(c_v \leq \ell_\top\right) &= \Pr\left(cT\gamma - Y \leq \ell_\top\right) = \Pr\left(cT\gamma - Y \leq cT\gamma - \mu' - \lambda'\right) \\ &= \Pr\left(Y \geq \mu' + \lambda'\right) = \Pr\left(Y \geq \left(1 + \frac{\lambda'}{\mu'}\right) \cdot \mu'\right) \\ &\leq \exp\left(-\frac{\lambda'^2}{3\mu'}\right) \leq \exp\left(-\frac{c}{6}\log n\right), \end{aligned}$$

where the first inequality follows by Lemma 3.4.8 applied to Y. Hence, the claim follows for c large enough after taking the union bound over all $n_1 \leq n$ nodes v with

$o_u \geq 2$. □

We now give the proof of our main theorem.

Theorem 3.4.10. *Let $\alpha = \frac{n_1 - n_2}{n} \in [1/n, 1]$ denote 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 $\mathcal{O}(T \cdot t_{mix})$ rounds (w.h.p.) and requires $\mathcal{O}(\gamma \cdot \log(k) + \log(T \cdot t_{mix}) + \log(\gamma \cdot T))$ memory bits per node for $\gamma \geq \frac{c \cdot \log n}{\alpha^2 T} + 2c'\Delta \geq 2\Delta$.*

Proof. 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 3.4.9. From Lemma 3.4.9 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 9) 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 $\tau \leq t_{\text{mix}}$. 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 $[k]$ ($\gamma \cdot \mathcal{O}(\log k)$ bits), three opinions (its own, its plurality guess, and the dominating opinion; $\mathcal{O}(\log k)$ 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 $\mathcal{O}(\log(\gamma T))$. Finally, the time step counter is bounded by $\mathcal{O}(\log(T \cdot t_{\text{mix}}))$ bits. This yields the claimed space bound.

Note that our choice of γ is dependent on both our concentration measures (Lemma 3.4.8) and the communication pattern being considered. In order to apply Lemma 3.4.8 we require that γ is suitably large. It follows from our proof that

$\gamma \geq \frac{c \cdot \log(n)}{\alpha^2 T}$ is sufficient. Additionally, since each node sends 2Δ tokens over each active edge, we require that $\gamma \geq 2\Delta^2$. This follows from γ being invariant. We therefore choose γ such that

$$\gamma \geq \frac{c \cdot \log n}{\alpha^2 T} + 2c'\Delta \geq 2\Delta$$

for some $c' \geq 0$ □

3.5 Conclusion

We have presented two protocols (Balance and Shuffle) that solve the plurality consensus problem on arbitrary graphs for any initial bias and a diverse set of natural communication patterns that satisfy a smoothing property. Both our protocols are inspired by load balancing. The Balance protocol can be thought of as a (slightly simplified) version of Alistarh et al. [5] or Kempe et al. [67] that generalises to arbitrary graphs and $k \geq 2$ opinions. This protocol is particular suitable when the number of opinions is small since the space required by depends on the number of opinions. Our Shuffle protocol does not share this dependency and is therefore more appropriate when the number of opinions is large. The protocol allows for a trade off between running time and memory requirements. However, the protocol may be impractical when the number of communication partners is large.

Using the terminology of Ghaffari and Parter [56] both our protocols can be referred to as "reading protocols". Ghaffari and Parter [56] suggest that, "Generally, it seems implausible that reading style algorithms would lead to a plurality algorithm for the random gossip model with polylogarithmic size messages and polylogarithmic time complexity". Our protocols show that this is in fact possible for a range of parameters. For example, our Balance protocol solves the plurality consensus problem on arbitrary graphs for any initial bias using polylogarithmic space and time when the number of opinions $k = \mathcal{O}(\log n)$. This dependence on the number of opinions is removed by our Shuffle protocol. When the maximum number of com-

munication partners Δ is small and for a sufficiently high bias $\alpha = \Omega\left(\frac{1}{\text{polylog}(n)}\right)$, the SHUFFLE protocol also solves the plurality consensus problem using polylogarithmic space in polylogarithmic time.

Chapter 4

Restricted PULL

In this chapter we study the effect of a modified communication pattern on the performance of algorithms for rumour spreading. We study a restricted version of the classical PULL algorithm (RPULL). Through demonstrating a relationship between our RPULL algorithm and the classical PUSH algorithm on d-regular graphs we are able to show that the broadcast time of RPULL can be upper bounded by the broadcast time of PUSH. Using this relationship we extend our results to arbitrary graphs. In particular we are able to show that RPULL is an optimal algorithm for a number of graph classes.

4.1 Introduction

Rumour spreading (broadcast) is a fundamental task in distributed computing. Initially a single source node knows a rumour that must be spread to all other nodes in the network. Examples of applications include maintaining consistency in replicated database systems [40], data aggregation problems [30, 67, 78], understanding social networks [33], and as a subroutine for running arbitrary distributed computations [29].

One of the most common models for the study of the rumour spreading problem is the *random phone call* model [66]. Consider a graph $G = (V, E)$ where $|V| = n$ and a source node $u \in V$ with a message that must be disseminated to all other

nodes of the graph. In synchronous rounds, each node of the graph initiates contact with a single neighbour chosen at random. In any round a node is said to be either *informed* or *uninformed*. Depending on the state of the node initiating contact a distinction between two types of operations is made. Nodes that are informed *push* the message to the neighbour they initiate contact with. Conversely, nodes that are uninformed attempt to *pull* the message from their chosen neighbour. When used exclusively these two approaches are referred to as the PUSH and PULL algorithms respectively. The combination of the two algorithms is known as the PUSH-PULL algorithm.

The suitability of each of the above algorithms may be application dependent. For example, the PUSH algorithm is particularly suitable when updates or the injection of new messages are infrequent. However, a drawback of the PUSH algorithm is that it is known to create a large message overhead [66]. On the complete graph, PUSH requires $\Theta(\log n)$ rounds and $\Theta(n \log n)$ message transmissions [40, 85]. On the other hand, Karp et al. [66] show that using the combination of PUSH and PULL spreads the rumour in $\mathcal{O}(\log n)$ rounds with $\mathcal{O}(n \log \log n)$ message transmissions. Berenbrink et al. [22] show a similar result for random d-regular graphs. The authors present an algorithm that uses both PUSH and PULL operations that has time complexity $\mathcal{O}(\log n)$ and uses $\mathcal{O}(n \log \log n)$ transmissions. This demonstrates a benefit of the PUSH-PULL algorithm. This is further demonstrated when we consider the broadcast time of PUSH-PULL on arbitrary graphs in comparison to either PUSH or PULL alone. For example, either PUSH or PULL alone require $\Omega(n)$ rounds on a Star graph. In contrast, the PUSH-PULL algorithm requires $\Theta(1)$ rounds. In this example the improvement relies on the ability of nodes to participate in more than one interaction per round. In particular, consider the centre node of a star graph. The centre node might be contacted by all of its $n-1$ neighbours. If the centre node knows the rumour then all nodes will learn the rumour after a single round of PULL. When the nodes are computationally limited or capable of limited communication this might not be feasible.

In order to obtain scalable solutions it is therefore desirable to limit the number of interactions that a node participates in. This limitation has been the subject of recent papers by Daum et al. [39], Ghaffari and Newport [55] and Kiwi and Caro [69]. The authors have proposed variants of the PULL algorithm that limit the number of interactions a node can participate in.

4.1.1 Related Work

There is an extensive body of literature for the basic rumour spreading process. Early results consider the PUSH algorithm on the complete graph. For the complete graph, the PUSH algorithm spreads the rumour to all nodes in $\mathcal{O}(\log n)$ rounds with high probability. Strong results are given by Frieze and Grimmett [54] and Pittel [85]. Frieze and Grimmett [54] show that with high probability the PUSH algorithm informs all nodes in $(1 + o(1)) \log_2 n + \ln n$ rounds. A tighter result is given by Doerr and Künnemann [43]. The authors show that the number of rounds required by the PUSH algorithm on a complete graph is very closely described by $\log n + \frac{1}{n} \cdot C_n$ where C_n is the completion time of the coupon collector problem with n coupons.

Demers et al. [40] propose the use of randomised rumour spreading algorithms for the maintenance of distributed database systems. Due to the distributed nature of the application, it is natural to analyse the performance of these algorithms on networks. Feige et al. [50] upper bound the broadcast time of the PUSH protocol on general graphs by $\mathcal{O}(n \log n)$ w.h.p.

The result of Feige et al. [50] shows there are graphs where PUSH alone performs poorly in comparison to the performance on the complete graph. Karp et al. [66] show that even on the complete graph PUSH incurs a large message overhead. Karp et al. [66] consider the PUSH-PULL algorithm. This reduces the number of message transmissions required. The authors show that the PUSH-PULL algorithm requires $\Theta(n \ln \ln n)$ messages to inform all nodes in $\Theta(\ln n)$ rounds in comparison to the $\mathcal{O}(n \ln n)$ message transmissions required by PUSH alone.

Subsequently there has been a large volume of work that considers the performance of the PUSH-PULL algorithm. Chierichetti et al. [33] consider the PUSH-PULL algorithm on preferential attachment graphs due to their relevance to models of social networks. The authors prove that the PUSH-PULL algorithm informs all nodes within $\mathcal{O}\left(\log^2 n\right)$ rounds w.h.p where as PUSH or PULL alone require polynomially many rounds. The result for PUSH-PULL was subsequently improved by Doerr et al. [41]. In other cases, random power law graphs are used to model social networks. A random power law graph is a random graph whose degree sequence follows a *power law*. i.e., the number of vertices with degree k is proportional to $k^{-\beta}$. Fountoulakis et al. [53] show for $2 < \beta < 3$ the PUSH-PULL algorithm requires $\Theta\left(\log\log n\right)$ rounds w.h.p. where as for $\beta > 3$ then $\Omega\left(\log n\right)$ rounds are required.

There are studies that suggest that several real world networks exhibit good expansion properties [71, 75]. Bounds for the broadcast time of the PUSH-PULL algorithm are known in terms of graph expansion properties. In particular, there are tight bounds for the broadcast time of the PUSH-PULL algorithm in terms of both *conductance* and *vertex expansion*. The vertex expansion, $\alpha = \alpha(G)$, of a graph G is defined for a non-empty set of vertices S as,

$$\alpha(G) := \min_{0<|S|<\frac{n}{2}} \frac{|\partial S|}{|S|} \tag{4.1}$$

where $\partial S = \{u : (u, v) \in E \wedge v \in S \subseteq V \wedge u \notin S\}$ is the outer boundary of S. Giakkoupis [59] shows that for graphs with vertex expansion at least α, $\mathcal{O}\left(\log^2(n)/\alpha\right)$ rounds suffice with high probability for PUSH-PULL. This result is tight in that the bound matches the lower bound given by Giakkoupis and Sauerwald [60]. The conductance $\phi = \phi(G)$ of a graph G is defined as,

$$\phi(G) := \min_{0<vol(S)<\frac{vol(V)}{2}} \frac{|E(S, V - S)|}{vol(S)} \tag{4.2}$$

where $vol(S) = \sum_{v\in S} d_v$ and $E(S, V - S)$ is the set of edges with an endpoint in S

and the other in $V - S$. Giakkoupis [58] shows that for a graph with conductance ϕ, PUSH-PULL spreads the rumour in $O(\phi^{-1} \log n)$ rounds with high probability. This bound matches the lower bound given by Chierichetti et al. [32].

Despite the large volume of work that considers the PUSH-PULL algorithm it is still of interest to consider the performance of the individual algorithms. The suitability of each algorithm depends on the nature of the application. As noted by Karp et al. [66], the PULL algorithm is particularly suited where changes are frequent or PULL operations are being performed due to some other task. The result of Feige et al. [50] states that there are graphs where PUSH alone performs "badly". It is therefore interesting to ask when PUSH or PULL alone is sufficient to spread a rumour quickly. There are both positive and negative results in the literature that explore this direction.

Fountoulakis et al. [51] generalise the result of Frieze and Grimmett [54] for the PUSH algorithm to dense random graphs with degree $\omega(\ln n)$. Fountoulakis and Panagiotou [52] show that PUSH behaves almost identically on random d-regular graphs as on the complete graph. The authors show that w.h.p $(1 + o(1))c_d \ln n$ rounds are sufficient where c_d is a constant that depends on d.

Elsässer and Sauerwald [48] show a relationship between the mixing time of a random walk and the broadcast time of the PUSH algorithm on several Cayley graphs. Sauerwald [89] derives an upper bound on the runtime of the PUSH algorithm of $\mathcal{O}(t_{mix} + \log n)$ where t_{mix} is the mixing time of a certain random walk. When $t_{mix} = \mathcal{O}(\log n)$ this gives an asymptotically optimal bound for the PUSH algorithm. This is the case for several notable graph classes. The author notes that this bound is not tight for Hypercubes and therefore employ separate methods to prove a bound of $\Theta(\log n)$ for Hypercubes w.h.p.

Meier and Peter [73] consider random power law graphs and build on the result of Fountoulakis et al. [53]. Meier and Peter [73] show that for every $\varepsilon > 0$, $\mathcal{O}(\log n)$ rounds are sufficient for PUSH to inform all but a ε-fraction of the nodes with probability $1 - o(1)$. Combined with the results of Fountoulakis et al. [53] this shows

that although PULL asymptotically improves the running time for $2 < \beta < 3$, the same is not true for $\beta > 3$.

The performance of the PUSH algorithm has been studied in terms of the expansion properties of the graph. Sauerwald and Stauffer [90] bound the runtime of PUSH by $\mathcal{O}(\alpha^{-1} \cdot \text{polylog}\, n)$ for regular graphs with vertex expansion α. Despite strong results for the PUSH-PULL algorithm, Sauerwald and Stauffer [90] note that vertex expansion does not guarantee the performance of the PUSH algorithm. For example, consider a complete graph with a single node connected with a single edge. This graph has constant vertex expansion but the PUSH algorithm requires $\Omega(n)$ rounds. For regular graphs, it is known that $\mathcal{O}(\log n/\phi)$ rounds w.h.p where ϕ is the conductance [78]. However, conductance does not guarantee the performance of PUSH or PULL alone on arbitrary graphs.

Despite each node initiating contact with at most one neighbour in the random phone call model, the PULL algorithm might require nodes to participate in multiple interactions per round. This might limit the suitability of the PULL algorithm for applications where the nodes are limited in their communicational abilities. Daum et al. [39] analyse a variant of the PULL algorithm that restricts the number of interactions each node participates in. The authors main result upper bounds the broadcast time of this restricted PULL (RPULL) algorithm in terms of the runtime of the classical PULL algorithm. They show that $\mathcal{O}\left(T_l \cdot \frac{\Delta}{\delta} \log(n)\right)$ rounds are needed w.h.p. where T_l is the runtime of PULL

Ghaffari and Newport [55] formalise an extension of the work by Daum et al. [39]. Ghaffari and Newport [55] call their model the *mobile telephone model*. In the mobile telephone model, each node can participate in at most one interaction per round. In addition to this restriction, the model allows the graph to undergo a bounded rate of change. The authors investigate if the expansion properties of a graph still provide good indicators of how fast a rumour can spread. The authors show that an optimal algorithm terminates in $\mathcal{O}(\alpha^{-1} \cdot \log n)$ rounds. There are however graphs with constant vertex expansion where PUSH-PULL requires $\Omega(\sqrt{n})$ rounds. The

optimal algorithm terminates in $\mathcal{O}\left(\frac{\Delta}{\delta \cdot \phi} \cdot \log n\right)$. There also exists a graph where every algorithm requires $\Omega\left(\frac{\Delta}{\delta \cdot \phi}\right)$ rounds.

4.1.2 Our Contribution

In this chapter we study the RPULL algorithm introduced by Daum et al. [39]. Under the RPULL algorithm each informed node that receives requests is able to reply to exactly one request where tie breaking is performed uniformly at random. We study the relationship between RPULL and PUSH. In doing so we are able to utilise known relationships between PUSH and PULL. Consequently we are able to draw conclusions about the relationship between RPULL and PULL. Daum et al. [39] show that comparing RPULL with PULL is not straight forward. The authors show by means of counter examples that standard approaches such as stochastic dominance and couplings between RPULL and PULL are not possible. This motivates our study of the relationship between RPULL and PUSH. In turn we will then use known relationships between PUSH and PULL to compare RPULL to PULL.

It is the relationship between PUSH and RPULL on d-regular graphs that we exploit in order to extend our results to arbitrary graphs. Our result is shown using a coupling between RPULL and a lazy variant of PUSH on d-regular graphs. Using this coupling we are able to show that RPULL is asymptotically the same as PUSH for d-regular graphs. By utilising the known results for PUSH of Sauerwald [89] we are able to bound the broadcast time of RPULL by $\mathcal{O}\left(t_{mix} + \log n\right)$ where t_{mix} is the mixing time of a certain random walk.

Daum et al. [39] state their bound on the broadcast time of RPULL in terms of the broadcast time of PULL. Combining the bound given by Giakkoupis [58] with the result of Daum et al. [39] gives a bound of $\mathcal{O}\left(\phi^{-1} \cdot \log^2(n)\right)$ rounds for RPULL on a d-regular graph with conductance ϕ. It follows from our result and that of Giakkoupis [58] that RPULL requires $\mathcal{O}\left(\phi^{-1} \cdot \log(n)\right)$ rounds where ϕ is the conductance of the graph. This improves the bound given by Daum et al. [39]

by a logarithmic factor for d-regular graphs. Furthermore it matches a lower bound for the optimal algorithm given in Ghaffari and Newport [55]. RPULL is therefore an optimal algorithm for d-regular graphs in the "mobile telephone model".

4.1.3 Definitions and Model

Let $G = (V, E)$ be a (simple) undirected graph and let $\mathbf{A}$ denote the adjacency matrix of G. Recall that $\mathbf{A}$ is defined as follows,

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

For a graph G we define the $n \times n$ matrix $\mathbf{P}(G)$ to be the transition matrix. The entry $\mathbf{P}_{i,j}$ is defined as follows.

$$\mathbf{P}_{i,j} = \begin{cases} \frac{1}{d_i} & \text{if } \mathbf{A}_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

The transition matrix determines the communication pattern of our algorithms. i.e., $\mathbf{P}_{i,j}$ is the probability that i contacts j. [1]

Recall that the Laplacian matrix $\mathbf{L}$ of a graph G is defined as follows:

Definition 4.1.1. *Given an undirected, unweighted graph $G = (V, E)$ with n vertices, the Laplace Matrix $\mathbf{L}(G)$ is an $n \times n$ matrix defined by*

$$\mathbf{L}_{u,v} = \begin{cases} d_u & \textit{if } u = v \\ -1 & \textit{if } \{u,v\} \in E \\ 0 & \textit{otherwise} \end{cases} \tag{4.4}$$

Let I^t be the set of informed nodes at the start of round t. The broadcast time

[1] Note that this is not the same as the event that i informs j.

of an algorithm is the first time step after which every vertex is informed. $T_{\mathcal{A}}^{\mathbf{R}}(G,p)$ denotes the number of rounds algorithm $\mathcal{A}$ requires to inform all nodes of G with probability at least $1-p$ and a transition matrix $\mathbf{R}$ (cf. [89]). For $0 < p < 1$,

$$T_{\mathcal{A}}^{\mathbf{R}}(G,p) := \min\{t \in \mathbb{N} | \Pr\left(I^t = V\right)\} \geq 1-p$$

The PUSH algorithm is defined as follows. In each round, each node $u \in I^t$ chooses a neighbour to send the rumour to according to the u'th row for the transition matrix $\mathbf{P}(G)$.

The RPULL algorithm is defined similarly. Our definition is the same as the algorithm found in Daum et al. [39] In each round, $i \notin I^t$ sends a request to a neighbour i with probability $\mathbf{P}_{i,j}$. Each node $v \in I^t$ that receives 1 or more requests responds to a single request. A node $v \in I^t$ that receives $r_v \geq 1$ requests, chooses a single request to respond to with probability $\frac{1}{r_v}$. i.e., where an informed node receives multiple requests tie breaking is done uniformly at random.

Denote the outer boundary of a set of vertices S by ∂S. The outer boundary is the set of nodes u such that for $u \notin S$ there exists an edge (u,v) such that $v \in S$.

$$\partial S = \{u : (u,v) \in E \wedge v \in S \subseteq V \wedge u \notin S\}$$

We denote the inner boundary of a set of vertices S as $H(S)$. $u \in H(S)$ if $u \in S$ and there exists an edge (u,v) such that $v \in \partial S$.

$$H(S) = \{u : (u,v) \in E \wedge u \in S \wedge v \in \partial S\}$$

Let $t_{mix}^{\mathbf{R}}(G,\varepsilon)$ denote the *mixing time* of a random walk on the graph G with transition matrix $\mathbf{R}$. Using Definition 1.1.13, $t_{mix}^{\mathbf{R}}(G,\varepsilon)$ is defined as follows,

$$t_{mix}^{\mathbf{R}}(G,\varepsilon) = \min\left\{t \in \mathbb{N} \quad | \quad \|r_s^t - \pi\| \leq \varepsilon \quad \text{for any starting vertex } s\right\}$$

where π denotes the stationary distribution vector, r_s^t denotes the probability dis-

tribution of a random walk after t steps starting at vertex s with transition matrix $\mathbf{R}$ and $\|\mu - \nu\|$ is the variation distance (See Definition 1.1.12).

4.2 Analysis

In this section we show bounds for the broadcast time of RPULL on arbitrary graphs. To show these results we first establish a relationship between RPULL and PUSH on d-regular graphs. We show that the broadcast time of RPULL is upper bounded by the broadcast time of PUSH for d-regular graphs (Theorem 4.2.8). Using known relationships between the classical PUSH and PULL algorithms this shows that the broadcast times of RPULL and PULL are asymptotically the same on d-regular graphs (Corollary 4.2.3).

Using our upper bound for the broadcast time of RPULL on d-regular graphs we extend our results to arbitrary graphs by utilising arguments and results from Sauerwald [89] for the PUSH algorithm. We obtain two results. Firstly, Lemma 4.2.10 extends the relationship between RPULL and PUSH to arbitrary graphs. The lemma upper bounds the broadcast time of RPULL as a function of the broadcast time of PUSH and the ratio of the minimum and maximum degrees of the graph. This result establishes a result between RPULL and PUSH on arbitrary graphs. However the resulting upper bound is not necessarily tight. We therefore follow the approach of Sauerwald [89] and upper bound the broadcast time of RPULL in terms of the mixing time of a certain random walk that we define in our analysis(Theorem 4.2.12). This bound for RPULL matches the bound given for PUSH by Sauerwald [89, Theorem 4].

We believe that the results shown in Lemma 4.2.10 and Theorem 4.2.12 both to be interesting since there are graphs where one or the other may not be tight. For example, consider the star graph. Lemma 4.2.10 implies a bound of $\mathcal{O}(n^2 \cdot \log n)$ where as Theorem 4.2.12 recovers a bound of $\mathcal{O}(n \log n)$. On the other hand, in the case of Hypercubes Theorem 4.2.12 is known not to be tight for PUSH due to the results of Feige et al. [50]. However using the regularity of Hypercubes(Theorem 4.2.8), we

have that the broadcast time of RPULL is asymptotically upper bounded by the broadcast time of the classical PUSH algorithm and therefore the broadcast time of RPULL is $\mathcal{O}(\log n)$.

4.2.1 Regular Graphs

We first consider the family of d-regular graphs for $d \geq 2$. Our goal of this section is to show the following result that states that on d-regular graphs the broadcast time of the RPULL algorithm is asymptotically upper bounded by the broadcast time of PUSH. This result is obtained through the construction of a coupling between RPULL and a lazy variant of PUSH (Lazy PUSH). Using our coupling we show that the growth of the informed set under RPULL stochastically dominates the growth of the informed set under Lazy PUSH. The theorem follows since the broadcast time of PUSH ad Lazy PUSH are asymptotically the same.

Theorem 4.2.8. *Let $T_{rp}^{\mathbf{P}}(G)$ and $T_{p}^{\mathbf{P}}(g)$ be the broadcast time of* RPULL *and* PUSH *on a graph G with transition matrix $\mathbf{P}$. For d-regular G (equivalently for symmetric transition matrix $\mathbf{P}$),*

$$T_{rp}^{\mathbf{P}}(G) = \mathcal{O}\left(T_{p}^{\mathbf{P}}(G)\right)$$

Theorem 4.2.8 allows us to show a relationship between the broadcast time of RPULL and the classical PULL algorithm on d-regular graphs. In particular we show that the broadcast time of RPULL and PULL are asymptotically the same on d-regular graphs (Corollary 4.2.3).

To show Corollary 4.2.3 we use the following lemma by Sauerwald [89] that relates the classical PUSH algorithm with the classical PULL algorithm as well as the PUSH-PULL algorithm.

Lemma 4.2.1 (Sauerwald [89] Lemma 11)**.** *If* $\mathbf{Q}$ *is a symmetric and stochastic* $n \times n$ *matrix then,*

$$T^{\mathbf{Q}}_{\text{PUSH}}(G, n^{-1}) = T^{\mathbf{Q}}_{\text{PULL}}(G, n^{-1})$$
$$T^{\mathbf{Q}}_{\text{PUSH}}(G, n^{-1}) = \Theta\left(T^{\mathbf{Q}}_{\text{PUSH}-\text{PULL}}(G, n^{-1}) + \log(n)\right)$$

In order to apply Lemma 4.2.1 we observe that the transition matrix $\mathbf{P}$ for d-regular graphs is symmetric. Using this observation, we can apply the first statement of Lemma 4.2.1 to relate PUSH and PULL on d-regular graphs. It follows that PUSH and PULL are asymptotically the same for d-regular graphs.

Observation 4.2.2. *For undirected, d-regular graphs, all non-zero entries of the transition matrix* $\mathbf{P}$ *take the value* $\frac{1}{d}$*. It follows that* $\mathbf{P}$ *is symmetric since* $\forall i, j : P_{ij} = Pji$*. Since* $\mathbf{P}$ *is symmetric and stochastic, it follows that* $\mathbf{P}$ *is doubly stochastic. i.e., all rows and columns sum to one.*

Corollary 4.2.3 bounds the broadcast time RPULL in terms of the broadcast time of PULL for d-regular graphs. The result states that for d-regular graphs, the broadcast times of RPULL and PULL are asymptotically the same. This improves the bound of Daum et al. [39] on the family of d-regular graphs by a logarithmic factor.

Corollary 4.2.3. *Let* $T^{\mathbf{P}}_{rp}(G)$ *and* $T^{\mathbf{P}}_{l}(g)$ *be the broadcast time of* RPULL *and* PULL *on a graph* G *with transition matrix* $\mathbf{P}$*. For d-regular* G *(equivalently for symmetric transition matrix* $\mathbf{P}$*),*

$$T^{\mathbf{P}}_{rp}(G) = \Theta\left(T^{\mathbf{P}}_{l}(G)\right)$$

Proof. The upper bound follows from the result shown in Lemma 4.2.8 relating PUSH and RPULL combined with Lemma 4.2.1 and Observation 4.2.2. For the

lower bound, observe that for any set of choices by the uninformed nodes, RPULL can not inform more nodes that PULL. □

In the remainder of this section we show Theorem 4.2.8. Theorem 4.2.8 upper bounds the broadcast time of RPULL by the broadcast time of PUSH. To compare the different algorithms we will consider the growth of the number of informed nodes in a given round. If we are able to show that in any round the growth of the number of informed nodes under RPULL is at least the growth of the number of informed nodes under PUSH, it follows that the broadcast time of RPULL is upper bounded by the broadcast time of PUSH. We begin by showing that it is not straight forward to compare RPULL and PUSH directly. In particular we show that it is not possible to show that for any round that a single round of RPULL stochastically dominates a single round of PUSH. Recall that a random variable X stochastically dominates Y (denoted $X \succeq Y$) if for all $c > 0$,

$$\Pr(X > c) \geq \Pr(Y > c).$$

For this reason we instead show that a single round of RPULL stochastically dominates a lazy variant of PUSH. Since the broadcast time of our lazy variant of PUSH is asymptotically the same as the broadcast time of PUSH it follows that the broadcast time of RPULL is asymptotically upper bounded by the broadcast time of PUSH.

The following example shows that it is not possible to show that for any round that a single round of RPULL stochastically dominates a single round of PUSH. Let ΔI_{rp}^t and ΔI_p^t be the random variables denoting the number of nodes informed in a round t for RPULL and PUSH respectively. We would like to be able to show that for any round t, $\Delta I_{rp}^t \succeq \Delta I_p^t$. i.e., the growth of the informed set under RPULL stochastically dominates the growth of the informed set under PUSH. However, it is possible to show that such a stochastic dominance is not possible.

Consider the following counter example. Figure 4.1 depicts the initial state with

a single informed node where edges (i, j) where both i and j are uninformed are omitted. For the state shown in Figure 4.1, observe that an execution of PUSH informs a new node with probability 1 in the first round. In fact this is true for any graph where self loops are excluded. In comparison, the first round of RPULL on a d-regular graph informs a new node with probability,

$$0.63 \leq 1 - e^{-1} \leq 1 - \left(1 - \frac{1}{d}\right)^d \leq \frac{3}{4} \tag{4.5}$$

for $d \geq 2$.

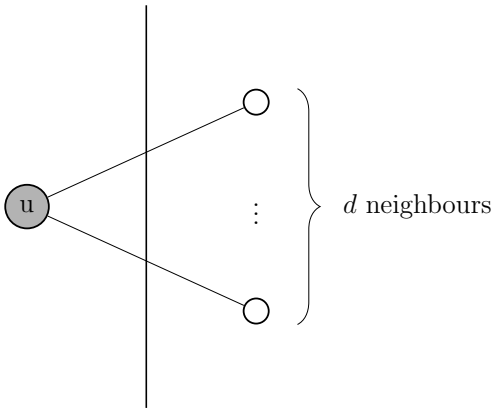

Figure 4.1: Restricted PULL: Initial State

This example shows that it is not possible to show that for any round a single round of RPULL stochastically dominates a single round of PUSH. For this reason instead of comparing RPULL and PUSH directly we will instead compare RPULL and a lazy variant of PUSH. We will refer to this variant of PUSH as Lazy PUSH and it is defined as follows. Under Lazy PUSH, each informed node participates in a given round with probability 1/2. If an informed node is participating in a round it executes a single round of PUSH. i.e., it sends the rumour to a neighbour chosen uniformly at random. Otherwise, if a node chooses not to participate then it performs no operations in that round. More formally we define the transition matrix $\mathbf{Q}$ for Lazy PUSH as follows:

$$\mathbf{Q} := \frac{1}{2}(\mathbf{P} + \mathbf{I})$$

where $\mathbf{P}$ is the transition matrix defined in Eq. 4.3.

The following observation states that the broadcast time of Lazy PUSH and PUSH are asymptotically the same.

Observation 4.2.4. *For a graph $G = (V, E)$, let $T_{lp}(G)$ and $T_p(G)$ denote the broadcast time of Lazy* PUSH *and* PUSH *respectively.*

$$T_{lp}(G) = \Theta\left(T_p(G)\right)$$

If we now revisit the previous example of the initial state (Figure 4.1), it follows that since in the first round PUSH creates a newly informed node with probability 1 that Lazy PUSH informs a new node with probability 1/2. Using our previous calculation (Eq. 4.5) it follows that for the initial state (Figure 4.1) we obtain that,

$$\Pr\left(\Delta I^1_{RP}\right) > \Pr\left(\Delta I^1_{LP}\right)$$

where $\Delta I^t := I^t - I^{t-1}$ for $t > 0$. i.e, Starting in the initial state, a single round of RPULL stochastically dominates a single round of Lazy PUSH.

It remains to show that for any round t, $\Delta I^t_{RP} \succeq \Delta I^t_{LP}$. Once shown it follows from this stochastic dominance that the broadcast time for RPULL is upper bounded by the broadcast time and Lazy PUSH. Furthermore, since the broadcast time of Lazy PUSH is asymptotically the same as PUSH, it follows that the broadcast time of RPULL is asymptotically upper bounded by the broadcast time of PUSH.

In the remainder of this section we prove that for an arbitrary round t that a single round of RPULL stochastically dominates a single round of Lazy PUSH (Lemma 4.2.7).

Comparing RPULL and Lazy PUSH

In order to show that RPULL stochastically dominates Lazy PUSH (Lemma 4.2.7) we construct a coupling for a single round of the RPULL and Lazy PUSH processes.

Lemma 4.2.7 can then be applied in successive rounds to show the desired result.

In the previous example we considered the contribution of a single informed node. This approach is based on the observation that each informed node can inform at most one new node per round under PUSH, Lazy PUSH and RPULL. Using this observation we are able to construct a coupling between a single round of Lazy PUSH and RPULL. Our coupling in Lemma 4.2.7 considers the following setup. Let $H(I^t) := \{u_1, u_2, \ldots, u_{h_t}\}$ be the set of nodes that constitute the inner boundary of I^t and $h^t = |H(I^t)|$. The set $H(I^t)$ contains all nodes that are able to inform a new node and thus contribute to the growth of the informed set. We must therefore consider the contribution of each $u_i \in H(I^t)$.

At the start of each round, each node $u \in H(I^t)$ generates a token. We consider the outcome of each token sequentially. i.e., we consider a single round as $|H(I^t)|$ substeps. For u's token to create a newly informed node in a given substep the following two conditions must be satisfied:

(a) The token needs to traverse an edge (u, v) such that $u \in I^t$ and $v \in \partial I^t$ and,

(b) $v \in \partial I^t$ is a vertex that has not received a token in a previous substep.

The number of tokens that satisfy both of the above conditions describes exactly the growth of the informed set. We will say that these tokens create newly informed nodes.

For a node $u_i \in H(I^t)$, the probability that u_i's token creates a newly informed node depends on the outcome of the other nodes $u_{j\neq i} \in H(I^t)$. For example, consider the node d_k shown in Figure 4.2. If the event that u's token informs d_k is true, this changes the number of nodes adjacent to v without a token. We refer to d_k as a *shared neighbour*.

We begin by handling the simpler case where u is a node such that it does not share a neighbour with any other node in I^t. That is the outcome for the tokens of $u_i \in H(I^t)$ and $u_{i-1} \in H(I^t)$ are independent.

Let $u \in I^t$ be a fixed node with $k \leq d$ uninformed neighbours such that the

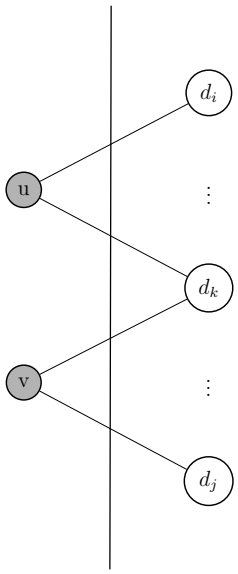

Figure 4.2: Restricted PULL: Dependency between shared neighbours

probability that u's token informs a new node is independent of the outcome of any other $v \in I^t$. i.e., u has no shared neighbours.

The probability that under RPULL u informs a new node is $1 - \left(1 - \frac{1}{d}\right)^k$ whilst the probability that under Lazy PUSH u informs a new node is $\frac{1}{2} \cdot \frac{k}{d}$. This can be seen as a generalisation of the example given in Figure 4.1. Lemma 4.2.5 shows that for a node u (as defined above) the probability that under RPULL u informs a new node is at least the probability that u informs a new node under Lazy PUSH.

Lemma 4.2.5. *For all $d > 1$ and $k \leq d$, the following inequality holds:*

$$\frac{1}{2} \cdot \frac{k}{d} \leq 1 - \left(1 - \frac{1}{d}\right)^k$$

Proof. First rewrite the term $\left(1 - \frac{1}{d}\right)^k$ as a binomial series.

$$\begin{aligned}\left(1 - \frac{1}{d}\right)^k &= \sum_{m=0}^{\infty} \binom{k}{m} \left(-\frac{1}{d}\right)^m \\ &= 1 - \frac{k}{d} + \frac{k(k-1)}{2d^2} - \frac{k(k-1)(k-2)}{6d^3} + \mathcal{O}\left(\frac{1}{d^4}\right)\end{aligned}$$

Note that this series converges absolutely, as $\left|-\frac{1}{d}\right| < 1$ for all values of d.

$$\implies 1 - \left(1 - \frac{1}{d}\right)^k = \frac{k}{d} - \frac{k(k-1)}{2d^2} + \frac{k(k-1)(k-2)}{6d^3} - \mathcal{O}\left(\frac{1}{d^4}\right)$$

Since the binomial series for $(1 - \frac{1}{d})^k$ converges absolutely it follows that each positive term of $1 - (1 - \frac{1}{d})^k$ will be larger than the negative term that follows. By considering the first two terms we obtain the following lower bound

$$1 - \left(1 - \frac{1}{d}\right)^k \geq \frac{k}{d} - \frac{k(k-1)}{2d^2}$$

Hence if we can prove that $\frac{k}{d} - \frac{k(k-1)}{2d^2} \geq \frac{1}{2} \cdot \frac{k}{d}$ our claim will follows.

Assume that $\frac{k}{d} - \frac{k(k-1)}{2d^2} \geq \frac{1}{2} \cdot \frac{k}{d}$ and rearrange:

$$\frac{1}{2} \cdot \frac{k}{d} \geq \frac{k(k-1)}{2d^2} \implies \frac{k \cdot d}{2d^2} \geq \frac{k(k-1)}{2d^2} \implies d \geq k - 1$$

This is always true, by the definition of k, hence the expression is always true, and the proof is complete.

□

Lemma 4.2.5 states that for any node without shared neighbours the probability that it is informed during in a single round of RPULL is greater than or equal to the probability that it is informed during a single round of Lazy PUSH. However, as previously stated we must consider situations such as the one shown in Figure 4.2 where there are additional dependencies. In order to handle the dependencies between nodes when there are shared neighbour we construct a coupling between RPULL and Lazy PUSH. This coupling will allow us to apply Lemma 4.2.5 to show the required stochastic dominance between RPULL and Lazy PUSH.

Overview of Coupling

For a fixed round t, we consider h^t substeps. Define the following set of events $\mathscr{E}^{\mathcal{A}}_{u_1}, \ldots, \mathscr{E}^{\mathcal{A}}_{u_{h^t}}$ where

$$\mathscr{E}^{\mathcal{A}}_{u_i} := \{u_i \text{ creates a new informed node under algorithm } \mathcal{A}\}$$

For $i \in [1, h^t]$ we reveal the events $\mathscr{E}^{\mathcal{A}}_{u_i}$ sequentially.

We define the following indicator random variables,

$$X_i^t = \begin{cases} 1 & \text{if } \mathscr{E}^{LP}_{u_i} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad Y_i^t = \begin{cases} 1 & \text{if } \mathscr{E}^{RP}_{u_i} \\ 0 & \text{otherwise} \end{cases}$$

and let,

$$X^t = \sum_{i=1}^{h^t} X_i \quad \text{and} \quad Y^t = \sum_{i=1}^{h^t} Y_i$$

Note that the probability that $X_i = 1$ depends on $X_1, X_2, \ldots X_{i-1}$. Similarly the probability that $Y_i = 1$ depends on $Y_1, Y_2, \ldots Y_{i-1}$.

We begin by considering the following equivalent definitions of Lazy PUSH and RPULL that will allow us to couple the movement of tokens in the two processes.

Let u be a node with $k \leq d$ uninformed neighbours. Under Lazy PUSH u informs a new node with probability,

$$\Pr\left(X_u = 1\right) = \Pr\left(\mathscr{E}^{LP}_u\right) = \frac{1}{2} \cdot \frac{k}{d}$$

We define Lazy PUSH as follows. If $X_u = 1$, u sends its token to one of its k uninformed neighbours uniformly at random. i.e, each of the uninformed neighbours $v_1, v_2, \ldots v_k$ receives the token from u with probability $\frac{1}{k}$.

We formulate RPULL in a similar way. Under RPULL u informs a new node

with probability,

$$\Pr\left(Y_u = 1\right) = \Pr\left(\mathscr{E}_u^{RP}\right) = 1 - \left(1 - \frac{1}{d}\right)^k$$

If $Y_u = 1$, then u gives its token to an uninformed neighbour v_i with probability $\frac{1}{k}$ for $i \in [1, k]$. Since $\Pr\left(Y_u = 1\right)$ is the probability that u informs a new node, to confirm that the movement of tokens in this process has the same marginal distribution as RPULL it remains to show that the u's token moves to each of the k uninformed neighbours with the same probability i.e., $\frac{1}{k}$. This is shown in Lemma 4.2.6.

By formulating the processes it allows us to defer the decisions of the uninformed nodes and couple the movement of tokens. Since in both cases, the token moves to an uninformed neighbour with equal probability we are able to use a common random variable to couple the movement of the tokens. This ensures that in each substep the two processes are in the same state and allows us to apply Lemma 4.2.5 in each substep.

Using Lemma 4.2.5, we know that if in a substep i both processes are in the same state then,

$$\Pr\left(Y_i = 1\right) \geq \Pr\left(X_i = 1\right)$$

i.e., the probability that RPULL informs a new node in the substep i is at least the probability then Lazy PUSH informs a new node. We use this intuition to couple RPULL and Lazy PUSH.

The following lemma states that given that u informs a new node under RPULL, the token moves to any of the k uninformed neighbours of u with equal probability.

Lemma 4.2.6. *Let $G = (V, E)$ be a d-regular graph. For $u \in H(I^t)$ with $k \leq d$ uninformed neighbours, let $v_1, v_2, \ldots v_k$ be the k uninformed neighbours of u. For each $i \in [1, k]$, then*

$$\Pr\left(u \text{ informs } v_i \;\middle|\; \mathcal{E}_u^{RP}\right) = \frac{1}{k}$$

Proof. The lemma follows by the definition of RPULL. Since the event $\mathcal{E}_u^{RP}$ holds it follows that u has received one or more requests from the nodes $v_1, v_2, \ldots v_k$. Assume there is a node v_j such that $\Pr\left(u \text{ informs } v_j \mid \mathcal{E}_u^{RP}\right) \neq \frac{1}{k}$. Without loss of generality, assume $\Pr\left(u \text{ informs } v_j \mid \mathcal{E}_u^{RP}\right) \geq \frac{1}{k}$. This implies that either u was more likely to choose v_j during the tie breaking step or that v_j was more likely to choose u. Since G is d-regular and tie breaking is done uniformly at random, this contradicts the definition of RPULL. □

We now show the following lemma that states that a round of RPULL stochastically dominates a round of Lazy PUSH.

Lemma 4.2.7. *For a d-regular graph G and an arbitrary round t. Let $\Delta I_{\mathcal{A}}^{t+1} := |I_{\mathcal{A}}^{t+1} - I_{\mathcal{A}}^{t}|$ be the number of newly informed nodes for algorithm $\mathcal{A}$.*

For every $c \geq 0$ it holds that

$$\Pr\left(\Delta I_{LP}^{t+1} > c\right) \leq \Pr\left(\Delta I_{RP}^{t+1} > c\right)$$

where LP denotes Lazy PUSH *and RP denotes* RPULL.

Proof. Recall the definition of the following indicator random variables,

$$X_i^t = \begin{cases} 1 & \text{if } \mathscr{E}_{u_i}^{LP} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad Y_i^t = \begin{cases} 1 & \text{if } \mathscr{E}_{u_i}^{RP} \\ 0 & \text{otherwise} \end{cases}$$

and let,

$$X^t = \sum_{i=1}^{h^t} X_i \quad \text{and} \quad Y^t = \sum_{i=1}^{h^t} Y_i$$

Rewriting Lemma 4.2.7 using these random variables we have:

$$\Pr\left(\Delta I_{LP}^{t} > c\right) = \Pr\left(X^t > c\right) \leq \Pr\left(Y^t > c\right) = \Pr\left(\Delta I_{RP}^{t} > c\right)$$

In subsequent descriptions we omit the superscript because it is clear from context

since we only discuss a single round.

We consider the round t as h^t substeps where $h^t = |H(I^t)|$ is the number of nodes that form the inner boundary of the informed set.

It remains to show that,

$$\Pr\left(\sum_{i=1}^{h^t} X_i > c\right) \leq \Pr\left(\sum_{i=1}^{h^t} Y_i > c\right) \tag{4.6}$$

Let $i \in (1, h^t)$ be the current substep and let $p_i := \Pr(X_i = 1 \mid \mathcal{N}_{i-1})$ and $q_i := \Pr(Y_i = 1 \mid \mathcal{N}_{i-1})$ where $\mathcal{N}_{i-1}$ denotes the nodes in ∂I^t that have received a token in the substeps $1, \ldots, i-1$. That is, the probability that i creates a newly informed node given the outcome of the previous substeps.

Let k_i be the number of uninformed neighbours of u_i. Define $U_i \in \{1, \ldots, k_i\}$ be a shared random variable that takes the values in $\{1, \ldots, k_i\}$ uniformly at random.

The coupling proceeds as follows:

- If $X_i = 1$, then $Y_i = 1$, then the token moves to U_i in both processes
- If $X_i = 0$, then $Y_i = 1$ with probability $q_i - p_i$

The crucial point here is that when both $X_i = Y_i = 1$ both processes inform the same neighbour. This ensures that the number of uninformed neighbours of a node u_j for $j \in [i+1, h^t]$ is the same for both processes and that in subsequent steps $p_{i+1} \geq q_{i+1}$.

Let t' be the first substep such that $X_{t'} \neq Y_{t'}$. By the definition of our coupling it follows that $X_{t'} = 0$ and $Y_{t'} = 1$, hence $\sum_{i=1}^{h^t} X_i \leq \sum_{i=1}^{h^t} \sum Y_i$. This is irrespective of the outcome of subsequent steps since each node can inform at most one new node.

It therefore follows that $Y \geq X$ which yields that in each time step RPULL $\succeq$ Lazy PUSH.

□

Lemma 4.2.7 states that for any round, the growth of the informed set under RPULL stochastically dominates the growth of the informed set under Lazy

PUSH. With this lemma we are now ready to show the main result in this section (Theorem 4.2.8). Theorem 4.2.8 states that the broadcast time of RPULL is asymptotically upper bounded by the broadcast of PUSH on d-regular graphs. To show the theorem, we reason through repeated application of Lemma 4.2.7 that the broadcast time of RPULL is upper bounded by the broadcast time of Lazy PUSH. The statement of the theorem then follows since the broadcast time of Lazy PUSH and PUSH are asymptotically the same.

Theorem 4.2.8. *Let $T_{rp}^{\mathbf{P}}(G)$ and $T_{p}^{\mathbf{P}}(g)$ be the broadcast time of* RPULL *and* PUSH *on a graph G with transition matrix $\mathbf{P}$. For d-regular G (equivalently for symmetric transition matrix $\mathbf{P}$),*

$$T_{rp}^{\mathbf{P}}(G) = \mathcal{O}\left(T_{p}^{\mathbf{P}}(G)\right)$$

Proof. Let $T_{lp}(G)$ be the broadcast time of Lazy PUSH on G. From our previous observation we have that $T_p(G) = \Theta\left(T_{lp}(G)\right)$. Lemma 4.2.7 states that in any round t, the growth of the informed set under RPULL stochastically dominates the growth of the informed set under Lazy PUSH. By applying the lemma in each round it follows that the broadcast time of RPULL is stochastically smaller than the broadcast time of Lazy PUSH. Since $T_{rp}(G)$ is stochastically smaller than $T_{lp}(G)$ it follows that

$$T_{rp}(G) = \mathcal{O}\left(T_{lp}(G)\right) = \mathcal{O}\left(T_p(G)\right)$$

□

4.2.2 Non-regular graphs

For non-regular graphs, our approach used for d-regular graphs can prove difficult to generalise. A key element of the coupling in the previous section is that conditioned on the event that an informed node u creates a newly informed node, each of u's uninformed nodes is equally likely to be the newly informed node. This is not

necessarily true in a non-regular graph since the neighbours of u have different degrees. It follows that the probability that each neighbour sends a request to u differs for each uninformed neighbour of u.

To generalise our result to arbitrary graphs we utilise results from Sauerwald [89]. The work of Sauerwald [89] studies the performance of PUSH on general graphs. In particular, Sauerwald [89] show that the broadcast time of PUSH is bounded by $\mathcal{O}\left(t_{mix} + \log(n)\right)$ where t_{mix} is the mixing time of a certain random walk. When t_{mix} is $\mathcal{O}\left(\log(n)\right)$ the results of Sauerwald [89] match optimal bounds. Using our result for d-regular graphs in combination with results by Sauerwald [89], we are able to show results that apply in a more general setting.

Our first result, Lemma 4.2.10 establishes a relationship between RPULL and PUSH on arbitrary graphs. This relationship is in terms of the maximum and minimum degrees of the graph. To show Lemma 4.2.10, we use the following results from Sauerwald [89]. Lemma 4.2.9 bounds the effect of changing the transition matrix on the broadcast time of PUSH. In particular, the effect of replacing the transition matrix $\mathbf{P}$ for $\mathbf{Q}$. Note that the transition matrix $\mathbf{Q}$ as defined above is symmetric.

Lemma 4.2.9 (Sauerwald [89] Lemma 12)**.** *Let $T_p^{\mathbf{P}}(G)$ denote the broadcast time of* PUSH *with transition matrix $\mathbf{P}$. For $\mathbf{Q} := \mathbf{I} - \frac{1}{\Delta} \cdot \mathbf{L}$, where $\mathbf{I}$ is the identity matrix and $\mathbf{L}$ is the Laplacian matrix of G, we have*

$$T_p^{\mathbf{P}}(G, n^{-1}) \leq T_p^{\mathbf{Q}}(G, n^{-1}) = \frac{\Delta}{\delta} \cdot \mathcal{O}\left(T_p^{\mathbf{P}}(G, n^{-1}) + \log(n)\right)$$

where Δ and δ are the maximum and minimum degrees of G.

Lemma 4.2.10 establishes a relationship between PUSH and RPULL for a graph G with maximum degree Δ and minimum degree δ.

Lemma 4.2.10. *For a graph G with minimum degree δ and maximum degree Δ,*

$$T_{rp}^{\mathbf{P}}(G, n^{-1}) = \frac{\Delta}{\delta} \cdot \mathcal{O}\left(T_p^{\mathbf{P}}(G, n^{-1}) + \log(n)\right)$$

where $T_p(G)$ and $T_{rp}(G)$ denote the broadcast time of PUSH *and* RPULL *respectively.*

Proof. The result follows from the result of Sauerwald [89] stated in Lemma 4.2.9 combined with Theorem 4.2.8 that states that RPULL asymptotically upper bounded by PUSH on regular graphs.

Let $\mathbf{Q} := \mathbf{I} - \frac{1}{\Delta} \cdot \mathbf{L}$ be a $n \times n$ transition matrix. Applying Theorem 4.2.8, It follows that $T_{rp}^{\mathbf{Q}}(G)$ is asymptotically the upper bounded $T_p^{\mathbf{Q}}(G)$. By substituting into Lemma 4.2.9, we get that

$$T_{rp}^{\mathbf{Q}}(G) = \frac{\Delta}{\delta} \cdot \mathcal{O}\left(T_p^{\mathbf{P}}(G) + \log(n)\right)$$

Using a standard coupling argument it follows that $T_{rp}^{\mathbf{Q}}(G)$ is greater than or equal to $T_{rp}^{\mathbf{P}}(G)$.

The same bound therefore holds for $T_{rp}^{\mathbf{P}}(G)$ and the statement follows. □

This establishes a relationship between the broadcast times of RPULL and PUSH on arbitrary graphs. However, the result in Lemma 4.2.10 is not necessarily tight. For example, consider a star graph on n vertices. Using the solution to the coupon collector problem, the broadcast time of PUSH, $T_p^{\mathbf{P}}(G) = \mathcal{O}(n \log n)$ w.h.p. The above lemma implies that $T_{rp}^{\mathbf{P}}(G) = \mathcal{O}(n^2 \log n)$.

For RPULL, when the source node is a leaf node, then the first step of RPULL is equivalent to the final step of PUSH. Once the rumour has reached the centre node, a further $\Theta(n)$ rounds is sufficient for RPULL to inform all nodes. It follows that in the worst case, $T_{rp}^{\mathbf{P}}(G) = \mathcal{O}(n \log n)$ w.h.p.

Since there are cases where the result in Lemma 4.2.10 is not necessarily tight we now follow the same approach as Sauerwald [89] and bound the broadcast of RPULL in terms of the mixing time of a random walk. Theorem 4.2.12 bounds the broadcast time of RPULL in terms of the mixing time of a random walk on G with transition matrix $\mathbf{Q} = \mathbf{I} - \frac{1}{\Delta} \cdot \mathbf{L}$ where Δ is the maximum degree of G. The proof of the theorem follows from arguments found in the proof of Theorem 4 in Sauerwald

[89].

The proof of Theorem 4.2.12 will use the following result stated as Theorem 3 in Sauerwald [89] that follows from results found in Boyd et al. [27]. Theorem 4.2.11 (Sauerwald [89, Theorem 3]) bounds the broadcast time of the PUSH-PULL algorithm in terms of the mixing time of a random walk.

Theorem 4.2.11 (Theorem 3 Sauerwald [89])**.** *For any graph G and a symmetric stochastic matrix $\mathbf{Q}$ it holds*

$$T^{\mathbf{Q}}_{\text{PUSH}-\text{PULL}}(G, n^{-1}) = \mathcal{O}\left(t^{\frac{1}{2}\mathbf{Q}+\frac{1}{2}\mathbf{I}}_{mix}(G, n^{-2}) + \log(n)\right)$$

Theorem 4.2.12. *For any graph G with maximum degree Δ,*

$$T^{\mathbf{P}}_{rp}(G, n^{-1}) = \mathcal{O}\left(t^{\mathbf{Q}}_{mix}(G, n^{-2}) + \log(n)\right)$$

where $\mathbf{P}$ is the transition matrix defined in Eq. 4.3, $\mathbf{Q} := \mathbf{I} - \frac{1}{\Delta} \cdot \mathbf{L}$ and $\mathbf{L}$ is the Lapalcian matrix of G

Proof. Since $\mathbf{Q}$ is symmetric, we first apply Lemma 4.2.1 to obtain,

$$T^{\mathbf{Q}}_{rp}(G) = \mathcal{O}\left(T^{\mathbf{Q}}_{p}(G)\right) = \Theta\left(T^{\mathbf{Q}}_{\text{PUSH}-\text{PULL}}(G, n^{-1}) + \log(n)\right)$$

The first equality holds since as previously stated $T^{\mathbf{Q}}_{p}(G)$ asymptotically upper bounds $T^{\mathbf{Q}}_{rp}(G)$ (Lemma 4.2.8).

In order to obtain the result we apply Theorem 4.2.11,

$$\begin{aligned} T^{\mathbf{P}}_{rp}(G) \leq T^{\mathbf{Q}}_{rp}(G) &= \Theta\left(T^{\mathbf{Q}}_{\text{PUSH}-\text{PULL}}(G, n^{-1}) + \log(n)\right) \\ &= \mathcal{O}\left(t^{\frac{1}{2}\mathbf{Q}+\frac{1}{2}\mathbf{I}}_{mix}(G, n^{-2}) + \log(n)\right) \\ &= \mathcal{O}\left(t^{\mathbf{Q}}_{mix}(G, n^{-2}) + \log(n)\right) \end{aligned}$$

□

Observe that when $t^{\mathbf{Q}}_{mix}(G) = \mathcal{O}(\log(n))$, Theorem 4.2.12 gives a bound of $\mathcal{O}(\log(n))$. This is optimal for both PUSH and RPULL since both the broadcast time of both processes is $\Omega(\log(n))$ [2]. As stated in Sauerwald [89] it is the case that $t^{\mathbf{Q}}_{mix}(G) = \mathcal{O}(\log(n))$ for graph classes such as the complete graphs, expanders and random graphs and, also for several Cayley graphs. For these graphs classes it follows that the bound is optimal for RPULL.

However, there are also graphs where the bound in Theorem 4.2.12 is not tight. For example, consider the Hypercube. Since $t^{\mathbf{Q}}_{mix}(G, e^{-1}) = \Theta(\log(n)\log\log(n))$ using Theorem 4.2.12 implies a bound on the broadcast time of RPULL of $\mathcal{O}(\log(n)\log\log(n))$. However, since it is known that for the Hypercube the broadcast time of PUSH is bounded by $\mathcal{O}(\log(n))$ we are able to recover this bound using either Lemma 4.2.8 or Lemma 4.2.10 since the Hypercube is a regular graph. It follows that the same bound holds for RPULL. For these reasons, we believe both results (Lemma 4.2.10 and Theorem 4.2.12) to be of interest.

4.3 Conclusion

In this chapter we have studied a restricted version of the classical PULL algorithm introduced by Daum et al. [39]. In each round of the restricted PULL (RPULL) algorithm, each uninformed node sends a request to a neighbour chosen uniformly at random. Each informed node that receives 1 or more requests is able to reply to exactly 1 of these requests where tie breaking is uniformly random.

By studying the relationship between the classical PUSH algorithm and RPULL we have shown that the broadcast time of RPULL is asymptotically the same as the classical PULL algorithm on d-regular graphs. It follows that RPULL obtains an optimal bound for d-regular graphs. Using known results for the classical rumour spreading algorithms on d-regular graphs, this result improves the previous bound by Daum et al. [39] for the running time of RPULL for d-regular graphs by a logarithmic factor.

[2] The number of informed nodes can at most double in each round

For arbitrary graphs we have shown that the broadcast time of RPULL is $\mathcal{O}(t_{mix} + \log n)$ where t_{mix} is the mixing time of a certain random walk. When t_{mix} is $\mathcal{O}(\log n)$ this bound is optimal since the number of informed nodes can at most double in each round. This is the case for several notable graph classes such as complete graphs, expanders, random graphs and several Cayley graphs.

Chapter 5

Conclusions and Outlook

We have presented algorithms for three problems relating to the efficient dissemination of information in large networks. Namely the problems of load balancing, plurality consensus and, rumour spreading. When designing efficient algorithms for these problems the communication pattern employed is an important consideration. Besides influencing the design of our algorithms, the communication pattern employed may also impact various measures of performance and determine the suitability of the algorithm for a given application.

Choosing a suitable communication pattern is a multifaceted problem. Traditional wisdom suggests that additional communication allows an algorithm to obtain better results. Certainly, in some cases this holds and additional communication can be beneficial to the performance of an algorithm. Seminal works have shown that for the Greedy[d] allocation scheme a small increase in the communication overhead exponentially improves the maximum load. This is the so called *power of two choices*. Many of the results in the literature consider models where tasks are allocated sequentially. This is a limitation of these models when considered in the context of distributed load balancing. In Chapter 2 we present an infinite, parallel model. This model considers a client-server scenario where in each time step each non-idle server (bins) processes a task and a number of concurrent new tasks (balls) arrive to be allocated. Our results show that the "power of two choices" phenomenon carries over to this novel infinite and parallel model. However, unlike

previous models where tasks are allocated sequentially, allowing extra communication in our model is no guarantee of better performance. In fact, in our model it is possible that allowing tasks to query extra servers may have a negative impact. This highlights the need to consider the communication pattern carefully.

Our results are not the first to consider an infinite process. However, our model addresses the criticism of previous models that the total number of balls in the system is fixed. i.e, the number of balls that are removed are then reallocated. In each time step of our model, the number of balls being allocated is a random variable and is independent of the number of balls that are removed. It is therefore possible for the total number of balls in the system to be arbitrarily large. Despite this we are able to show a strong self stabilization property. This property combines the notion of positive recurrence with a snapshot property that bounds the maximum load of any bin in an arbitrary time step (with high probability).

An open question concerns the removal of the restriction our model has on the number of tasks allocated in a time step. The task generation model we consider upper bounds the number of tasks being allocated by the number of bins. By removing this restriction, our model could be adapted for analysing the Greedy[d] allocation schemes in scenarios where there is a high volume of traffic. Although further work is required to analyse how the maximum load diverges, our results regarding the smoothness of the allocation under the two choice process still hold when this restriction is removed.

When designing algorithms for the problems related to the efficient dissemination of information in large networks it is not sufficient to consider performance metrics such as the runtime or maximum load in isolation. The problems studied in the preceding chapters have found application in a number of different areas. It is therefore necessary to consider the suitability of the algorithm based on the requirements of a given application. These requirements are often determined by the limitations of the nodes in the network. These limitations may exist due to the heterogeneity of nodes or due to the network consisting of simpler devices such as

sensors or biological entities. In these cases our models must reflect that the nodes are limited in both their computational and communication ability. This is the focus of Chapter 3 and Chapter 4. The algorithms presented solve their respective problem where communication is limited.

In Chapter 4, we consider a variant of the classical PULL algorithm under weaker assumptions for the rumour spreading problem. The classical PULL algorithm has been extensively studied along with its symmetric counterpart PUSH. Both are suitable where nodes have limited computational resources. Recent work has begun to consider the effect of restricted communication. Under the classical PULL algorithm a node may inform multiple nodes in a given round. This is an unrealistic assumption in some of the previous examples of networks consisting of simpler nodes. The work of Daum et al. [39] and Ghaffari and Newport [55] consider the effect of removing this assumption. Our results have shown that for a number of notable graph classes, our restricted PULL algorithm is an optimal algorithm. Moreover, for certain graph classes our results positively answer a conjecture by Daum et al. [39] regarding the difference in runtime between the classical PULL and restricted PULL algorithms. It remains an open question to answer this for arbitrary graphs or under different tie breaking assumptions.

In addition to communicational limitations, in networks that consist of simple devices we must also consider the computational limitations of the nodes. The algorithms we presented in Chapter 3 are able to solve the plurality consensus problem for numerous communication patterns and on arbitrary network topologies. However, in order to be able to solve the plurality consensus problem under such a wide range of parameters there is a price. This price is paid by the memory overhead of the algorithms. This gives rise to scenarios where the algorithms may not be suitable due to the memory requirements of the algorithms exceeding the capability of the nodes. In these scenarios previous studies have presented algorithms that may be more suitable. For example, recent results by Berenbrink et al. [23] and Ghaffari and Parter [56] showed that there exists protocols that solve the plurality consensus

problem in a polylogarithmic number of rounds using only polylogarithmic memory. This answered an open question in the literature. The approach taken by the authors' algorithms is to amplify the bias towards the plurality opinion. These algorithms apply when the network topology is the complete graph. It remains an open question if algorithms based on this approach are able to generalise to arbitrary network topologies.

The work in the preceding chapters has studied the performance of algorithms under a number of communication patterns. The study of problems related to the efficient dissemination of information in large scale networks remains a fundamental problem. As examples of large scale networks continue to grow in size and number, the heterogeneity of devices is an important consideration. It is important that future models reflect the restrictions this imposes on the communication patterns considered.

Bibliography

[1] Micah Adler, Petra Berenbrink, and Klaus Schröder. "Analyzing an Infinite Parallel Job Allocation Process". In: *Proceedings of the 6th Annual European Symposium on Algorithms*. ESA '98. London, UK, UK: Springer-Verlag, 1998, pp. 417–428. ISBN: 3-540-64848-8. URL: http://dl.acm.org/citation.cfm?id=647908.740138.

[2] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Rasmussen. "Parallel randomized load balancing". In: *Random Structures & Algorithms* 13.2 (1998), pp. 159–188. ISSN: 1098-2418. DOI: 10.1002/(SICI)1098-2418(199809)13:2<159::AID-RSA3>3.0.CO;2-Q. URL: http://dx.doi.org/10.1002/(SICI)1098-2418(199809)13:2%3C159::AID-RSA3%3E3.0.CO;2-Q.

[3] Attahiru Sule Alfa. "Algorithmic analysis of the BMAP/D/k system in discrete time". In: *Adv. in Appl. Probab.* 35.4 (Dec. 2003), pp. 1131–1152. DOI: 10.1239/aap/1067436338. URL: http://dx.doi.org/10.1239/aap/1067436338.

[4] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. "Time-space trade-offs in population protocols". In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2017, pp. 2560–2579.

[5] 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)*. 2015, pp. 47–56.

[6] Noga Alon, Chen Avin, M Koucký, Michal Koucky, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. "Many Random Walks Are Faster Than One". In: *Combinatorics, Probability and Computing* (2007), p. 15. ISSN: 0963-5483. DOI: 10.1017/S0963548311000125. eprint: 0705.0467. URL: http://arxiv.org/abs/0705.0467.

[7] Dana Angluin, James Aspnes, and David Eisenstat. "A simple population protocol for fast robust approximate majority". In: *Distributed Computing* 21.2 (2008), pp. 87–102.

[8] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. "The computational power of population protocols". In: *Distributed Computing* 20.4 (2007), pp. 279–304.

[9] James Aspnes and Eric Ruppert. "An Introduction to Population Protocols". In: *Bulletin of the EATCS* 93 (2007), pp. 98–117.

[10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. "The Internet of Things: A survey". In: *Computer Networks* 54.15 (2010), pp. 2787–2805. ISSN: 1389-1286. DOI: http://doi.org/10.1016/j.comnet.2010.05.010. URL: http://www.sciencedirect.com/science/article/pii/S1389128610001568.

[11] 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*. 2008, pp. 121–132.

[12] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. "Balanced Allocations". In: *SIAM Journal on Computing* 29.1 (1999), pp. 180–200. DOI: 10.1137/S0097539795288490.

[13] Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Gustavo Posta. "Self-Stabilizing Repeated Balls-into-Bins". In: *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*. Ed. by Guy E.

Blelloch and Kunal Agrawal. ACM, 2015, pp. 332–339. ISBN: 978-1-4503-3588-1. DOI: 10.1145/2755573.2755584. URL: http://doi.acm.org/10.1145/2755573.2755584.

[14] 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)*. 2014, pp. 247–256.

[15] 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)*. 2015, pp. 371–390.

[16] 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*. IEEE, 2009, pp. 3661–3664.

[17] Florence Bénézit, Patrick Thiran, and Martin Vetterli. "The Distributed Multiple Voting Problem". In: *J. Sel. Topics Signal Processing* 5.4 (2011), pp. 791–804.

[18] Petra Berenbrink, Colin Cooper, Tom Friedetzky, Tobias Friedrich, and Thomas Sauerwald. "Randomized diffusion for indivisible loads". In: *J. Comput. Syst. Sci.* 81.1 (2015), pp. 159–185.

[19] Petra Berenbrink, Artur Czumaj, Matthias Englert, Tom Friedetzky, and Lars Nagel. "Multiple-Choice Balanced Allocation in (Almost) Parallel". English. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Ed. by Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio. Vol. 7408. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 411–422. ISBN: 978-3-642-32511-3. DOI: 10.1007/978-

3-642-32512-0_35. URL: http://dx.doi.org/10.1007/978-3-642-32512-0_35.

[20] Petra Berenbrink, Artur Czumaj, Tom Friedetzky, and Nikita D. Vvedenskaya. "Infinite Parallel Job Allocation (Extended Abstract)". In: *Proceedings of the Twelfth Annual ACM Symposium on Parallel Algorithms and Architectures*. SPAA '00. Bar Harbor, Maine, USA: ACM, 2000, pp. 99–108. ISBN: 1-58113-185-2. DOI: 10.1145/341800.341813. URL: http://doi.acm.org/10.1145/341800.341813.

[21] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. "Balanced Allocations: The Heavily Loaded Case". In: *SIAM Journal on Computing* 35.6 (2006), pp. 1350–1385. DOI: 10.1137/S009753970444435X.

[22] Petra Berenbrink, Robert Elsässer, and Tom Friedetzky. "Efficient randomised broadcasting in random regular networks with applications in peer-to-peer systems". In: *Distributed Computing* 29.5 (2016), pp. 317–339.

[23] 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)*. to appear. 2016.

[24] Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, Lars Nagel, and Christopher Wastell. "Self-stabilizing Balls into Bins in Batches: The Power of Leaky Bins [Extended Abstract]". In: *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. PODC '16. Chicago, Illinois, USA: ACM, 2016, pp. 83–92. ISBN: 978-1-4503-3964-3. DOI: 10.1145/2933057.2933092. URL: http://doi.acm.org/10.1145/2933057.2933092.

[25] Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, and Chris Wastell. "Plurality Consensus in Arbitrary Graphs: Lessons Learned from Load Balancing". In: *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*. Ed. by Piotr Sankowski and

Christos D. Zaroliagis. Vol. 57. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, 10:1–10:18. ISBN: 978-3-95977-015-6. DOI: 10.4230/LIPIcs.ESA.2016.10. URL: http://dx.doi.org/10.4230/LIPIcs.ESA.2016.10.

[26] Petra Berenbrink, Kamyar Khodamoradi, Thomas Sauerwald, and Alexandre Stauffer. "Balls-into-bins with Nearly Optimal Load Distribution". In: *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA '13. Montréal, Québec, Canada: ACM, 2013, pp. 326–335. ISBN: 978-1-4503-1572-2. DOI: 10.1145/2486159.2486191. URL: http://doi.acm.org/10.1145/2486159.2486191.

[27] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. "Randomized gossip algorithms". In: *IEEE Transactions on Information Theory* 52.6 (2006), pp. 2508–2530. ISSN: 00189448.

[28] Luca Cardelli and Attila Csikász-Nagy. "The cell cycle switch computes approximate majority". In: *Scientific reports* 2 (2012).

[29] Keren Censor-Hillel, Bernhard Haeupler, Jonathan a. Kelner, and Petar Maymounkov. "Global Computation in a Poorly Connected World: Fast Rumor Spreading with No Dependence on Conductance". In: *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*. 2011, pp. 961–970. ISBN: 9781450312455. DOI: 10.1145/2213977.2214064. arXiv: 1104.2944. URL: http://arxiv.org/abs/1104.2944.

[30] Jen-Yeu Chen and Gopal Pandurangan. "Optimal gossip-based aggregate computation". In: *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*. ACM. 2010, pp. 124–133.

[31] Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. "Programmable chemical controllers made from DNA". In: *Nature nanotechnology* 8.10 (2013), pp. 755–762.

[32] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. "Almost tight bounds for rumour spreading with conductance". In: *Proceedings of the 42nd ACM symposium on Theory of Computing* (2010), pp. 1–20.

[33] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. "Rumor spreading in social networks". In: *Theoretical Computer Science* 412.24 (2011), pp. 2602–2610. ISSN: 03043975. DOI: 10.1016/j.tcs.2010.11.001. arXiv: arXiv:1102.1487v1. URL: http://dx.doi.org/10.1016/j.tcs.2010.11.001.

[34] Andrea E. F. Clementi, Miriam Di Ianni, Giorgio Gambosi, Emanuele Natale, and Riccardo Silvestri. "Distributed community detection in dynamic graphs". In: *Theor. Comput. Sci.* 584 (2015), pp. 19–41.

[35] Colin Cooper, Robert Elsässer, and Tomasz Radzik. "The Power of Two Choices in Distributed Voting". In: *Automata, Languages, and Programming - 41st International Colloquium, (ICALP)*. 2014, pp. 435–446.

[36] 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)*. 2015, pp. 248–262. DOI: 10.1007/978-3-662-48653-5_17.

[37] A. Czumaj and V. Stemann. "Randomized allocation processes". In: *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*. Oct. 1997, pp. 194–203. DOI: 10.1109/SFCS.1997.646108.

[38] Artur Czumaj. "Recovery Time of Dynamic Allocation Processes". In: *Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures*. SPAA '98. Puerto Vallarta, Mexico: ACM, 1998, pp. 202–211. ISBN: 0-89791-989-0. DOI: 10.1145/277651.277686. URL: http://doi.acm.org/10.1145/277651.277686.

[39] Sebastian Daum, Fabian Kuhn, and Yannic Maus. "Rumor Spreading with Bounded In-Degree". In: 336495.336495 (2015). ISSN: 16113349. eprint: 1506.00828. URL: http://arxiv.org/abs/1506.00828.

[40] Alan Demers, Dan Greene, Carl Houser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. "Epidemic algorithms for replicated database maintenance". In: *ACM SIGOPS Operating Systems Review* 22.1 (1988), pp. 8–32. ISSN: 01635980. DOI: 10.1145/43921.43922.

[41] Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. "Social Networks Spread Rumors in Sublogarithmic Time". In: *Electronic Notes in Discrete Mathematics* 38 (2011), pp. 303–308. ISSN: 15710653. DOI: 10.1016/j.endm.2011.09.050. URL: http://dx.doi.org/10.1016/j.endm.2011.09.050.

[42] 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)*. 2011, pp. 149–158.

[43] Benjamin Doerr and Marvin Künnemann. "Tight Analysis of Randomized Rumor Spreading in Complete Graphs". In: *2014 Proceedings of the Eleventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. 2014, pp. 82–91.

[44] Moez Draief and Milan Vojnovic. "Convergence Speed of Binary Interval Consensus". In: *SIAM J. Control and Optimization* 50.3 (2012), pp. 1087–1109.

[45] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. ISBN: 978-0-521-88427-3. URL: http://www.cambridge.org/gb/knowledge/isbn/item2327542/.

[46] Devdatt P. Dubhashi and Desh Ranjan. "Balls and bins: A study in negative dependence". In: *Random Struct. Algorithms* 13.2 (1998), pp. 99–124.

[47] Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Frederik Mallmann-Trenn, and Horst Trinker. "Efficient k-Party Voting with Two Choices". In: *CoRR* abs/1602.04667 (2016). URL: http://arxiv.org/abs/1602.04667.

[48] Robert Elsässer and Thomas Sauerwald. "Broadcasting vs. mixing and information dissemination on Cayley graphs". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 2007, pp. 163–174.

[49] G. Fayolle, V.A. Malyshev, and M.V. Menshikov. *Topics in the Constructive Theory of Countable Markov Chains*. Cambridge University Press, 1995. ISBN: 9780521461979. URL: https://books.google.ca/books?id=lTJltFEnnHcC.

[50] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. "Randomized broadcast in networks". In: *Random Structures & Algorithms* 1.4 (1990), pp. 447–460.

[51] Nikolaos Fountoulakis, Anna Huber, and Konstantinos Panagiotou. "Reliable broadcasting in random networks and the effect of density". In: *Proceedings - IEEE INFOCOM* (2010), pp. 1–9. ISSN: 0743166X. DOI: 10.1109/INFCOM.2010.5462084.

[52] Nikolaos Fountoulakis and Konstantinos Panagiotou. "Rumour Spreading on Random Regular Graphs and Expanders". In: *Random Structures & Algorithms* 43.2 (2013), pp. 201–220.

[53] Nikolaos Fountoulakis, Konstantinos Panagiotou, and Thomas Sauerwald. "Ultra-fast rumor spreading in social networks". In: *Proceedings of the Twenty- ...* (2012), pp. 1642–1660. URL: http://dl.acm.org/citation.cfm?id=2095246.

[54] A.M. Frieze and G.R. Grimmett. *The shortest-path problem for graphs with random arc-lengths*. 1985.

[55] Mohsen Ghaffari and Calvin Newport. "How to Discreetly Spread a Rumor in a Crowd". In: *Distributed Computing (DISC)* 2.5 (2016), pp. 623–626. ISSN: 0302-9743.

[56] Mohsen Ghaffari and Merav Parter. "A Polylogarithmic Gossip Algorithm for Plurality Consensus". In: *PODC '16 Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing 16* 1 (2016), pp. 117–126.

[57] Bhaskar Ghosh and S. Muthukrishnan. "Dynamic Load Balancing by Random Matchings". In: *J. Comput. Syst. Sci.* 53.3 (1996), pp. 357–370.

[58] George Giakkoupis. "Tight bounds for rumor spreading in graphs of a given conductance". In: *STACS11*. 2011, pp. 1–17.

[59] George Giakkoupis. "Tight Bounds for Rumor Spreading with Vertex Expansion". In: *SODA14*. 2013, pp. 1–27. eprint: 1302.6243.

[60] George Giakkoupis and Thomas Sauerwald. "Rumor spreading and vertex expansion". In: *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (SODA'12)*. 2012, pp. 1623–1641.

[61] Gaston H. Gonnet. "Expected Length of the Longest Probe Sequence in Hash Code Searching". In: *J. ACM* 28.2 (Apr. 1981), pp. 289–304. ISSN: 0004-5411. DOI: 10.1145/322248.322254. URL: http://doi.acm.org/10.1145/322248.322254.

[62] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete mathematics - a foundation for computer science (2. ed.)* Addison-Wesley, 1994. ISBN: 978-0-201-55802-9.

[63] B. Hajek. "Hitting-Time and Occupation-Time Bounds Implied by Drift Analysis with Applications". In: *Advances in Applied Probability* 14.3 (), pp. 502–525.

[64] Juraj Hromkovič, Ralf Klasing, Andrzej Pelc, Peter Ruzicka, and Walter Unger. *Dissemination of information in communication networks: broadcasting, gossiping, leader election, and fault-tolerance*. Springer Science & Business Media, 2005.

[65] A.E. Kamal. "Efficient solution of multiple server queues with application to the modeling of ATM concentrators". In: *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*. Vol. 1. Mar. 1996, 248–254 vol.1. DOI: 10.1109/INFCOM.1996.497900.

[66] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. "Randomized rumor spreading". In: *Proceedings 41st Annual Symposium on Foundations of Computer Science* (2000). ISSN: 0272-5428. DOI: 10.1109/SFCS.2000.892324.

[67] David Kempe, Alin Dobra, and Johannes Gehrke. "Gossip-Based Computation of Aggregate Information". In: *Proceedings 44th Symposium on Foundations of Computer Science (FOCS)*. 2003, pp. 482–491.

[68] Nam K Kim, Mohan L Chaudhry, Bong K Yoon, and Kilhwan Kim. "A Complete and Simple Solution to a Discrete-Time Finite-Capacity BMAP/D/c Queue". In: *Applied Mathematics* 3.12 (2012), p. 2169.

[69] Marcos Kiwi and Christopher Thraves Caro. "FIFO Queues are Bad for Rumor Spreading". In: *IEEE Transactions on Information Theory* (2016).

[70] 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)*. 2008, pp. 15–24.

[71] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. "Statistical properties of community structure in large social and information networks". In: *Proceedings of the 17th international conference on World Wide Web*. ACM. 2008, pp. 695–704.

[72] David A. Levin and Yuval Perres. *Markov Chains and Mixing Times*. American Mathematical Society, Dec. 2008. ISBN: 978-0-8218-4739-8.

[73] Florian Meier and Ueli Peter. "Push Is Fast On Sparse Random Graphs". In: *SIAM Journal on Discrete Mathematics* 51.6 (2017), pp. 3232–3258. DOI: 10.1137/090745854.

[74] 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)*. 2014, pp. 871–882.

[75] Milena Mihail, Christos Papadimitriou, and Amin Saberi. "On certain connectivity properties of the internet topology". In: *Journal of Computer and System Sciences* 72.2 (2006), pp. 239–251.

[76] Michael Mitzenmacher. "The Power of Two Choices in Randomized Load Balancing". In: *IEEE Trans. Parallel Distrib. Syst.* 12.10 (2001), pp. 1094–1104. DOI: 10.1109/71.963420. URL: http://doi.ieeecomputersociety.org/10.1109/71.963420.

[77] Michael Mitzenmacher and Eli Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005. ISBN: 978-0-521-83540-4.

[78] Damon Mosk-Aoyama and Devavrat Shah. "Computing Separable Functions via Gossip". In: *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*. Denver, Colorado, USA, 2006, pp. 113–122. ISBN: 1595933840. DOI: 10.1145/1146381.1146401.

[79] Elchanan Mossel, Joe Neeman, and Omer Tamuz. "Majority dynamics and aggregation of information in social networks". In: *Autonomous Agents and Multi-Agent Systems* 28.3 (2014), pp. 408–429.

[80] Elchanan Mossel and Grant Schoenebeck. "Reaching Consensus on Social Networks". In: *Innovations in Computer Science - (ICS)*. 2010, pp. 214–229.

[81] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. ISBN: 0-521-47465-5.

[82] David Peleg. "Local majorities, coalitions and monopolies in graphs: a review". In: *Theor. Comput. Sci.* 282.2 (2002), pp. 231–257.

[83] Yuval Peres, Kunal Talwar, and Udi Wieder. "The $(1+\beta)$-choice Process and Weighted Balls-into-Bins". In: *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SODA '10. Austin, Texas: Society for Industrial and Applied Mathematics, 2010, pp. 1613–1619. ISBN: 978-0-898716-98-6.

[84] 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)*. 2009, pp. 2527–2535.

[85] Boris Pittel. "On Spreading a Rumour". In: *SIAM Journal on Applied Mathematics* 47.1 (1987), pp. 245–255.

[86] Martin Raab and Angelika Steger. ""Balls into Bins" - A Simple and Tight Analysis". In: *Randomization and Approximation Techniques in Computer Science, Second International Workshop, RANDOM'98, Barcelona, Spain, October 8-10, 1998, Proceedings*. Ed. by Michael Luby, José D. P. Rolim, and Maria J. Serna. Vol. 1518. Lecture Notes in Computer Science. Springer, 1998, pp. 159–170. ISBN: 3-540-65142-X. DOI: 10.1007/3-540-49543-6_13. URL: http://dx.doi.org/10.1007/3-540-49543-6_13.

[87] Kenneth H. Rosen, John G. Michaels, Jonathan L. Gross, Jerrold W. Grossman, and Douglas R. Shier, eds. *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 1999.

[88] Thomas Sauerwald. "Randomized protocols for information dissemination". PhD thesis. University of Paderborn, Germany, 2008. URL: http://ubdok.uni-paderborn.de/servlets/DerivateServlet/Derivate-7251/Dissertation%20Sauerwald.pdf.

[89] Thomas Sauerwald. "On mixing and edge expansion properties in randomized broadcasting". In: *Algorithmica (New York)* 56 (2010), pp. 51–88. DOI: 10.1007/s00453-008-9245-4.

[90] Thomas Sauerwald and Alexandre Stauffer. "Rumor Spreading and Vertex Expansion on Regular Graphs". In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. 2011, pp. 462–475. ISBN: 9780898719932.

[91] 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)*. 2012, pp. 341–350.

[92] Thomas Sauerwald and He Sun. “Tight Bounds for Randomized Load Balancing on Arbitrary Network Topologies”. In: *CoRR* abs/1201.2715 (2012). full version of FOCS’12. arXiv: 1201.2715 [cs.DM].

[93] Khosrow Sohraby and Ji Zhang. “Spectral decomposition approach for transient analysis of multi-server discrete-time queues”. In: *INFOCOM’92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*. IEEE. 1992, pp. 395–404.

[94] Volker Stemann. “Parallel Balanced Allocations”. In: *Proceedings of the Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*. SPAA ’96. Padua, Italy: ACM, 1996, pp. 261–269. ISBN: 0-89791-809-6. DOI: 10.1145/237502.237565. URL: http://doi.acm.org/10.1145/237502.237565.

[95] Kunal Talwar and Udi Wieder. “Balanced Allocations: A Simple Proof for the Heavily Loaded Case”. In: *CoRR* abs/1310.5367 (2013). URL: http://arxiv.org/abs/1310.5367.

[96] Kunal Talwar and Udi Wieder. “Balanced Allocations: A Simple Proof for the Heavily Loaded Case”. English. In: *Automata, Languages, and Programming*. Ed. by Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias. Vol. 8572. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 979–990. ISBN: 978-3-662-43947-0. DOI: 10.1007/978-3-662-43948-7_81. URL: http://dx.doi.org/10.1007/978-3-662-43948-7_81.