# Efficient Randomized Algorithms for Information Dissemination, Distributed Voting, and Plurality Consensus

**PhD Thesis**

# Efficient Randomized Algorithms for Information Dissemination, Distributed Voting, and Plurality Consensus

Dominik S. Kaaser

November 2016
MMXVI

Dissertation submitted to the Faculty of Natural Sciences
at the University of Salzburg in partial fulfillment of the
requirements for the doctoral degree Dr. techn.

Supervisor: Univ.-Prof. Dr. Robert Elsässer
Department of Computer Sciences
University of Salzburg

**Department of Computer Sciences**
University of Salzburg
Jakob-Haringer-Straße 2
5020 Salzburg
Austria

**Dominik S. Kaaser**
*Efficient Randomized Algorithms for Information Dissemination, Distributed Voting, and Plurality Consensus*
November 2016 · commit d0a6b08

*to the reader*

# Contents

# Preface

Moore's law, named after Gordon Moore, co-founder of Intel, describes the observation that the number of transistors on integrated circuits doubles approximately every two years. The prediction proved accurate for several decades, and progress in digital electronics is strongly linked to Moore's law. A similar behavior can be observed in the peak performance of high performance computing systems ranked in the TOP500 list, which over the last decades also grew exponentially.

In contrast, in the same period the bandwidth of computer networks has grown linearly over time.

Given that the National University of Defense Technology, China, announced plans to build an *exascale* cluster system which targets a peak performance of $10^{18}$ flops, and that such a system expectedly suffers from a tremendously large rate of unrecoverable errors, there is obviously a need for efficient and robust communication protocols. These protocols should solve various communication tasks efficiently in terms of a small communication overhead and robustly, allowing convergence even under presence of a relatively large number of errors.

The main goal of my thesis is to devise such protocols and to analyze their performance in a theoretical framework. The main ingredients are so-called randomized algorithms. Therefore, a significant part of my work also deals with the analysis of stochastic processes, with the goal to show that with a *high probability* the proposed protocols indeed give correct results.

The thesis was created while I worked in the Efficient Algorithms Group at the University of Salzburg under the supervision of Robert Elsässer. It contains our results over the last three years. While some of these results have already been presented at international conferences, this thesis contains the full versions of the papers including all proofs and, to some extent, additional content.

For full understanding of the presented results, the well-disposed reader is expected to have basic knowledge in algorithm analysis and probability theory, as the fundamental concepts from these areas will not be covered in this thesis. However, in the first part a short introduction is given to those stochastic techniques which are well-established in the scientific community but might be missing in a standard computer science curriculum.

## Acknowledgment

It is my pleasure to take this opportunity to express my gratitude and thank everyone who encouraged and supported me throughout the time of writing this thesis. In particular, I am grateful to my PhD advisor Robert Elsässer for giving me the opportunity to work, study, and research under his supervision in the Efficient Algorithms Group at the University of Salzburg. Our cooperation forms the foundation of the results presented in this thesis and his guidance was an invaluable contribution to my scientific and personal development.

Besides my supervisor, I would like to thank Tomasz Radzik for reviewing my thesis and all members of the dissertation committee for their valued feedback. I also thank Petra Berenbrink for the great times in Vancouver and in Hamburg, where she provided me the opportunity to work with her research group. I very much enjoyed – and still enjoy – our collaboration.

Furthermore, I would like to thank my fellow PhD students, co-authors, and friends Frederik Mallman-Trenn, Emanuele Natale, and Peter Palfrader. The last years would not have been the same without the stimulating discussions, the sleepless nights when approaching deadlines, and all the fun we have had.

I also wish to thank Andreas Bilke and Günther Eder, as well as all friends and colleagues from the Department of Computer Sciences at the University of Salzburg. Over the last years, we have shared many experiences and it has always been a pleasure to work with them.

I thank my parents and my family for their support throughout my entire educational career.

Last but not the least, I would like to express my sincere gratitude to my girlfriend Linda Schallmoser for her unconditional support, understanding patience and loving encouragement, always.

Salzburg, Austria
November 2016

Dominik S. Kaaser

Parts of this thesis were supported by the Austrian Science Fund (FWF) under project numbers P 25214 and P 27613.

# Abstract

In this thesis, we analyze randomized algorithms to solve various communication tasks in distributed systems. The problems we discuss are information dissemination, load balancing, distributed voting and plurality consensus.

Information dissemination is a fundamental problem in parallel and distributed computing. In the broadcasting problem, a single message has to be spread among all nodes of a graph. A prominent communication protocol for this problem is based on the so-called random phone call model (Karp et al., FOCS 2000). In each step, every node opens a communication channel to a randomly chosen neighbor, which can then be used for bi-directional communication.

Berenbrink et al., ICALP 2010, considered the so-called gossiping problem in the random phone call model, where each node starts with its own message and all messages have to be disseminated to all nodes in the network. It is known that the bound on the number of message transmissions produced by randomized broadcasting in complete graphs cannot be achieved in sparse graphs even if they have best expansion and connectivity properties. In the first result, we analyze whether a similar influence of the graph density also holds w.r.t. the performance of gossiping. We study analytically and empirically the communication overhead generated by gossiping algorithms in the random phone call model in random graphs. We further consider simple modifications of the random phone call model in these graphs. Our results indicate that, unlike in broadcasting, there seems to be no significant difference between the performance of randomized gossiping in complete graphs and sparse random graphs.

The problem of diffusion-based load balancing is defined as follows. We are given an interconnection network and a number of load items, which are arbitrarily distributed among the nodes of the network. The goal is to redistribute the load in iterative discrete steps such that at the end each node has (almost) the same number of items. In diffusion load balancing, nodes are only allowed to balance their load with their direct neighbors.

In our second result, we show empirically that second order schemes, which are usually much faster than first order schemes, will not balance the load completely on a number of networks within reasonable time. However, the maximum load difference at the end seems to be bounded by a constant value, which can be further decreased if a first order scheme is applied once this value is achieved by second order scheme.

In the deterministic binary majority process we are given a simple graph where each node has one out of two initial opinions. In every round, each node adopts the majority opinion among its neighbors. It is known that this process always converges in $\mathrm{O}(|E|)$ rounds to a two-periodic state in which every node either keeps its opinion or changes it in every round.

In our third result, we show that it is NP-hard to decide whether there exists an initial opinion assignment for which it takes more than $k$ rounds to converge to the two-periodic state, for a given integer $k$. We then identify suitable modules of a graph $G$ to obtain a new graph $G^{\Delta}$ that can be computed in linear time. The worst-case convergence time of $G^{\Delta}$ is an upper bound on that of $G$. Our new bounds asymptotically improve the best known bounds for various graph classes.

In our final result, we consider the problem of distributed plurality consensus. We assume the presence of $k$ distinct opinions in the complete graph. In the most basic two-choices process we are given a graph in which initially every node holds one of $k$ different opinions. In each step, every node chooses two neighbors uniformly at random. If the opinions of the two neighbors coincide, then this opinion is adopted.

We show that if $k = \mathrm{O}(n^{\varepsilon})$ for some small $\varepsilon$, then this protocol converges to the initial majority within $\mathrm{O}(k \cdot \log n)$ steps, with high probability, as long as the initial difference between the largest and second largest opinion is $\Omega(\sqrt{n \log n})$.

To further reduce the run time, we combine the two-choices process with a simple rumor spreading algorithm and obtain a significantly faster algorithm. Our main contribution is the adaption of this algorithm to the asynchronous setting, where only one randomly chosen node per step performs at most two queries. If the difference between the two largest opinions is at least $\Omega(b)$ where $b$ is the size of the second largest opinion, and $k = \exp\Big(\mathrm{O}\Big(\log n / \log^2 \log n\Big)\Big)$, then our algorithm achieves the best possible run time of $\mathrm{O}(\log n)$ in the model where a node is allowed to communicate with at most constantly many other nodes per step.

# Introduction

1

# Introduction

## 1.1 Distributed Systems

A *distributed system* is a collection of computing devices that interact via a communication network in order to solve a common problem. The main goal is to improve performance by solving suitable subproblems in parallel, while at the same time the large number of compute nodes naturally provides robustness against failures in individual components. This general definition covers a wide range of modern computer systems, from weakly coupled systems such as sensor networks to tightly coupled systems such as shared-memory multi-processor systems. We will, however, focus in this thesis on the more loosely coupled range. More precisely, we consider distributed systems subject to the following limitations.

**Lack of common memory.** We assume that the computing devices do not share a common memory. They rather communicate by passing messages over the communication network.

**Lack of common clock.** Closely related to the lack of common memory, we also assume that the computing devices do not have access to a global clock. For some of our results, however, this assumption is weakened in the following sense. We assume that the nodes start the distributed computation at the same time and thus a weaker notion of common time exists or, to some extent, can be established. Furthermore, in theoretical models of distributed systems it is often assumed that all nodes operate in synchronous rounds.

**Lack of network structure.** While in classical computer networks a client-server architecture is well established, for distributed systems the peer-to-peer computing paradigm has become increasingly popular. In particular, designated control-structures usually form single points of failure and therefore hierarchical network structures are avoided when designing robust and scalable systems. Again, this constraint is weakened for some of our results. We therefore complement our findings with an elaborate robustness analysis.

**Heterogeneity.** In contrast to classical high performance computing, where a set of tightly coupled homogeneous compute nodes is required, we assume that in distributed computing we are given a collection of heterogeneous systems. In particular, the computing devices may significantly vary in processing speeds and/or memory.

For a more detailed introduction to distributed systems, see, e.g., the textbooks by Attiya and Welch [AW04] and Kshemkalyani and Singhal [KS08].

The benefits of distributed computing systems allow improved performance and resilience against component failure. However, they may come at a high cost. Because of the lack of common memory, each component can only be aware of the information that it acquires during the execution of an algorithm. It therefore has only a limited, local view of the global state. From the lack of a common clock and the heterogeneity of the compute nodes it furthermore follows that the system operates in an asynchronous manner. Thus the times at which events such as the receiving of a message occur cannot be known precisely. Finally, due to the lack of network structures known a priori, even simple tasks such as broadcasting, that is, sending a message to all other nodes, cannot be done as efficiently as if the network structure was known.

Our main contribution in this work is to give efficient solutions to fundamental problems that arise in distributed systems such as information dissemination or consensus. To present a rigorous analysis of the correctness of our algorithms, we first formally describe the underlying model of the distributed system in Section 2.1. The remainder of this work is organized as follows.

## 1.2 Organization

Following this introductory part, this work is organized into four main parts. At the beginning of each part we give a short overview over the problem statement, the related work, and our contributions. In this introduction, we will give a formal definition of our model of a distributed system. We furthermore introduce the basic mathematical tools that we use in the analysis of randomized algorithms.

The first part, Part I, presents our results on randomized gossiping algorithms for efficient information dissemination in large networks. In Part II, we present our results on diffusion-based load balancing algorithms. Finally, in Part III and in Part IV we analyze distributed voting and plurality consensus processes.

## 1.3 Publications and Contributions

This thesis is based on four individual papers, three of which were accepted at peer-reviewed conferences. The following table, Table 1.1, gives an overview over the author's contribution to all four papers. In this table, we identify for each part of this thesis the main theorems and give a description of the author's contribution to the corresponding chapters.

### Contributions

| Part | Chapter | Result | Contribution |
|---|---|---|---|
| Part I [EK15] | Chapter 6 | Theorem 1 | While the author was the main contributor to this result, the proofs were developed in cooperation with and under supervision of the author's PhD advisor. |
| | Chapter 7 | Theorem 25<br>Theorem 26 | There has been only a limited contribution by the author to the theoretical analysis of the memory model. |
| | Chapter 8 | Simulations | The simulation software was developed and the empirical analysis was conducted by the author. |
| Part II [ABEK15] | Chapter 10 | Theorem 31<br>Theorem 34<br>Theorem 37 | The contributions by the author to the theoretical work in Chapter 10 were only limited. Note that in Chapter 10 we only state those results from [ABEK15] which are required for understanding the remainder of the part. This chapter should not be accounted to the contributions of the author and to the contribution of this thesis. |
| | Chapter 11 | Simulations | The author's main contribution to this topic is the empirical analysis presented in Chapter 11. The simulation software was implemented and the empirical analysis was conducted by the author, including data presentation and explanations in the chapter. |

| Part | Chapter | Result | Contribution |
|---|---|---|---|
| Part III [KMN16] | Chapter 13 | Theorem 38 | The author is the main contributor to this chapter. |
| | Chapter 14 Chapter 15 | Theorem 39 | The results were developed with equal contributions while Frederik Mallmann-Trenn and Emanuele Natale were visiting the University of Salzburg in May 2015 and February 2015, respectively. |
| Part IV | Chapter 17 | Theorem 61 | The results were developed together with Robert Elsässer, Tom Friedetzky, Frederik Mallmann-Trenn, and Horst Trinker, while Tom Friedetzky and Frederik Mallmann-Trenn were visiting the University of Salzburg in May 2015. |
| | Chapter 18 | Theorem 62 | The author is the main contributor to this result. |
| | Chapter 19 | Theorem 63 | The results were developed together with Frederik Mallmann-Trenn and Robert Elsässer, with equal contributions, while Frederik Mallmann-Trenn was visiting the University of Salzburg in May 2016. |
| | Chapter 20 | Simulations | The author implemented a preliminary simulation software which provided the initial impulse for this research. Additionally, Gregor Bankhamer conducted extensive simulations which, however, were not included in this thesis. |

**Table 1.1:** contributions of the author to various chapters

### Presentations

All of the papers which had been accepted to peer-reviewed conferences were presented by the author of this thesis at the corresponding conferences. A list of presentations given by the author at these conferences and at various workshops is given in Table 1.2.

| Date<br>Conference or Workshop | Location | Reference |
|---|---|---|
| July 8 – 9, 2014<br>International Workshop on Algorithms and Software for Scientific Computing | Vienna, Austria | [EK15] |
| May 25 – 29, 2015<br>29th IEEE International Parallel & Distributed Processing Symposium | Hyderabad, India | |
| June 29 – July 2, 2015<br>35th IEEE International Conference on Distributed Computing Systems | Columbus, Ohio, USA | [ABEK15] |
| February 22 – 24, 2016<br>Austrian HPC Meeting 2016 | Grundlsee, Austria | |
| October 5 – 8, 2015<br>29th International Symposium on Distributed Computing | Tokyo, Japan | [KMN15] |
| August 22 – 26, 2016<br>41st International Symposium on Mathematical Foundations of Computer Science | Kraków, Poland | [KMN16] |

**Table 1.2:** presentations given by the author

### Peer-Reviewed Publications

The following list contains a detailed record of all publications which were accepted to peer-reviewed conferences and/or journals during the time of writing this thesis.

[BHH+13] Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader: *Weighted Straight Skeletons in the Plane.* In *Proceedings of the 25th Canadian Conference on Computational Geometry (CCCG)*, 2013, pages 13–18.

[HK13] Martin Held and Dominik Kaaser: *Curvature-Continuous Approximation of Planar Curvilinear Profiles.* In *Proceedings of the Computer Aided Design Conference*, 2013, pages 88–89.

[BHH+14] Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader: *Straight Skeletons of Monotone Polygons.* In *Proceedings of the 30th European Workshop on Computational Geometry (EuroCG)*, 2014.

[HK14] Martin Held and Dominik Kaaser: *C2 Approximation of Planar Curvilinear Profiles by Cubic B-Splines*. In *Computer-Aided Design and Applications*, volume 11 (2), 2014, pages 206–219. DOI: 10.1080/16864360.2014.846092.

[ABEK15] Hoda Akbari, Petra Berenbrink, Robert Elsässer, and Dominik Kaaser: *Discrete Load Balancing in Heterogeneous Networks with a Focus on Second-Order Diffusion*. In *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015, pages 497–506. DOI: 10.1109/ICDCS.2015.57.

[BHH+15a] Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader: *A Simple Algorithm for Computing Positively Weighted Straight Skeletons of Monotone Polygons*. In *Information Processing Letters*, volume 115 (2), 2015, pages 243–247. DOI: 10.1016/j.ipl.2014.09.021.

[BHH+15b] Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader: *Weighted Straight Skeletons in the Plane*. In *Computational Geometry: Theory and Applications*, volume 48 (2), 2015, pages 120–133. DOI: 10.1016/j.comgeo.2014.08.006.

[EK15] Robert Elsässer and Dominik Kaaser: *On the Influence of Graph Density on Randomized Gossiping*. In *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2015, pages 521–531. DOI: 10.1109/IPDPS.2015.32.

[KMN15] Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale: *Brief Announcement: On the Voting Time of the Deterministic Majority Process*. In *Proceedings of the 29th International Symposium on Distributed Computing (DISC)*, 2015.

[KMN16] Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale: *On the Voting Time of the Deterministic Majority Process*. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2016. DOI: 10.4230/LIPIcs.MFCS.2016.55.

2

# Algorithmic Ingredients

Hinc incipit algorismus.
Haec algorismus ars praesens dicitur in qua
talibus indorum fruimur bis quinque figuris
0. 9. 8. 7. 6. 5. 4. 3. 2. 1.

*(Alexander de Villa Dei,
Carmen de Algorismo, c. 1220)*

The Euclidean algorithm to compute the greatest common divisor of two integers is a prototypical example of an algorithm. In two simple variants, the algorithm may be specified in one of the following ways.

```
Algorithm gcd($a, b$)
    while $a \neq b$ do
        if $a > b$ then
            $a \leftarrow a - b$;
        else
            $b \leftarrow b - a$;
    return $a$;
```

```
Algorithm gcd($a, b$)
    while $b \neq 0$ do
        $c \leftarrow b$;
        $b \leftarrow a \mod b$;
        $a \leftarrow c$;
    return $a$;
```

The algorithm is named after the ancient Greek mathematician Euclid, who described it around 300 BC in the seventh book *Elementary Number Theory* of his Elements [Hea08]. The word *algorithm*, however, is not derived from the Greek roots αριϑμός for *number*. In fact, the modern word *algorithm* and the Greek word αριϑμός are false cognates. (The word *algorithm* is not derived from the Greek ἀλγος for *pain* either, as Erickson notes in his introduction to algorithms [Eri14].) The origin rather lies in the latinized name of the Persian mathematician Muḥammad ibn Mūsā al-Khwārizmī (Arabic: محمد بن موسى الخوارزمى) who lived c. 780 to 850 AD [BM91].

Al-Khwārizmī popularized the modern decimal system, which originated from India, for basic arithmetical operations. The initial quote to this chapter by Alexander de Villa Dei from his Carmen de Algorismo shows that around 1200, the name al-Khwārizmī had been latinized to *algorismi*. It had lost its original meaning and the word *algorismus* was used instead to denote arithmetic techniques in the decimal system. It took, however, until the second half of the 20th century until the advent of modern computer science, along with suitable computing hardware, again changed the meaning of the word *algorithm*.

In their book [CLRS09], Cormen, Leiserson, Rivest, and Stein define an algorithm informally as

> "[...] a sequence of computational steps that transform the input into the output. [CLRS09]

However, the article by Blass and Gurevich [BG03] shows that for a rigorous definition of the term *algorithm* an involved survey on the foundations of theoretical computer science is required. While such a survey is beyond the scope of this thesis, we will nevertheless define in the following section a basic model that allows us to specify and analyze our algorithms.

## 2.1 Models for Distributed Systems

The analysis of algorithms and data structures is a prominent area in computer science. Over the last decades, a generally accepted framework for specification and analysis of algorithms has been established. Especially for the analysis of sequential algorithms, the underlying machine model is vastly agreed upon.

In comparison, the analysis of algorithms for distributed systems is a relatively new field. From the broad definition of distributed systems it follows that defining a suitable machine model that eventually lends itself for a rigorous analysis is much more difficult. While in classical sequential algorithms the quantities of interest, namely run time and memory consumption, are easily recognized, this does not carry over to the analysis of distributed algorithms. In the analysis of distributed algorithms, apart from the overall run time and the memory consumption per node, also the communication complexity and the fault tolerance have to be considered [AW04]. For many problems in sequential models, optimal algorithms can be devised, while results for distributed systems often constitute a trade-off between run time and communication complexity.

In this thesis, we model the distributed system as a graph $G = (V, E)$ where $V$, the set of nodes, denotes the set of computing components of the distributed system and $E \subseteq V \times V$, the set of edges, represents the underlying communication network. We assume that a compute node $v \in V$ can directly communicate with $u \in V$ if and only if the edge $(v, u)$ is present in $E$.

The algorithms we investigate are defined in the so-called synchronous model. In this model, we assume that the execution of the algorithm runs synchronously in discrete rounds. (In Part IV we give an extension of the synchronous algorithm to an asynchronous model.) In the execution of an algorithm in the synchronous model, in every round, each node performs an action according to the algorithm's definition in parallel. Nodes may communicate with their direct neighbors, where we assume that they see the previous round's state. More formally, we assume that all nodes split their rounds into an update step and a commit step. In the update step, nodes compute their new state based on their current state and on their neighbor's state, and in the commit step, all nodes simultaneously adopt the new state.

## 2.2 Randomized Algorithms

In contrast to deterministic algorithms, so-called *randomized algorithms* use random bits as an additional input. These random bits are used at least once to make a random choices in the execution of the algorithm. While it seems unintuitive at best to rely on random decisions, there are many problems which can be solved much more efficiently using randomized algorithms compared to the best known deterministic solutions. Additionally, in many cases the randomized algorithms are much simpler and easier to implement.

> "From the highly theoretical notion of probabilistic theorem proving to the very practical design of PC Ethernet cards, randomness and probabilistic methods play a key role in modern computer science." [MU05]

As a consequence from relying on random decisions, it may happen that the output of a randomized algorithm is no longer deterministically defined upon the inputs, but rather is a random variable. As a consequence, these algorithms may return an incorrect result with a certain probability. The class of these algorithms is known as *Monte Carlo* algorithms. A classical example for a Monte Carlo algorithm is the Miller-Rabin primality test [Mil76, Rab80], which has a one-sided error probability. That means, the test either finds a *witness* for the compositeness of a given number $n$ and therefore returns that $n$ is definitely composite (note that this witness is not a non-trivial factor of $n$, but rather a base for which Fermat's little theorem is violated), or the test returns that $n$ is probably prime. Finally, by repeated application, the error probability can be significantly reduced.

As a second class of algorithms, it may also be the case that the run time (or any other resource) of a randomized algorithm is a random variable. In that case we speak of *Las Vegas* algorithms. Many authors require that (deterministic) algorithms always terminate, see, e.g., the work by Knuth [Knu97]. In the context of randomized algorithms, we require that the algorithm *eventually* terminates. The textbook example for such a Las Vegas algorithm is randomized quicksort.

Intuitively, it may seem unusual to devise algorithms that may return erroneous results or run for a very long time. However, if the probability that the algorithm show this undesired behavior is very small, the benefits of simpler algorithms that run fast in expectation may very well outweigh a small error probability [MU05]. In this thesis, one of our main tasks is to perform a probabilistic analysis of our algorithms. We will show by performing a careful analysis that our algorithms work correctly and efficiently *with high probability*. The notion of high probability along with the basic mathematical tools that we use for the analysis of randomized algorithms will be introduced in the following chapter.

# 3

# Stochastic Ingredients

The *phenomenon of concentration of measure* is a building block in the analysis of randomized algorithms. The underlying observation is that a function of a large number of random variables is concentrated in a relatively narrow range, under assumptions such as Lipschitz continuity and certain conditions on the dependence among the random variables [DP09]. We can use this phenomenon of concentration of measure in the analysis of randomized algorithms, and thereby

> "[...] we can argue that the observable behavior of randomized algorithms is 'almost deterministic'. In this way, we can obtain the satisfaction of deterministic results, and at the same time retain the benefits of randomized algorithms, namely their simplicity and efficiency." [DP09]

In this chapter, we give a formal introduction to some of the mathematical tools that we use for the analysis of randomized distributed algorithms in this thesis. Additionally, we will adhere to the following conventions.

## Conventions

To increase readability, we omit ceilings and floors from numbers that strictly should be integers (but are not), when the rounding error does not affect the algorithm or its analysis. For example, we may specify an algorithm to run for $\log n/\log\log n$ rounds instead of $\lfloor \log n/\log\log n \rfloor$ rounds.

The expression $\log n$ denotes $\log_2 n$, the logarithm to base 2, and $\ln n$ denotes the natural logarithm of $n$. Furthermore, $\log^k n$ denotes the exponentiation $(\log n)^k$ and $\log\log n$ denotes the composition $\log(\log(n))$.

Throughout this thesis, the expression *with high probability* means a probability of at least $1 - n^{-\Omega(1)}$, where $n$ is the input size. At various places, we will show that an event occurs with high probability, to later use union bound to conclude that the event occurs at a large number of nodes (or multiple rounds). We therefore assume that the exact exponent, hidden in the asymptotic $\Omega(1)$ notation, is large enough such that union bound again yields a high probability.

## 3.1 Union Bound

The following inequality is known as *Boole's inequality* or the *union bound.* It is very simple, but nevertheless tremendously useful [MU05]. It states that for any finite or countable set of stochastic events, the probability that at least one of the events occurs is bounded from above by the sum of the probabilities of the individual events. More formally, let $\{\mathcal{E}_1, \mathcal{E}_2, \dots\}$ be a finite or countable set of stochastic events. Then we have

$$\Pr\left[\bigcup_{i\geq 1} \mathcal{E}_i\right] \leq \sum_{i\geq 1} \Pr[\mathcal{E}_i] \enspace .$$

## 3.2 Chernoff Bounds

The Chernoff bound gives an exponentially decreasing bound on tail distributions of sums of independent Bernoulli random variables.

Let $X_i$ for $i \in \{1, \dots, n\}$ be indicator random variables defined as

$$X_i = \begin{cases} 1 & \text{with probability } p_i, \\ 0 & \text{otherwise,} \end{cases}$$

and let the random variable $X$ be defined as $X = \sum_{i=1}^{n} X_i$, the sum of $n$ independent Bernoulli trials, with expected value $\mu = \mathrm{E}[X]$. By applying Markov's inequality to the moment-generating function and by exploiting independence, one gets for a $\delta$ with $0 < \delta < 1$ [HR90]

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

and

$$\Pr[X \leq (1-\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \enspace .$$

In the remainder of this thesis, we will often use the following, slightly looser but much more convenient bounds [MU05, DP09].

$$\begin{aligned} &\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2\mu}{3}} && \text{for } 0 < \delta < 1 \\ &\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta\mu}{3}} && \text{for } 1 < \delta \\ &\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2\mu}{2}} && \text{for } 0 < \delta < 1 \end{aligned}$$

Furthermore, we will use the following monotonicity property. Let $\mu_L$ be a lower bound on $\mu$ and let $\mu_H$ be an upper bound on $\mu$ such that $\mu_L < \mu < \mu_H$. The following bounds hold [MU05, DP09].

$$\Pr[X \geq (1+\delta)\mu_H] \leq e^{-\frac{\delta^2 \mu_H}{3}} \qquad \text{for } 0 < \delta < 1 \tag{3.1}$$

$$\Pr[X \leq (1-\delta)\mu_L] \leq e^{-\frac{\delta^2 \mu_L}{2}} \qquad \text{for } 0 < \delta < 1 \tag{3.2}$$

While the Chernoff bounds given in (3.1) and (3.2) have proven very useful in the analysis of randomized algorithms, the limitation to independent Bernoulli trials is a major drawback. However, under certain circumstances these limitations can be overcome. More precisely, it can be shown that if the $X_i$s are negatively associated, above bounds still hold. See, e.g., the work by Dubhashi and Ranjan on balls and bins [DR98] for further examples of negative associations.

In the following section, we will present the Azuma-Hoeffding inequality, which can also be applied when the $X_i$s are not independent. We will give a short introduction to so-called *martingales* and the *method of bounded differences*. Together with Chernoff bounds, the Azuma-Hoeffding bound completes the box of stochastic tools used in our work.

## 3.3 The Azuma-Hoeffding Inequality

*Martingales*, a well-studied concept from classical probability theory, are defined as a sequence of random variables $\mathbf{X} = X_0, X_1, \ldots$ such that

$$\mathrm{E}[X_i | X_{i-1}, \ldots, X_0] = X_{i-1} \qquad \text{for } i \geq 1 \ .$$

For a review of conditional probabilities and expectations, see, e.g., the textbook by Motwani and Raghavan [MR95]. In our analysis, we focus on martingales with *bounded differences*, that is, sequences of random variables which satisfy the following condition.

Let $\mathbf{X} = X_0, X_1, \ldots$ be a martingale. We say that $\mathbf{X}$ satisfies the *bounded differences* condition with parameters $a_i, b_i \in \mathbb{R}$ if

$$a_i \leq X_i - X_{i-1} \leq b_i \qquad \text{for } i > 0 \ .$$

Based on above definition, the Azuma-Hoeffding inequality is stated as follows [DP09]. Let $\mathbf{X} = X_0, X_1, \ldots$ be a martingale that satisfies the bounded differences condition with parameters $a_i$ and $b_i$. Then

$$\left.\begin{array}{l} \Pr[X_n \geq X_0 + t], \\ \Pr[X_n \leq X_0 - t] \end{array}\right\} \leq \exp\left(\frac{-2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right) \qquad \text{for } i > 0 \ .$$

If we have $|X_i - X_{i-1}| \leq c_i$ for parameters $c_i \in \mathbb{R}$, it holds that [MR95]

$$\Pr[|X_n - X_0| \geq t] \leq 2 \exp\left(\frac{-t^2}{2\sum_{i=1}^{n} c_i^2}\right) \qquad \text{for } i > 0 \ .$$

In contrast to the Chernoff bounds given in the previous section, we do no longer require independence among the variables $X_i$. The assumption of independence is rather replaced by the martingale property.

The final stochastic ingredient used in this thesis is the so-called *Doob martingale* and the *method of bounded differences*. Let $X_1, X_2, \ldots, X_n$ be a sequence of random variables. The Doob sequence $Y_0, \ldots, Y_n$ of a function $f$ defined upon $X_1, \ldots, X_n$ is defined as [DP09]

$$Y_i = \mathrm{E}[f(X_1, \ldots, X_n)|X_i, \ldots, X_1] \qquad \text{for } 0 \leq i \leq n \ .$$

It follows that $Y_0 = \mathrm{E}[f(X_1, \ldots, X_n)]$ and $Y_n = f(X_1, \ldots, X_n)$. The Doob sequence of the function $f$ defines a martingale such that [DP09]

$$\mathrm{E}[Y_i|X_{i-1} \ldots X_1] = Y_{i-1} \qquad \text{for } 0 \leq i \leq n \ .$$

The Doob martingale provides the link to the following statement, which is known as the method of bounded differences. Let $X_1, \ldots, X_n$ be an arbitrary sequence of random variables and let $f$ be a function such that for each $i$ with $1 \leq i \leq n$ there is a $c_i \geq 0$ such that

$$|\mathrm{E}[f(X_1, \ldots, X_n)|X_i, \ldots, X_1] - \mathrm{E}[f(X_1, \ldots, X_n)|X_{i-1}, \ldots, X_1]| \leq c_i \ .$$

Then the deviation from the expectation is bounded as [DP09]

$$\Pr[|f(X_1, \ldots, X_n) - \mathrm{E}[f(X_1, \ldots, X_n)]| \geq t] \leq 2 \exp\left(\frac{-t^2}{2\sum_{i=1}^{n} c_i^2}\right) \ .$$

For further versions of this bound, formal proofs, and applications including the classical balls-into-bins experiment, see, e.g., the textbooks by Motwani and Raghavan [MR95], Mitzenmacher and Upfal [MU05], and Dubhashi and Panconesi [DP09].

# 4 Our Results

## 4.1 Information Dissemination

Information dissemination is the first fundamental problem from the area of parallel and distributed computing that we discuss in this thesis. Given a network, the goal is to spread one or several messages efficiently among all nodes of the network. This problem has been extensively analyzed in different communication models and on various communication networks. When talking about information dissemination, we will distinguish between *one-to-all* communication called broadcasting and *all-to-all* communication called gossiping. Much of the work devoted to information dissemination refers to the broadcasting problem. That is, a distinguished node of the network has a piece of information, which then has to be distributed to all nodes in the system. In gossiping, every node has its own piece of information, and all these messages must be distributed to all other nodes in the network.

For our results, we consider the so-called *random phone call model* (and a generalization thereof), which has been introduced by Demers et al. [DGH+87] and analyzed in detail by Karp et al. [KSSV00]. The main ingredient in the random phone call model is the following extension of the model described in Section 2.1. In each step of the synchronous algorithms defined for that model, every node opens a communication channel to a randomly chosen neighbor. The channel can then be used for bi-directional communication to exchange messages.

In the case of broadcasting, it is known that the performance of push-pull algorithms in complete graphs cannot be achieved in random graphs of small or moderate degree [Els06]. However, this does not carry over to the gossiping problem. In Part I, we show that, concerning the number of message transmissions, the performance of the algorithms developed by Berenbrink et al. [BCEG10] in the random phone call model can be achieved in random graphs as well. Note that regarding the impact of the graph density on the run time a similar study has been conducted by Fountoulakis et al. [FHP10]. They showed that there is almost no difference between the run time of the push algorithm in complete graphs and random graphs of various degrees, as long as the expected degree is $\omega(\log n)$.

In contrast to the result by Chen and Pandurangan [CP12] for a slightly weaker model, where they essentially showed a lower bound of $\Omega(n \log n)$ on the message complexity regardless of the run time for any gossiping algorithm, our result shows that in random graphs one can obtain the same improvement on the number of message transmissions w.r.t. the algorithms studied so far as in complete graphs. More precisely, we adapt the algorithm by Berenbrink et al. [BCEG10] to random graphs of suitable expected node degree and achieve a run time of $\mathrm{O}\left(\log^2 n / \log\log n\right)$ using only $\mathrm{O}(n \log n / \log\log n)$ messages.

Additionally, we will also describe a modification of this model that allows to design an $\mathrm{O}(\log n)$-time algorithm which requires only $\mathrm{O}(n \log\log n)$ message transmissions. Finally, we will present a series of plots showing our results from an empirical evaluation of our algorithms.

## 4.2 Load Balancing

In the second part of this thesis we consider the problem of *load balancing*. In this problem, we assume that we are given a parallel machine which consists of a large number of processing nodes that cooperate to solve a common task. However, in many real-world applications such as finite element simulations it may happen that the amount of work load generated on the compute nodes significantly deviates from the optimum. Now since the total run time of the parallel computation clearly depends on the slowest processor, one can obtain a substantial benefit by balancing the load among all processors. The goal is therefore to redistribute the load such that at the end each node has (almost) the same load [FWM94]. In Part II, however, we only consider *diffusion based load balancing* schemes, where nodes are only allowed to communicate with their direct neighbors to exchange load items. As before, we assume that the load balancing procedure runs in synchronous rounds, during which the exchange of load takes place.

We distinguish between *continuous* and *discrete* settings. In the continuous case it is assumed that the load can be split into arbitrarily small pieces. This assumption is very helpful when analyzing these algorithms [DFM99], albeit not realistic for many applications such as finite element simulations or scheduling problems. For discrete load balancing algorithms, however, we assume that tasks consist of atomic units of load. Therefore, only integral amounts of load can be transferred [EMS06, ABS16].

In Part II we state the theoretical main results from [ABEK15]. We describe the framework for randomly rounding continuous diffusion schemes to discrete schemes, we state a bound on the deviation between the so-called randomized *second order schemes* and their continuous counterparts, and we state a bound for the minimum initial load in a network that is sufficient to prevent the occurrence of so-called *negative load*.

As our main contribution, we support these results by extensive simulations on various graph classes, comparing the performance of FOS and SOS and giving an empiric insight into the behavior of diffusion based load balancing processes.

## 4.3 Distributed Voting and Plurality Consensus

Distributed voting and plurality consensus are the final two fundamental problems in distributed computing which we analyze in this thesis in Part III and Part IV. In both problems, we are given a communication network of players. Each of these players initially has one opinion from a set of possible opinions. The main goal is that players communicate such that eventually all players agree on one opinion. Similar to the previous results, we again assume that players only communicate with their direct neighbors in the network. Typically, one would demand from such a distributed voting procedure to run accurately, that is, the opinion with the highest number of initial supporters should win, and efficiently, that is, the voting process should converge within as few communication steps as possible. Additionally, voting and consensus algorithms are usually required to be simple, fault-tolerant, and easy to implement [Joh89].

### Distributed Voting According to the Majority Rule

In Part III, we study the following process which we denote the *deterministic binary majority voting process*. In this process, we are given a graph $G = (V, E)$ where each node has one of two possible opinions, for instance, *black* and *white*. The process runs synchronously in discrete rounds. In each round, every node computes the majority opinion among its neighbors, which is then adopted. Note that computing the majority is a deterministic operation. It is known that this process always *converges* to a two-periodic state. In this two-periodic state, nodes either keep their opinion or change their opinion in every round. Although nodes may have alternating opinions in every round, we still denote the number of rounds which is required to reach this two-periodic state as *convergence time*. We will furthermore denote the maximum of the convergence time over all possible initial opinion assignments as the *voting time*. The best known bounds on the voting time are linear in the number of edges. However, on many graph classes the process converges much faster. For example, on the clique the process converges in at most one round. In Chapter 14, we therefore perform a careful analysis of a potential function argument that had been used to prove the $\mathrm{O}(|E|)$ bounds and show that it is possible to efficiently compute much better bounds for these cases.

There would not be much interest in computing better bounds, if one could efficiently compute the maximum convergence time over all possible initial opinion assignments. However, in Chapter 13 we show that this is unlikely to be the case, since we can show that computing the *voting time* is NP hard by reducing 3SAT to the corresponding *voting time decision problem*.

Finally, in Chapter 15, we round off our main results by various additional interesting computational properties of the majority process. For example, we disprove a monotonicity of the convergence time w.r.t. the potential function and show that the voting time is not, at least straightforwardly, bounded by the diameter of the graph.

## Plurality Consensus and the Power of Two Choices

Finally, we consider the following plurality consensus process. We are again given a network of players modeled as a graph $G = (V, E)$. As before, each player in the network starts with one initial opinion from a set of possible opinions. In the original problem, the voting process runs in synchronous rounds, during which the players are allowed to communicate with their direct neighbors in the network with the main goal to eventually agree on one of the initial opinions. If all nodes agree on one opinion, we say this opinion *wins* and the process *converges*.

One straightforward variant is the so-called *pull voting* running in discrete rounds during which each player contacts a node chosen uniformly at random from the set of its neighbors and adopts the opinion of that neighbor. In [CEOR13], Cooper et al. show that the convergence time until a single message emerges for pull voting on any connected graph $G = (V, E)$ is asymptotically almost always in $\mathrm{O}(n/(\nu(1-\lambda_2)))$. In this bound, $\lambda_2$ is the second largest eigenvalue of the transition matrix of a random walk on the graph $G$. The parameter $\nu$ measures the *regularity* of $G$ with $1 \leq \nu \leq n^2/(2m)$, where the equality $\nu = 1$ holds for regular graphs. However, many other fundamental problems in distributed computing such as information dissemination [KSSV00] or aggregate computation [KDG03] can be solved much more efficiently. Therefore, Cooper et al. [CER14] considered a modified version. In their process, every node is allowed to contact two neighbors. Only if both neighbors have the same opinion, this opinion is adopted. In their model, for random $d$-regular graphs, all nodes agree after $\mathrm{O}(\log n)$ steps on the largest initial opinion with high probability, provided that $c_1 - c_2 = \Omega\left(n\sqrt{1/d + d/n}\right)$.

Our first main contribution in Chapter 17 is an extension of the results by Cooper et al. [CER14] on the complete graph to more than two colors. That is, in our model we assume that every node of the clique $K_n$ initially has one of $k$ possible opinions where $k = \mathrm{O}(n^\epsilon)$ for some small positive constant $\epsilon$.

Then, in Chapter 18 we investigate a modified model which we call the *memory model*. In this model, we allow each node to store and transmit one additional bit. Note that also in the classical two-choices protocol each node implicitly is assumed to have local memory, which is used, e.g., to store its current opinion. The main difference between the classical model and the memory model is that in the memory model each node also transmits one additional bit along with its opinion when contacted by a neighbor. The use of this additional bit allows us to drastically reduce the run time of the plurality consensus process. Note that the first protocol by Berenbrink et al. [BFGK16] and the protocol by Ghaffari and Parter [GP16] are similar to our work but were developed independently.

Finally, in Chapter 19 we adapt the memory-based approach to the asynchronous setting. In the asynchronous model, we assume that each node is equipped with a random clock which ticks according to a Poisson distribution in expectation once per time unit. (Analogously, one can say that the time between two ticks has an exponential distribution with parameter $\lambda = 1$.) However, since the Poisson process has the memoryless property, we model the asynchronous process by a sequence of discrete time steps. At each time step, one node is selected uniformly at random to perform its tick. As our main contribution, we show that if the difference between the sizes of the largest two opinions is at least $\Omega(c_2)$, where $c_2$ is the size of the second largest opinion, and $k = n^{\mathrm{O}\left(1/(\log\log n)^2\right)}$, then our algorithm achieves the best possible run time of $\mathrm{O}(\log n)$ assuming a node is allowed to communicate with at most constant many other nodes per step.

# Part I

# Information Dissemination

Robert Elsässer and Dominik Kaaser: *On the Influence of Graph Density on Randomized Gossiping*. In *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2015, pages 521–531. DOI: 10.1109/IPDPS.2015.32.

# 5 Introduction

In this part, we analyze a simple randomized gossiping protocol. While many authors often refer to *gossip protocols* when writing about any type of (randomized) information dissemination protocol, we will in the following make the distinction between broadcasting, *one-to-all* communication, and gossiping, *all-to-all* communication. Efficient gossip protocols for information dissemination are applied, e.g., in routing, maintaining consistency in replicated databases, multicasting, and leader election, see [BT89, FL94, HKP+05].

There are two main approaches to design efficient algorithms for broadcasting or gossiping. One way is to exploit structural properties of the networks on which the protocols are deployed on with the aim to design efficient *deterministic* schemes [HKP+05]. While the resulting protocols are usually (almost) optimal, they are often not fault tolerant (note that there are also deterministic schemes which have polylogarithmic run time on the graphs we consider and are highly robust, see the work by Haeupler [Hae13]). Another approach is to design simple randomized algorithms, which are inherently fault tolerant and scalable. Prominent examples of such algorithms are based on the so-called *random phone call model*, which has been introduced by Demers et al. [DGH+87] and was later analyzed in detail by Karp et al. [KSSV00]. The algorithms in this model are synchronous, that is, the nodes act in synchronous steps. In each step every node opens a communication channel to a randomly chosen neighbor. This channel can then be used for bi-directional communication to exchange messages between the corresponding nodes. It is assumed that the nodes may decide which messages they send (they are also allowed to send none of their messages in some step). Furthermore, nodes are able to combine several messages to one single packet, which then can be sent through a channel. Clearly, one question is how to count the message complexity if several pieces of information are contained in such a packet; we will come back to this question later.

Karp et al. motivated their work with consistency issues in replicated databases, in which frequent updates occur. These updates must be disseminated to all other nodes in the network to keep the database consistent. They analyzed the run time and number of message transmissions produced by so-called push and pull algorithms w.r.t. one single message in complete graphs. In order to determine the communication overhead, they counted the number of transmissions of this message through the links in the network. They argued that since updates occur frequently nodes have to open communication channels in each step anyway. Thus, the cost of opening communication channels amortizes over the total number of message transmissions.

Motivated by the application above, Berenbrink et al. considered the gossiping problem [BCEG10]. They assume that sending a packet through an open channel is counted once, no matter how many messages are contained in this packet. However, nodes may decide not to open a channel in a step, while opening a communication channel is also counted for the communication complexity. The first assumption is certainly unrealistic in scenarios, in which all original messages of the nodes have to be disseminated to all other nodes; although network coding might overcome the inpracticability of this assumption in certain applications, see for instance the work by Haeupler [Hae12]. On the other side, in the case of leader election, aggregate computation such as computing the minimum or the average, or for consensus the above assumption might be feasible, since the size of the exchanged messages can asymptotically be bounded by the size of a single message.

The algorithms developed so far in the random phone call model use so-called *push* and *pull* transmissions. As described above, the nodes open communication channels to randomly selected neighbors. If a message is sent from the node which called the neighbor and initiated the communication, then we talk about a push transmission w.r.t. that message. If the message is transmitted from the called node to the node that opened the channel, then we talk about a pull transmission.

As already stated in Chapter 4, an important question is, whether the results known for complete graphs also hold in sparse networks with very good expansion and connectivity properties. Such networks naturally arise in certain real world applications such as peer-to-peer systems [BW01, Gnu]. While it is known for broadcasting that the performance of push-pull algorithms in complete graphs cannot be achieved in random graphs of small or moderate degree [Els06], this seems not to be the case w.r.t. gossiping. As we show in the following chapters, the performance of the algorithms developed in [BCEG10] can be achieved in random graphs as well.

## 5.1 Related Work

A huge amount of work has been invested to analyze information dissemination in general graphs as well as some special network classes. In this part, we only concentrate on randomized protocols that are based on the random phone call model. This model has been introduced by Demers et al. [DGH+87] along with a randomized algorithm that solves the problem of mutual consistency in replicated databases.

Many papers analyze the run time of randomized broadcasting algorithms that only use push transmissions. To mention some of them, Pittel [Pit87] proved that in complete graphs a rumor can be distributed in $\log_2(n) + \ln(n) + O(1)$ steps. Feige et al. [FPRU90] presented optimal upper bounds for the run time of this algorithm in various graph classes including random graphs, bounded degree graphs, and the hypercube.

In their paper, Karp et al. [KSSV00] presented an approach that requires only $O(\log n)$ time and $O(n \log\log n)$ message transmissions, with high probability, which is also shown to be asymptotically optimal. This major improvement is a consequence of their observation that an algorithm that uses only pull steps is inferior to the push approach as long as less than half of the nodes are informed. After that, the pull approach becomes significantly better. This fact is used to devise an algorithm that uses both, push and pull operations. Additionally, a termination mechanism is introduced.

The random phone call model, as well as some variants of it, have also been analyzed in other graph classes. We mention here the work of Chierichetti et al. [CLP10] and Giakkoupis [Gia11] who related the run time of push-pull protocols to the conductance of a graph; or the work of Giakkoupis and Sauerwald [GS12, Gia14] on the relationship between push-pull and vertex expansion. To overcome bottlenecks in graphs with small conductance, Censor-Hillel and Shachnai used the concept of weak conductance to improve the run time of gossiping [CS12]. Earlier results related randomized information dissemination to random walks on graphs, see, e.g., [MS06, ES09]. Modifications of the random phone call model resulted in an improved performance of randomized broadcasting w.r.t. the communication complexity in random graphs [ES08] and w.r.t. the run time in the preferential attachment model [DFF11]. We use the basic idea of these modifications in Chapter 7.

Randomized gossiping in complete graphs has been extensively studied by Berenbrink et al. [BCEG10]. In their paper, they provided a lower bound argument that proves $\Omega(n \log n)$ message complexity for any $O(\log n)$ time algorithm. This separation result marks a cut between broadcasting and gossiping in the random phone call model. Furthermore, the authors gave two algorithms at the two opposite points of the time and message complexity trade-off. Finally, they slightly modified the random phone call model to circumvent these limitations and designed a randomized gossiping protocol which requires $O(\log n)$ time and $O(n \log\log n)$ message transmissions.

Chen and Pandurangan [CP12] used gossiping algorithms for computing aggregate functions in complete graphs, see also [KDG03]. They showed a lower bound of $\Omega(n \log n)$ on the message complexity regardless of the run time for any gossiping algorithm. However, for this lower bound they assumed a model that is slightly weaker than the one used in this part. In the main part of their paper, they presented an algorithm that performs gossiping in $\mathrm{O}(\log n)$ time using $\mathrm{O}(n \log\log n)$ messages by building certain communication trees. Furthermore, they also designed gossip protocols for general graphs. For all these algorithms, they assumed a communication model which is more powerful than the random phone call model.

Another interesting application of randomized gossiping is in the context of resilient information exchange. Alistarh et al. [AGGZ10] proposed an algorithm with optimal $\mathrm{O}(n)$ communication overhead, which can tolerate oblivious faults. For adaptive faults they provided a gossiping algorithm with a communication complexity of $\mathrm{O}\left(n \log^3 n\right)$. Their model, however, is stronger than the random phone call model or some simple variants of it.

Random graphs first appeared in probabilistic proofs by Erdős and Rényi [ER59]. Much later, they were described in the works by Bender and Canfield [BC78], Bollobás [Bol80] and Wormald [Wor81b, Wor81a]. Aiello et al. generalized the classical random graph model, introducing a method to generate and model power law graphs [ACL01]. The properties of Erdős-Rényi graphs have been surveyed by Bollobás [Bol01]. Various properties of random graphs, including random regular graphs, were presented by Wormald [Wor99]. In recent years, random graphs were also analyzed in connection with the construction and maintenance of large real world networks, see, e.g., the work by Kermarrec et al. [KMG03].

## 5.2 Our Results

In this part, we extend the results by Berenbrink et al. [BCEG10] to random graphs with degree $\Omega\left(\log^K n\right)$ where $K \geq 5$ can be an arbitrary constant. In [BCEG10], the authors first proved a lower bound, which implies that any address-oblivious algorithm in the random phone call model with run time $\mathrm{O}(\log n)$ produces a communication overhead of at least $\Omega(n \log n)$ in complete graphs. On the other side, it is easy to design an $\mathrm{O}(\log n)$-time algorithm, which generates $\mathrm{O}(n \log n)$ message transmissions. The first question is whether increasing the run time can decrease the communication overhead. This has been answered positively for complete graphs. That is, in [BCEG10] an algorithm with run time $\mathrm{O}\left(\log^2 n / \log\log n\right)$ and message complexity $\mathrm{O}(n \log n / \log\log n)$ was presented. However, it is still not clear whether this result can be achieved in sparser graphs as well. One might intuitively think that results obtained for complete graphs should be extendable to sparse random graphs as well, as long as the number of time steps is less than the smallest degree. However, in the related model of randomized broadcasting there is a clear separation between results achievable in complete graphs and in random graphs of degree $n^{\mathrm{o}(1/\log\log n)}$, see also [KSSV00, Els06].

In this part, we show that in random graphs one can obtain the same improvement on the number of message transmissions w.r.t. the algorithms studied so far as in complete graphs. In light of the fact that in the slightly different communication model analyzed by Chen and Pandurangan in their lower bound theorem [CP12] such an improvement is not even possible in complete graphs, our result provides evidence for a non-trivial advantage of the well-established random phone call model, that is, the possibility to improve the communication overhead by increasing the run time. Furthermore, we will present a modification of this model – as in [BCEG10] – to derive an $\mathrm{O}(\log n)$-time algorithm, which produces only $\mathrm{O}(n \log\log n)$ message transmissions, with high probability, and analyze the robustness of this algorithm.

In this part, we will show our first result w.r.t. the configuration model in Chapter 6, while the second result is proved for Erdős-Rényi graphs in Chapter 7. Nevertheless, both results can be shown for both random graph models, and the proof techniques are essentially the same. Here we only present one proof w.r.t. each graph model. Note that by performing an elaborate case analysis our results can be extended to random graphs with degree $\Omega\left(\log^{2+\epsilon} n\right)$ for a small constant $\epsilon \geq 0$. If applicable, we will state and show our lemmas in Chapter 6 for the more general case $d = \Omega\left(\log^{2+\epsilon} n\right)$.

In our analysis, we divide the execution time of our algorithms into several phases as in the case of complete graphs. Although the algorithms and the overall analysis are in the same spirit as in [BCEG10], we encountered several differences concerning the details. At many places, results obtained almost directly in the case of complete graphs required additional probabilistic and combinatorial techniques in random graphs. Moreover we observed that, although the overall results are the same for the two graph classes, there are significant differences in the performance of the corresponding algorithms in some of the phases mentioned before. This is due to the different structures we have to deal with in these two cases. To obtain our results, it was necessary to incorporate these structural differences into the dynamical behavior of the gossiping algorithms. For the details as well as a high level description of our algorithms see Chapter 6 and Chapter 7.

## 5.3 Model and Notation

We investigate the gossiping problem in the random phone call model in which $n$ players are able to exchange messages in a communication network. In our first model, we use a Erdős-Rényi random graph $G = G(n,p) = (V,E)$ to model the network where $V$ denotes the set of players and $E \subseteq V \times V$ is the set of edges. In this model, we have a probability of $p$ that for two arbitrary nodes $v_1, v_2 \in V$ the edge $(v_1, v_2)$ exists, independently. Let $d$ denote the expected degree of an arbitrary but fixed node $v$. In this part, we only consider undirected random graphs for which $d \geq \log^K n$ for a suitable constant $K \geq 5$. In this model the node degree of every node is concentrated around the expectation, that is, $d_v = \deg(v) = d \cdot (1 \pm \mathrm{o}(1))$, with high probability.

We also investigate the so-called configuration model introduced by Bollobás [Bol80]. We adapt the definition by Wormald [Wor99] as follows. Consider a set of $d \cdot n$ edge *stubs* partitioned into $n$ cells $v_1, v_2, \ldots, v_n$ of $d$ stubs each. A perfect matching of the stubs is called a *pairing*. Each pairing corresponds to a graph in which the vertices are the cells and the pairs define the edges. A pairing can be selected uniformly at random in different ways. In particular, the first stub in the pair can be chosen using any arbitrary rule as long as the second stub is chosen uniformly at random from the remaining unpaired stubs. Note that this process can lead to multiple edges and loops. However, with high probability the number of such edges is a constant [Wor99]. In our analysis we apply the principle of deferred decisions [MR95]. That is, we assume that at the beginning all nodes have $d$ stubs which are not yet connected. If a node chooses a link for communication for the first time in a step, then we connect the corresponding stub of the node with a free stub in the graph, while leaving all other stubs as they are.

We furthermore assume that each node has an estimation of $n$, which is accurate within constant factors. In each step, every node $v$ is allowed to open a channel to one of its neighbors denoted by $u$ chosen uniformly at random (in Chapter 7 we consider a simple modification of this model). This channel is called outgoing for $v$ and incoming for $u$. We assume that all open channels are closed at the end of every step. Since every node opens at most one channel per step, at most one outgoing channel exists per node.

Each node has access to a global clock, and all actions are performed in parallel in synchronous steps. At the beginning, each node $v$ stores its original message $m_v(0) = m_v$. Whenever $v$ receives messages, either over outgoing channels or over incoming channels, these messages are combined together. That is, $v$ computes its message in step $t$ by successively combining all known messages together, resulting in $m_v(t) = \bigcup_{i=0}^{t-1} m_v^{(\mathrm{in})}(i)$, where $m_v^{(\mathrm{in})}(i)$ denotes the union of all incoming, that is, received, messages over all connections in a step $i$ (with $m_v^{(\mathrm{in})}(0) = m_v$). This combined message is used for any transmission in step $t$. We will omit the step parameter $t$ and use $m_v$ to denote the node's message if the current step is clear from the context.

# 6 Traditional Model

In this chapter we present our algorithm to solve the gossiping problem. This algorithm is an adapted version of `fast-gossiping` presented by Berenbrink et al. [BCEG10]. It works in multiple phases, starting with a distribution process, followed by a random walk phase and finally a broadcasting phase. These phases are described below. Each phase consists of several rounds which may again consist of steps. The algorithm uses the following per-node operations, defined in Table 6.1.

| Operation | Description |
|---|---|
| `open()` | open a connection to a randomly chosen neighbor |
| `push(`$m$`)` | send $m$ over the outgoing channel |
| `pull(`$m$`)` | **send** $m$ over **incoming** channel(s) (see [KSSV00]) |
| `pushpull()` | a combination of `push` and `pull` |
| `receive()` | receive the messages from all open channels |
| `close()` | close all open channels |

**Table 6.1:** per-node communication operations for gossiping algorithms

In Phase II of Algorithm 6.1 we require each node to store messages associated with incoming random walks in a queue $q_v$ which we assume to support an `add` operation for adding a message at the end and a `pop` operation to remove the first message. The current queue status can be obtained via the `empty` operation which yields a Boolean value indicating whether the queue is empty or not. Additionally, we assume that each incoming message $m$ in the second phase phase has a counter `moves`$(m)$ attached that indicates how many real *moves* it has already made. This counter can be accessed using the `moves` operation and is described in more detail in the random walks section. Above operations are summarized in Table 6.2.

```
Algorithm FastGossiping($G$)
    Phase I
        for step $t = 1$ to $12 \log n / \log\log n$ do
            at each node $v$ do in parallel
                open();
                push($m_v$);                    // $m_v(t)$ as defined in Section 5.3
                $m_v \leftarrow m_v \cup$ receive();
                close();
    Phase II
        let $\ell$ denote a large constant;
        for round $r = 1$ to $4 \log n / \log\log n$ do
            at each node $v$ do in parallel
                with probability $\ell / \log n$ do
                    open();
                    push($m_v$);                // start a random walk
                    close();
            for step $t = 1$ to $6\ell \log n$ do
                at each node $v$ do in parallel
                    for each incoming message $m'$ do
                        if moves($m'$) $\leq c_{moves} \cdot \log n$ then
                            $q_v$.add($m' \cup m_v$);
                            $m_v \leftarrow m_v \cup m'$;
                    if $\neg$empty($q_v$) then
                        open();
                        push($q_v$.pop());
                        close();
            for each node $v$ do
                if $\neg$empty($q_v$) then
                    $v$ becomes active;
            for step $t = 1$ to $1/2 \cdot \log\log n$ do
                at each node $v$ do in parallel
                    if $v$ is active then
                        open();
                        push($m_v$);
                        close();
                    if $v$ has incoming messages then
                        $v$ becomes active;
            All nodes become inactive;
    Phase III
        for $t = 1$ to $8 \log n / \log\log n$ do
            at each node $v$ do in parallel
                open();
                pushpull($m_v$);
                $m_v \leftarrow m_v \cup$ receive();
                close();
```

**Algorithm 6.1:** the `fast-gossiping` algorithm

| Operation | Description |
|---|---|
| $q_v$`.add(`$m$`)` | add $m$ at the end of $q_v$ |
| $q_v$`.pop()` | remove and return the first element of $q_v$ |
| $q_v$`.empty()` | return whether the queue is empty or not |
| `moves(`$m$`)` | return the number of moves of a random walk |

**Table 6.2:** queue operations for gossiping algorithms

We now state our first main theorem as follows.

**Theorem 1.** *The gossiping problem can be solved in the random phone call model on random regular graphs with node degree $\Omega\left(\log^K n\right)$ for a suitable constant $K \geq 5$ in $\mathrm{O}\left(\log^2 n/\log\log n\right)$ time using $\mathrm{O}(n\log n/\log\log n)$ transmissions, with high probability.*

## 6.1 Phase I – Distribution

The first phase consists of $12\log n/\log\log n$ steps. In every step, each node opens a channel, pushes its messages, and closes the communication channel. Clearly, this phase meets the bounds for runtime and message complexity.

Let $k \geq 6$ denote a constant. We prove our result with respect to the configuration model described in Section 5.3. After the first phase, we have at least $\log^k n$ informed nodes w.r.t. each message, with high probability. We analyze our algorithm throughout this chapter with respect to one single message $m$ and at the end use a union bound to show that the result holds with high probability for all initial messages.

**Definition 1.** *Let $I_m(t)$ be the set of vertices that are informed of message $m$ in a step $t$, that is, vertices in $I_m(t)$ have received $m$ in a step prior to $t$. Accordingly, $|I_m(t)|$ is the number of informed nodes in step $t$. Let $H_m(t)$ be the set of uninformed vertices, that is, $H_m(t) = V \setminus I_m(t)$.*

We now bound the probability that during a communication step an arbitrary but fixed node opens a connection to a previously informed vertex, that is, the communication is redundant and thus the message is wasted. Let $v$ denote this vertex with corresponding message $m_v$.

At the beginning, we consider each connection in the communication network as unknown, successively pairing new edges whenever a node opens a new connection (see principle of deferred decisions in Section 5.3). Note, however, that this is only a tool for the analysis of our algorithm and does not alter the underlying graph model. We observe that each node has $d_v$ communication stubs with $\log^{2+\epsilon} n \leq d_v < n$. We consider a stub *wasted* if it was already chosen for communication in a previous step. Since throughout the entire first phase each node opens at most $12 \log n / \log \log n$ channels, there still will be $\Theta(d_v)$ *free* stubs available with high probability. Observe that the number of stubs that are additionally paired due to incoming channels can be neglected using a simple balls-into-bins argument [RS98]. If a node chooses a free stub, it is paired with another free stub chosen uniformly at random from the graph $G$.

**Lemma 2.** *After the distribution phase, every message is contained in at least $\log^k n$ nodes, with high probability, where $k \geq 6$ is a constant.*

To show Lemma 2 which corresponds to Phase I of Algorithm 6.1 we first state and show Lemma 3, Lemma 4, Lemma 5, and Lemma 6.

**Lemma 3.** *The probability that an arbitrary but fixed node $v$ opens a connection to a previously uninformed vertex w.r.t. message $m$ is at least $1 - \mathrm{O}\left(\log^{-1} n\right)$.*

*Proof.* The first phase runs for $12 \log n / \log \log n$ steps with the goal to reach at least $\log^k n$ informed vertices. We apply the principle of deferred decision as described in Section 5.3 to bound the number of *uncovered* (wasted) stubs that have already been connected. The total number of uncovered stubs at a node can be bounded by $S = \mathrm{O}(\log n)$ with high probability applying a simple balls-into-bins argument [RS98]. Then,

$$\Pr[v \text{ chooses a } \textit{wasted} \text{ stub}] \leq \frac{\mathrm{O}(\log n)}{d_v} \quad .$$

If in step $t$ a *free* stub is chosen, the probability that the corresponding communication partner $u$ has already been informed (or will be informed in exactly the same step) can be bounded by

$$\Pr[u \text{ is informed}] \leq \frac{d \cdot |I_m(t)|}{(d-S)n} \quad .$$

Therefore, the probability $p'$ that $v$ opens a connection to an uninformed communication partner and thus spreads the message to an uninformed node is

$$p' \geq \Pr[v \text{ chooses a } \textit{free} \text{ stub to } u] \cdot \Pr[u \text{ is uninformed}]$$

which yields for sufficiently large $n$

$$p' \geq \left(1 - \frac{1}{\log n}\right) \cdot \left(1 - \frac{d \cdot \log^k n}{(d-S)n}\right) \geq 1 - \mathrm{O}\left(\frac{1}{\log n}\right) \quad . \qquad \square$$

**Lemma 4.** *Let $C$ denote a large constant. After the first $T = 4 \log n / \log \log n$ steps, at least $C$ of nodes are informed of message $m_v$ with high probability.*

*Proof.* During these first $4 \log n / \log \log n$ steps we aim to reach at least $C$ informed nodes with high probability. Therefore, we have a probability of at most $C/d_v$ that an informed node $v$ opens a connection to another informed node and thus causes redundant communication. Furthermore, the probability that in an arbitrary but fixed step $t$ every communication attempt fails and every node $v \in I_m(t)$ performs only redundant communication can also be upper bounded by $C/d$.

We define an indicator random variable $X_i$ as

$$X_i = \begin{cases} 1 & \text{if } |I_m(i+1)| \geq |I_m(i)| + 1 \\ 0 & \text{otherwise} \end{cases}$$

which we sum up to obtain the number of informed nodes $X = \sum_{i=1}^{T} X_i$. We then bound the probability that more than $C$ steps fail, that is, the number of successful transmissions $X$ is smaller than $C$, as

$$\begin{aligned} \Pr[X \leq C] &\leq \sum_{i=0}^{C} \binom{T}{i} \cdot \left(1 - \frac{1}{d}\right)^{i} \cdot \left(\frac{C}{d}\right)^{T-i} \\ &< \sum_{i=0}^{C} \left(\frac{4 \log n \cdot e}{\log\log n \cdot i}\right)^{i} \cdot \left(\frac{C}{\log^{2+\epsilon} n}\right)^{\frac{4\log n}{\log\log n} - i} \\ &\ll \frac{1}{n^2} \end{aligned}$$

where in the second inequality we used that $\binom{T}{i} \leq \left(\frac{T \cdot e}{i}\right)^{i}$. $\square$

**Lemma 5.** *Let $t \in [4 \log n / \log\log n, 12 \log n / \log\log n]$ denote an arbitrary but fixed step. Then $|I_m(t+1)| \geq 1.5 \cdot |I_m(t)|$ with probability at least $1 - \log^{-1-\Omega(1)} n$.*

*Proof.* According to Lemma 4 we have $C \leq |I_m(t)| \leq \log^k n$ where $C$ denotes a large constant. In each step, every node opens a connection to a randomly chosen communication partner. Let $c$ denote a constant. According to Lemma 3, this attempt to inform a new node fails with a probability smaller than $c/\log n$. We now define the indicator random variable $X_i$ for $v_i \in I_m(t)$ as follows.

$$X_i = \begin{cases} 1 & \text{if } v_i \text{ opens a connection to } u \in I_m(t) \\ 0 & \text{otherwise.} \end{cases}$$

The aggregate random variable $X = \sum_{i=1}^{|I_m(t)|} X_i$ with expected value $\mathrm{E}[X] \leq c \cdot |I_m(t)| / \log n$ represents the total number of failed communication attempts. Clearly, we get $|I_m(t+1)| = 2|I_m(t)| - X$. Therefore, we upper bound $X$, using Equation 12 from [HR90] as follows:

$$\begin{aligned} \Pr\left[X \geq \frac{1}{2}|I_m(t)|\right] &\leq \left(\frac{2c}{\log n} \cdot 2\left(1 - \frac{c}{\log n}\right)\right)^{|I_m(t)|/2} \\ &\leq \left(\frac{4c}{\log n}\right)^{|I_m(t)|/2} \end{aligned}$$

We can now apply the lower bound for the number of informed nodes, $|I_m(t)| \geq C$, and obtain for large $n$

$$\Pr[|I_m(t+1)| \geq 1.5 \cdot |I_m(t)|] \geq 1 - \log^{-C/2+1} n \ . \qquad \square$$

**Lemma 6.** *At least $4 \log n / \log \log n$ attempts out of the $8 \log n / \log \log n$ last steps in Phase I succeed such that $|I_m(t+1)| \geq 1.5 \cdot |I_m(t)|$, with high probability. That is, half of the steps lead to an exponential growth.*

*Proof.* As of Lemma 5, the growth in each step can be lower bounded by $|I_m(t+1)|/|I_m(t)| \geq 1.5$ with probability at least $1 - \log^{-1-\Omega(1)} n$. We now define the indicator random variable $X_i$ as

$$X_i = \begin{cases} 1 & \text{if } |I_m(i+1)| < 1.5 \cdot |I_m(i)| \\ 0 & \text{otherwise.} \end{cases}$$

We sum up these indicator random variables and obtain the random variable $X = \sum_{i=1}^{8 \log n / \log \log n} X_i$ which represents the number of steps that fail to inform a sufficiently large set of new nodes. Again, we use Equation 12 from [HR90] to bound $X$ as follows.

$$\begin{aligned} \Pr\left[X \geq \frac{4 \log n}{\log \log n}\right] &\leq \left(\frac{4}{\log^{1+\Omega(1)} n}\left(1 - \frac{1}{\log^{1+\Omega(1)} n}\right)\right)^{\frac{4 \log n}{\log \log n}} \\ &\ll n^{-3} \end{aligned} \qquad \square$$

We now combine these results to give a proof for Lemma 2 which concludes the first phase.

*Proof of Lemma 2.* Since each message $m$ starts in its original node, we initially have $|I_m(0)| = 1$. We conclude from Lemma 6 that with high probability in at least $4 \log n / \log \log n$ steps the number of nodes informed of $m$ increases by a factor of at least 1.5 as long as $|I_m(t)| \leq \log^k n$. Thus, we have with high probability

$$\left|I_m\left(\frac{12 \log n}{\log \log n}\right)\right| \geq \min\left\{\log^k n, 1.5^{\frac{4 \log n}{\log \log n}}\right\} = \log^k n \ .$$

We apply a union bound over all messages and the lemma follows. $\square$

## 6.2 Phase II – Random Walks

After the first phase, each message is contained with high probability in at least $\log^k n$ nodes, where $k \geq 6$ is a constant. We aim to reach $n \cdot 2^{-\log n / \log \log n}$ informed nodes for each message in the second phase and therefore assume for any message $m$ and any step $t$ in Phase II that $\log^k n \leq |I_m(t)| \leq n \cdot 2^{-\log n / \log \log n}$.

At the beginning of Phase II a number of nodes start so-called random walks. If a random walk arrives at a node in a certain step then this node adds its messages to the messages contained in the random walk and performs a push operation, that is, the random walk moves to a neighbor chosen uniformly at random. This is done for $\mathrm{O}(\log n)$ steps. To ensure that no random walk is lost, each node collects all incoming messages (which correspond to random walks) and stores them in a queue to send them out one by one in the following steps. The aim is to first collect and then distribute messages corresponding to these walks. After the random walk steps all nodes containing a random walk become *active*. A broadcasting procedure of $1/2 \cdot \log\log n$ steps is used to increase the number of informed nodes by a factor of $\Theta(\sqrt{\log n})$. The entire second phase runs in $4\log n/\log\log n$ rounds which correspond to the outer `for`-loop in Phase II of Algorithm 6.1. Each round consists of $\mathrm{O}(\log n)$ steps. Thus, the run time of this phase is in $\mathrm{O}\left(\log^2 n/\log\log n\right)$.

Note that although random walks carry some messages, we assume in our analysis that the nodes visited by the random walks do not receive these messages from the random walks. That is, the nodes are not necessarily informed *after* they were visited by a random walk and thus are not accounted to $I_m$.

In the following, we consider an arbitrary but fixed round $r$ that belongs to the second phase with $1 \leq r \leq 4\log n/\log\log n$. Whenever we use the expression $I_m(r)$, we mean the set of informed nodes at the beginning of the corresponding round $r$, even though the informed set may be larger in some step of this round.

At the beginning of each round, every node flips a coin. With a probability of $\ell/\log n$, where $\ell$ denotes a large constant, the node starts a random walk. We first need to bound the total number of random walks which are initiated. As their number does not depend on the underlying graph, we can use the result of [BCEG10] for the number of random walks and obtain $\Theta(n/\log n)$ random walks with high probability. Therefore, the bounds on the message complexity of $\mathrm{O}(n\log n/\log\log n)$ are met during the random walks phase. In the following we only consider random walks that carry an arbitrary but fixed message $m$.

We observe that these random walks are not independent from each other, since a random walk $w$ incoming at node $v$ is enqueued into a queue $q_v$. Therefore, $w$ may be delayed before it is sent out again by $v$ and this delay is based on the number of (other) random walks that are currently incident at node $v$. If $v$ eventually sends out the random walk $w$, we say $w$ makes a *move*. It is now an important observation that the actions of the random walks in a specific step are *not* independent from each other. Their *moves*, however, are.

Now a question that arises naturally is whether the number of moves made by an arbitrary but fixed random walk $w$ is large enough to *mix*. This question is covered in Lemma 7, where we will argue that the number of moves taken by every random walk is $\Omega(\log n)$ and therefore larger than the mixing time of the network. In the following lemmas, especially in Lemma 8, we will also require that the random walks are not correlated, which clearly is not true if we consider the steps made by the algorithm. However, the moves of the random walks are independent from each other. That is, after mixing time moves, the node that hosts random walk $w$ after its $i$-th move is independent from the nodes that host any other of the random walks after their $i$-th moves. We furthermore require, e.g., in Lemma 13, that after some mixing steps the random walks are distributed (almost) uniformly at random over the entire graph. This is enforced as we stop every random walk once it has reached $c_{\text{moves}} \cdot \log n$ moves for some constant $c_{\text{moves}}$. Note that we implicitly attach a counter to each random walk which is transmitted alongside the actual message. In the first inner `for`-loop in Phase II of Algorithm 6.1 we then refuse to enqueue random walks that have already made enough moves.

Note that starting with Lemma 8, when we talk about random walks in a certain step $i$ we always mean each random walk after its $i$-th move. This does not necessarily have to be one single step of the algorithm, and the corresponding random walks are scattered over multiple steps. Since, however, the moves of the random walks are independent from each other, the actual step can be reinterpreted in favor of the random walk's movements. What remains to be shown is that every random walk makes indeed $\Omega(\log n)$ moves. This is argued in the following lemma.

**Lemma 7.** *The random walks started in Phase II of Algorithm 6.1 make $\Omega(\log n)$ moves, with high probability.*

*Proof.* At the beginning we fix one single random walk $r$, and let $P$ be the sequence of the first $\log n/4$ nodes visited by this random walk, whenever $r$ is allowed to make a move. Note that some nodes in $P$ may be the same (e.g., the random walk moves to a neighbor $v$ of some node $u$, and when the random walk is allowed to make a move again, then it jumps back to $u$). Clearly, the number of random walks moving around in the network is $O(n/\log n)$, with high probability. For simplicity, let us assume that there are exactly $n/\log n$ random walks (a few words on the general case are given at the end of this proof). We now consider the number of vertices in the neighborhood $N(v)$ of a vertex $v$ which host a random walk at some time step $i$. We show by induction that for each time step $1 \leq i \leq \log n/4$ and any node $v$ with probability $1 - 2i/n^3$ it holds that

1. The number of vertices hosting at least one random walk is at most $\frac{d}{\log n}\left(1 + \frac{2i}{\log n}\right)$.
   This set is denoted by $N_1(v)$.
2. The number of vertices hosting at least two random walks is at most $\frac{d}{\log^2 n}\left(1 + \frac{2i}{\log n}\right)$.
   This set is denoted by $N_2(v)$.
3. The number of vertices hosting three or more random walks is at most

$\frac{di}{\log^3 n}$.
This set is denoted by $N_3(v)$.

For the proof we condition on the event that there are at most 4 circles involving $v$, the nodes of the first neighborhood of $v$, and the nodes of the second neighborhood of $v$. Note that this event holds with very high probability for a large range of $d$, that is, $d \leq n^\alpha$ for some $\alpha$ constant but small, see, e.g., [DFS09], [BES14], or for random regular graphs a similar proof done by Cooper, Frieze, and Radzik [CFR09]. These edges can be treated separately at the end and are neglected for now. Moreover, in the configuration model it is possible to have multiple edges or loops. However, for this range of degrees there can be at most constantly many, which are treated as the circle edges mentioned above at the end. We use $c_{\text{circle-edges}}$ to denote the constant for the number of multiple edges and circle edges. For the case $d > n^\alpha$, different techniques have to be applied, however, a similar proof as in the complete graph case can be conducted. For now, we assume that $d \geq \log^5 n$. For random regular graphs of degree $d \in [\log^{2+\epsilon} n, \log^5 n]$ the proof ideas are essentially the same, however, at several places an elaborate case analysis becomes necessary.

Now to the induction. In the first time step, the hypothesis obviously holds, that is, each node starts a random walk with probability $1/\log n$, independently. Assume now that the induction hypothesis holds for some time step $i$, and we are going to show that it also holds for step $i+1$. Note that the assumption holds in the neighborhood of each node, and thus, also in the neighborhoods of the nodes of $N(v)$. We start by showing claim 3. In each step, every node of $N_1(v)$ will release a random walk. There are $d$ vertices in $N(v)$, and $di/\log^3 n$ nodes with at most 3 random walks. A node of $N(v) \setminus N_2(v)$ becomes an $N_3(v)$ node with probability at most

$$\binom{\frac{d}{\log n}\left(1+\frac{2i}{\log n}\right)}{3} \cdot \frac{1}{d^3} \quad . \tag{6.1}$$

(Note that above equation is an approximation of the more exact calculation involving the sum $\sum_{i=2}^{|N_1(w)|} \binom{|N_1(w)|}{3} \left(\frac{1}{d}\right)^3 \left(1-\frac{1}{d}\right)^{(|N_1(w)|-i)}$ where $w$ is a neighbor of $v$. This sum can be approximated efficiently by using bounds on the tail of the binomial distribution. We work here with the simpler expression in (6.1).) Therefore the expected value of these nodes is at most

$$\mathrm{E}[Z] = \binom{\frac{d}{\log n}\left(1+\frac{2i}{\log n}\right)}{3} \cdot \frac{1}{d^3} \cdot d \quad .$$

Since we only consider the nodes which are not involved in any cycles and do not have multiple edges each node $w'$ in the second neighborhood of $v$ sends a random walk to the corresponding neighbor in $N(v)$ independently of the other nodes in the second neighborhood. Thus we can apply Chernoff bounds and obtain that the number of the nodes in $N(v)$ which receive a random walk is $\mathrm{E}[Z](1+\mathrm{o}(1))$.

An $N_2(v)$ node becomes an $N_3(v)$ node with probability

$$\binom{\frac{d}{\log n}\left(1+\frac{2i}{\log n}\right)}{2} \cdot \frac{1}{d^2} \ .$$

Again, since the neighborhoods of the different nodes are disjoint (up to at most 4 edges, which can be treated separately and therefore are neglected in the future), we may apply Chernoff bounds, and obtain an upper bound for $N_3(v)$ as follows.

$$\begin{aligned}
&\binom{\frac{d}{\log n}\left(1+\frac{2i}{\log n}\right)}{3} \cdot \frac{1}{d^3} \cdot d \cdot (1+\mathrm{o}(1)) + \\
&\binom{\frac{d}{\log n}\left(1+\frac{2i}{\log n}\right)}{2} \cdot \frac{1}{d^2} \cdot \frac{d}{\log^2 n} \cdot \left(1+\frac{2i}{\log n}\right) \cdot (1+\mathrm{o}(1)) + \\
&|N_3(v)| + c_{\text{circle-edges}}
\end{aligned}$$

Recall that we initially neglected circle edges and multiple edges. In the worst case, the nodes incident at these edges send a random walk to $N_3(v)$ (as well as to $N_2(v)$ and $N_1(v)$) in every step and therefore the last expression $c_{\text{circle-edges}}$ denotes a constant for these additional incoming messages. Noting that furthermore $i < \log n/4$ we obtain that $N_3(v) \leq d(i+1)/\log^3 n$ in the next step, with high probability.

Concerning the $N_2(v)$ nodes, a node being in $N(v) \setminus N_2(v)$ becomes an $N_2(v)$ node with probability

$$\binom{\frac{d}{\log n}\left(1+\frac{2i}{\log n}\right)}{2} \cdot \frac{1}{d^2} \ .$$

Similarly, a node in $N_2(v)$ will still remain in $N_2(v)$ with a probability of $\frac{1}{\log n}\left(1+\frac{2i}{\log n}\right)$. Applying again Chernoff bounds for both cases separately, we obtain the result. Additionally, we add the $N_3(v)$ nodes to the $N_2(v)$ nodes, and a similar calculation as above shows that the given bound is not exceeded.

Now we concentrate on nodes in $N_1(v)$. A node being in $N(v) \setminus N_2(v)$ becomes (or remains) an $N_1(v)$ node with probability

$$\frac{1}{\log n}\left(1+\frac{2i}{\log n}\right) \ .$$

Note that there can be at most $d$ nodes in this set. Applying Chernoff bounds as above and adding the $N_2(v)$ nodes to this set, we obtain the upper bound.

Now, we know that every neighborhood $N(v)$ has at most $\frac{d}{\log n}\left(1+\frac{2i}{\log n}\right)$ nodes which possess at least one random walk in step $i$, with high probability. This implies that in each time step, the number of random walks sent to $v$ is a random variable which has a binomial distribution with mean

$$\frac{1}{\log n}\left(1+\frac{2i}{\log n}\right) \leq \frac{3}{2\log n} \ .$$

That is, if we denote by $X_v$ this random variable then $X_v$ can be modeled by the sum of $3d/2 \log n$ Bernoulli random variables with success probability $1/d$. Thus, within $\log n/4$ steps, $v$ collects in total at most $X$ random walks, where

$$\Pr\left[X > \frac{3}{4} \cdot c \cdot \log n / \log\log n\right] \le \left( \frac{e^{\frac{c \log n}{\log\log n} - 1}}{\left(\frac{c \log n}{\log\log n}\right)^{\frac{c \log n}{\log\log n}}} \right)^{\frac{3}{4}} \le \frac{1}{n^5} \ ,$$

if the constant $c$ is large enough. This also implies that at any node, there will be no more than $c \log n/\log\log n$ many random walks for some proper $c$, and hence, if a random walk arrives, it is enough to consider the last $c \log n/\log\log n$ steps. That is, when a random walk arrives to a node, the number of random walks can be represented by the sum of $c \log n/\log\log n$ independent random variables $X_v$ (as described above) with binomial distribution having mean $\mathrm{O}(1/\log n)$ each. The probability above is an upper bound, and this bound is independent of the distribution of the random walks among the vertices (conditioned on the event that the induction hypotheses 1, 2, and 3 hold, which is true with very high probability).

Consider now some time steps $t_1, t_2, \ldots, t_i, \ldots$ which denote movements of the random walk $r$ from one vertex to another one. Whenever $r$ makes a move at some time $t_i$, it has to wait for at most

$$\sum_{j=t_i - \frac{c \log n}{\log\log n}+1}^{t_i} X_j \tag{6.2}$$

steps, where $X_j$ is a random variable having binomial distribution with mean $O(1/\log n)$. One additional step after this waiting time $r$ will make a move. If a random walk leaves a node twice, at time $t_i$ and $t_j$ respectively (with $t_i < t_j$), then we consider the sum in (6.2) from $\max\{t_j - c \log n/\log\log n + 1, t_i + 1\}$. Observe that $t_i$ is a random variable that depends on $t_{i-1}$ and the random variable for above waiting time. In order to have

$$\sum_{i=1}^{t} \left( \sum_{j=t_i - \frac{c \log n}{\log\log n}+1}^{t_i} X_j(t_i) + 1 \right) = \frac{\log n}{4} \tag{6.3}$$

with some probability at least $1/n^2$, $t$ must be $\Omega(\log n)$ where $X_j(t)$ is a random variable with the same properties as $X_j$ described above. This implies that within $\log n/4$ steps, $r$ makes $\Omega(\log n)$ moves, with high probability. This holds, since

$$\Pr\left[\sum_{i=1}^{t} \sum_{j=t_i - \frac{c \log n}{\log\log n}+1}^{t_i} \sum_{k=1}^{\frac{3d}{2}} X_{ijk} \ge \frac{\log n}{4}\right] = \frac{1}{n^{\omega(1)}}$$

for $t = \mathrm{O}(\log n)$. The sum $\sum_{k=1}^{\frac{3d}{2}} X_{ijk}$ represents the random variable $X_j(t_i)$ (see above, where $X_{ijk}$ is a Bernoulli random variable with success probability $1/d$) and the second sum represents the inner sum from (6.3). Observe that above sum represents an upper bound on the sum of the random walks that random walk $r$ meets when moving from one node to another according to the sequence $P$ defined at the beginning. That is, the sum gives the time $r$ has to wait at the nodes without making a move.

Note that in the proof we showed that if at the beginning there are $n/\log n$ randomly chosen nodes starting a random walk, then each random walk makes at least $\Omega(\log n)$ moves with high probability. If we start $\ell \cdot n/\log n$ random walks, then the proof can be adapted accordingly so that $\Omega(\log n)$ moves are also performed by each random walk, with high probability. (The calculations become a bit more complex, however.) Noting that the eigenvalues of the transition matrix of these graphs are inverse polynomial in $d$, the random walks are well mixed. □

**Lemma 8.** *During the $\Theta(\log n)$ steps that follow the coin flip, $I_m(r)$ is visited by random walks at least $\Omega(|I_m(r)|)$ times, with high probability.*

*Proof.* Let $m$ denote an arbitrary but fixed message and $I_m(r)$ the corresponding set of vertices that are informed of $m$ at the beginning of a round $r$. Depending on the coin flip each node starts a random walk with probability $\ell/\log n$ and therefore we have a total number of random walks in $\Theta(n/\log n)$. Let $X$ denote the random variable for the number of random walks that currently reside in $I_m(r)$ in an arbitrary but fixed step of round $r$. In expectation we have $\mathrm{E}[X] = |I_m(r)| \cdot \ell/\log n$ such random walks. We use Chernoff bounds on $X$ and obtain that

$$\Pr\left[|X - \mathrm{E}[X]| > \frac{\mathrm{E}[X]}{\log n}\right] \leq n^{-\Omega\left(\frac{|I_m(r)|}{\log^3 n}\right)} .$$

Therefore, we conclude that this number of random walks is concentrated around the expected value with high probability and thus is in $\Theta(|I_m(r)|/\log n)$. Since these random walk moves are not correlated and choose their next hop uniformly at random we conclude that in any such step the number of random walks that reside in $I_m(r)$ is in $\Theta(|I_m(r)|/\log n)$ with high probability. Using union bounds over all $\Theta(\log n)$ steps following the coin flip we conclude that there are $\Theta(|I_m(r)|)$ random walk visits in the set of informed vertices $I_m(r)$ in these $\Theta(\log n)$ steps, with high probability. □

Note that a rigorous analysis of the behavior of similar parallel random walks on regular graphs has been already considered by Becchetti et al. [BCN+15b]. Furthermore, see also the work by Becchetti et al. [BCN+15a] for a similar analysis on the complete graph.

These $\Theta(|I_m(r)|)$ random walks do not necessarily need to be distinct. It may happen that a single random walk visits the set $I_m(r)$ multiple times, in the worst case up to $\Theta(\log n)$ times. We therefore have to give bounds the number of random walks that visit $I_m(r)$ only a constant number of times.

We now distinguish two cases. Let $\kappa$ denote a large constant. In the following, we consider only *sparse* random graphs with expected node degree $d$ for which $\log^{2+\epsilon} n \leq d \leq \log^{\kappa} n$. We observe that if $d \leq \log^{\kappa} n$ the informed set consists of several connected components, which we call regions, that have a diameter of at most $\mathrm{O}(\log\log n)$ each and a distance between each other of at least $\Omega(\log\log n)$ (see Lemma 13).

Let $v$ denote an arbitrary but fixed vertex and let $T(v)$ denote the subgraph induced by nodes that can be reached from $v$ using paths of length at most $\mathrm{O}(\log\log n)$. It has been shown in Lemma 4.7 from [BES14] that $T(v)$ is a *pseudo-tree* with high probability, that is, a tree with at most a constant number of additional edges. Therefore, we can assign an orientation to all edges of $T(v)$ in a natural way, pointing from the root node $v$ towards the leafs. Thus, any edge in $T(v)$ is directed from $v_1$ to $v_2$ if $v_1$ is in at most the same level as $v_2$. We consider edges violating the tree property with both nodes on the same level as oriented in both ways. Whenever a random walk takes a step that is directed towards the root of the tree, we speak of a *backward move*.

**Lemma 9.** *Assume $d \leq \log^{\kappa} n$. An arbitrary but fixed random walk leaves the set of informed vertices $I_m(r)$ to a distance in $\Omega(\log\log n)$ and does not return to $I_m(r)$ with a probability of at least $1 - \log^{-2} n$.*

To show Lemma 9 we first introduce and show Lemma 10 and Lemma 11.

**Lemma 10.** *Assume $d \leq \log^{\kappa} n$. Any random walk originating in a node of $T(v)$ takes in the first $2\log\log n$ steps only a constant number of backward moves with probability at least $1 - \log^{-3} n$.*

*Proof.* We consider an arbitrary but fixed random walk $w$ that is informed with $m_v$, that is, it carries $m_v$, and focus on the first $\log\log n$ steps after $w$ was informed for the first time. Let $X_i$ denote the random variable for the orientation of the edge taken by $w$ in the $i$-th step, defined as

$$X_i = \begin{cases} 1 & \text{if } w \text{ takes a back edge in step } i \\ 0 & \text{otherwise.} \end{cases}$$

From the pseudo-tree-property of $T(v)$ we can conclude that the probability of $w$ using a back edge is at most $\mathrm{O}(1/d)$, since every node has one edge to its parent and additionally at most a constant number of edges that are directed backwards.

Let $c \geq 3$ denote a constant. We define the random variable $X = \sum_{i=1}^{\log\log n} X_i$ for the number of back edges taken by $w$ in $2\log\log n$ steps with expected value $\mathrm{E}[X] \leq \mathrm{O}(\log\log n/d)$. Since we can assume that $X$ has a binomial distribution we can directly derive the probability that more than a constant number of $c$ steps taken by $w$ are backward steps using $\binom{n}{k} \leq \left(\frac{n \cdot e}{k}\right)^k$ as follows.

$$\begin{aligned}
&\Pr[X \geq c] \\
&= \sum_{i=c}^{2\log\log n} \binom{2\log\log n}{i} \cdot \left(\frac{\mathrm{O}(1)}{d}\right)^i \cdot (1-\mathrm{o}(1))^{2\log\log n - i} \\
&< \sum_{i=c}^{2\log\log n} \left(\frac{2\log\log n \cdot e}{i}\right)^i \cdot \left(\frac{\mathrm{O}(1)}{d}\right)^i \\
&< \sum_{i=c}^{2\log\log n} \left(\frac{\mathrm{O}(\log\log n)}{i \cdot \sqrt{d}}\right)^i \cdot \left(\frac{1}{\sqrt{d}}\right)^i \\
&\leq \mathrm{O}(\log\log n) \cdot \left(\frac{\mathrm{O}(\log\log n)}{\sqrt{d}}\right)^c \cdot \left(\frac{1}{\sqrt{d}}\right)^c \\
&< \log^{-c} n \leq \log^{-3} n \qquad \square
\end{aligned}$$

In the following we consider random walks that are more than $\log\log n$ steps away from the set of informed nodes.

**Lemma 11.** *Assume $d \leq \log^{\kappa} n$. The probability that a random walk does not return to an informed region $T(v)$ in $\mathrm{O}(\log n)$ steps once it has a distance to the region that is greater than $\log\log n$ steps is at least $1 - \log^{-3} n$.*

*Proof.* Let $w$ denote an arbitrary but fixed random walk and let $a$ denote a constant. We use a Markov chain to model and analyze the behavior of the random walk $w$ with respect to its distance to $I_m$. Let $X$ denote a random variable for the number of steps $w$ takes backward. Because of the pseudo-tree property the probability that the random walk moves backward can be bounded by $p' = \mathrm{O}(1/d)$ for any node with distance $\mathrm{O}(\log\log n)$ to the root. Thus, the probability that $w$ takes $\tau$ backward steps in a total of $a \log n$ tries can be bounded by

$$\begin{aligned}
\Pr[X = \tau] &\leq \binom{a\log n}{\tau} \left(\frac{4}{\log^2 n}\right)^{\tau} \left(1 - \frac{4}{\log^2 n}\right)^{a\log n - \tau} \\
&< \binom{a\log n}{\tau} \left(\frac{4}{\log^2 n}\right)^{\tau}
\end{aligned}$$

which gives using $\binom{n}{k} \leq \left(\frac{n \cdot e}{k}\right)^k$

$$\Pr[X = \tau] < \left(\frac{ea\log n}{\tau}\right)^{\tau} \left(\frac{4}{\log^2 n}\right)^{\tau} = \left(\frac{4ea}{\tau \log n}\right)^{\tau} .$$

We now consider only those random walks that have a distance larger than $\log\log n$ to the root node of the local informed tree. Note that there remains a *safety belt* around the informed set, since the broadcasting procedure performed by each random walk at the end of the round (see last `For`-loop in Phase II of Algorithm 6.1) builds up a tree with height at most $1/2 \cdot \log\log n$. We investigate $\tau = 1/2 \cdot \log\log n$, the distance to cross this *safety belt*, and observe

$$\Pr[X = \tau] \leq \left(\frac{1}{\log n}\right)^{\log\log n/2}$$

and therefore

$$\begin{aligned}\Pr[X \geq \tau] &< \sum_{\tau=\log\log n/2}^{a\log n} \left(\frac{4ea}{\tau \log n}\right)^{\tau} \\ &< (a\log n - \log\log n/2)\left(\frac{1}{\log n}\right)^{\log\log n/2} \\ &\leq \log^{-3} n \enspace . \end{aligned}$$ □

We are now ready to prove Lemma 9.

*Proof of Lemma 9.* From Lemma 10 and Lemma 11 we conclude that with probability at least

$$1 - \left(1 - \frac{1}{\log^3 n}\right)\left(1 - \frac{1}{\log^3 n}\right) \geq 1 - \log^{-2} n$$

an arbitrary but fixed random walk $w$ leaves the set of informed vertices to some distance in $\mathrm{O}(\log\log n)$ and does not return. Together, this yields Lemma 9. □

We will show in Lemma 13 that the distance between the informed regions is at least $\Omega(\log\log n)$. Thus we can show Lemma 12 using the following definition.

**Definition 2** (Safe Area)**.** *A* safe area *is a set of nodes that are uninformed and have distance at least $\log\log n$ to any informed node.*

**Lemma 12.** *Assume $d \leq \log^{\kappa} n$. The number of random walks that visit $I_m(r)$ at most a constant number of times is $\Theta(|I_m(r)|)$ with high probability.*

*Proof.* Let $c$ denote a constant. We examine steps $s \in [\log n, 2\log n]$ after the coin flip. In Lemma 8 we showed that the number of random walks visits in the informed set $I_m(r)$ is in $\Theta(|I_m(r)|)$ with high probability. Let $W$ denote this number. Let furthermore $Q$ be the set of random walks that visit $I_m(r)$ at most a constant number of $c$ times and let $P$ be the set containing all the other random walks. The inequality

$$W \leq c \cdot |Q| + \log n \cdot |P|$$

holds since the random walks in $Q$ hit $I_m(r)$ at most $c$ times, and the random walks in $P$ at most $\log n$ times, respectively. The probability that a random walk does not leave the set of informed vertices to a distance of $\log\log n$ can be bounded by $\log^{-3} n$ according to Lemma 10. Furthermore, we need to show that with probability $\log^{-2} n$ the random walk hits any other informed region at most a constant number of times. This follows from the idea of a *safety belt* as described in the proof of Lemma 11, where we observed that the probability that a random walk returns through this region of distance $1/2 \cdot \log\log n$ to any informed node can be bounded by $\log^{-3} n$. A simple union bound over all $\Theta(\log n)$ steps gives a probability of $\log^{-2} n$ that a random walk hits an informed node.

It is crucial that in above analysis we regard only informed nodes that arose from random walks broadcasting in a *safe area* according to Definition 2, thus giving us above setup of informed balls, safety belts and long distances between informed regions. We show these properties in Lemma 13. Therefore, we can bound the probability that an individual random walk visits $I_m(r)$ more often than a constant number of $c$ times by $\log^{-2} n$

We now bound the number of random walks in $P$, that is, the number of random walks that return more often than a certain constant number of $c$ times. Let the indicator random variable $X_i$ be defined for a random walk $w_i$ as

$$X_i = \begin{cases} 1 & \text{if } w_i \text{ returns more often than } c \text{ times} \\ 0 & \text{otherwise.} \end{cases}$$

The random variable $X = |P| = \sum_{i=1}^{W} X_i$ describes the number of random walks that return more often than $c$ times. The expected value of $X$ can be bounded by $\mathrm{E}[X] \leq W \cdot \log^{-2} n$. Since all random walks are independent we apply Chernoff bounds and obtain for sufficiently large $n$

$$\Pr\left[X \geq \left(1 + \frac{1}{\log n}\right) \mathrm{E}[X]\right] \leq e^{-\frac{\Theta(|I_m(r)|)}{3\log^4 n}} \leq n^{-\omega(1)} .$$

Therefore, with high probability

$$\begin{aligned} W &\leq c \cdot |Q| + \log n \cdot |P| \\ &\leq c \cdot |Q| + \log n \cdot \left(1 + \frac{1}{\log n}\right) \cdot \frac{W}{\log^2 n} \end{aligned}$$

and thus

$$c \cdot |Q| \geq W \cdot \left(1 - \left(\frac{1}{\log n} + \frac{1}{\log^2 n}\right)\right)$$

which gives $|Q| = \Theta(W)$. Since $W = \Theta(|I_m(r)|)$ we finally obtain that $|Q| = \Theta(|I_m(r)|)$. □

**Lemma 13.** *Assume $d \leq \log^{\kappa} n$. The number of random walks that terminate in a* safe area *is in $\Theta(|I_m(r)|)$.*

*Proof.* Let $W$ denote the number of random walks. Each random walk performs at the end $\mathrm{O}(\log n)$ mixing steps. Thus, the random walks are distributed (almost) uniformly at random over the entire graph. For the analysis, we now proceed as follows. We uncover one random walk after another, thereby omitting random walks that stopped too close to another previously uncovered random walk. For each of these steps, the probability $p_{\text{unsafe}}$ that a random walk ends up in an unsafe area can be bounded as follows.

$$p_{\text{unsafe}} \le \frac{|I_m(r)| d^{\log\log n}}{n} \le \frac{\log^{\kappa \log\log n} n}{2^{\log n/\log\log n}}$$

We define for every random walk $w_i$ an indicator random variable $X_i$ as

$$X_i = \begin{cases} 1 & \text{if } w_i \text{ ends in an unsafe area} \\ 0 & \text{otherwise} \end{cases}$$

and bound the random variable $X = \sum_{i=1}^{W} X_i$ representing the number of random walks that end within an unsafe area. The expected number of these walks is $\mathrm{E}[X] = p_{\text{unsafe}} W$. Since all random walks are independent, applying Chernoff bounds yields for large $n$

$$\Pr\left[X \ge \left(1 + \frac{1}{\log n}\right) \mathrm{E}[X]\right] \le e^{-\frac{\mathrm{E}[X]}{3\log^2 n}} \le n^{-\omega(1)} \quad .$$

Therefore, there are $\Theta(|I_m(r)|)$ random walks in safe areas with high probability. □

**Lemma 14.** *Assume $d > \log^{\kappa} n$. A random walk visits the set $I_m(r)$ at most a constant number of times with probability at least $1 - \log^{-2} n$. Furthermore, the number of random walks that visit the set $I_m(r)$ a constant number of times is $\Theta(|I_m(r)|)$ with high probability.*

*Proof.* Since our algorithm runs for at most $\mathrm{O}\left(\log^2 n/\log\log n\right)$ time, each node has during the second phase at least $\Omega(d)$ free stubs available. Therefore we bound the probability that in step $t$ an arbitrary but fixed random walk $w$ located at node $v$ opens an already used stub or connects to a node $u$ in the informed set $I_m(t)$ as follows.

$$\Pr[v \text{ opens a used stub}] \le \mathrm{O}\left(\log^{-\kappa+2}\right)$$
$$\Pr[u \in I_m(t)] \le |I_m(t)| \cdot d / \left(n\left(d - \log^2 n\right)\right)$$

Therefore, we obtain a probability $p'$ that an unused stub is chosen and the corresponding communication partner was not previously informed of $p' > 1 - \log^{-3} n$. We apply union bounds over all random walk steps and observe that with probability $p' > 1 - \log^{-2} n$ a random walk does not hit any other informed node.

To show the second part of Lemma 14 we analyze the random walks phase from the following point of view. We know that with probability at most $\log^{-2} n$ a random walk hits the informed set. Therefore, we consider the experiments of starting one random walk after another. Each of these trials fails with at most above probability. However, the trials are negatively correlated, since we omit those random walks that interfere with the informed set and thus also with another random walk. Note that we only regard those random walks as valid that do not interfere with the informed set at least once and only choose communication stubs that have not been previously selected.

Since the correlation is negative we can apply Chernoff bounds on the the number of random walks that fail. Let $X_i$ denote an indicator random variable for the $i$-th random walk, defined as

$$X_i = \begin{cases} 1 & \text{if the } i\text{-th random walk fails} \\ 0 & \text{otherwise.} \end{cases}$$

Let furthermore $X$ be the number of random walks that fail, defined as $X = \sum_{i=1}^{W} X_i$. From Lemma 8 we obtain that the total number of random walks visits in $I_m(t)$ is in $\Theta(|I_m(t)|)$. The expected value of $X$ can be bounded by $\mathrm{E}[X] \le W \cdot \log^{-2} n = \mathrm{o}(|I_m(t)|)$. We show that $X$ is concentrated around its expected value as follows.

$$\Pr\left[X \ge \left(1 + \frac{1}{\log n}\right)\mathrm{E}[X]\right] \le e^{-\frac{\mathrm{E}[X]}{3\log^2 n}} \le n^{-\omega(1)}$$

Therefore we have only $\mathrm{o}(|I_m(t)|)$ random walks that exhibit undesired behavior with high probability and thus the lemma holds. $\square$

**Lemma 15.** *The broadcasting procedure during the last $1/2 \log\log n$ steps of a round $r$ in Phase II informs $\Theta(|I_m(r)| \cdot \sqrt{\log n})$ nodes, with high probability.*

*Proof.* Let $w$ denote an arbitrary but fixed random walk and, again, let $\kappa$ denote a large constant. We distinguish the following two cases to show that the probability that a node $u_i$ opens a connection to an already informed node can be bounded for both, sparse and dense random graphs by $\log^{-2} n$.

**Case 1: $d \le \log^{\kappa} n$.** Each random walk operates in its own *safe area* as described in Lemma 13. That means, we only consider random walks that have a distance of at least $\log\log n$ between each other. Therefore, in a broadcast procedure of at most $1/2 \cdot \log\log n$ steps no interaction between the corresponding broadcast trees can occur. Let $u_i$ be the $i$-th node with respect to a level-order traversal of the message distribution tree of nodes informed by an arbitrary but fixed random walk. Let furthermore $X_i$ denote an indicator random variable for the connection opened by $u_i$ defined as

$$X_i = \begin{cases} 1 & \text{if } u_i \text{ opens a back connection} \\ 0 & \text{otherwise.} \end{cases}$$

The claim follows from the *pseudo-tree* structure of the local subgraph, since every node has at most a constant number of edges directed backwards and furthermore we only regard random walks in a safe area, that is, random walks with a distance of $\log\log n$ steps between each other. Therefore, the probability probability that the node $u_i$ opens a connection to an already informed node can be bounded by $\mathrm{O}(1/d) \leq \log^{-2} n$.

We denote the random variable for the number of nodes that open backward connections as $X$ and observe that $X \leq \sqrt{\log n}$ since the number of steps is $1/2 \log\log n$. Using above indicator random variable we set $X = \sum X_i$ with expected value $\mathrm{E}[X] \leq \log^{-3/2} n$. Let $c \geq 3$ denote a constant. Since we can assume that $X$ has a binomial distribution we can bound the probability that more than a constant number of $c$ nodes open backward connections directly by $\Pr[X \geq c] \leq \log^{-3} n$ as follows.

$$
\begin{aligned}
&\Pr[X > c] \\
&= \sum_{i=c+1}^{\sqrt{\log n}} \binom{\sqrt{\log n}}{i} \left(\frac{1}{\log^2 n}\right)^i \left(1 - \frac{1}{\log^2 n}\right)^{\sqrt{\log n}-i} \\
&\leq \sum_{i=c+1}^{\sqrt{\log n}} \left(\frac{\sqrt{\log n}\cdot e}{i}\right)^i \left(\frac{1}{\log^2 n}\right)^i \left(1 - \frac{1}{\log^2 n}\right)^{\sqrt{\log n}-i} \\
&\leq \sum_{i=c+1}^{\sqrt{\log n}} \left(\frac{e}{i\log^{3/2} n}\right)^i \left(1 - \frac{1}{\log^2 n}\right)^{\sqrt{\log n}-i} \\
&\leq \left(\sqrt{\log n} - c - 1\right) \left(\frac{e}{(c+1)\log^{3/2} n}\right)^{c+1} \\
&\leq \frac{1}{\log^c n} \leq \log^{-3} n \ .
\end{aligned}
$$

In the worst case these $c$ nodes are the $c$ topmost nodes of the message distribution tree and the corresponding branches of this tree are lost. However, for a constant $c$ the resulting informed set is still in $\Theta\big(\sqrt{\log n}\big)$.

**Case 2: $d > \log^{\kappa} n$.** We consider the number of connection stubs that are available at an arbitrary but fixed node $v$ and observe that the probability that $v$ opens an already used stub can be bounded by

$$\Pr[v \text{ opens a used stub to } u] \leq \mathrm{O}\Big(\log^{-\kappa+2} n\Big) \ ,$$

that is, the total number of connections opened over the number of available stubs. Furthermore, we bound the probability that the target stub belongs to a node in the informed set as

$$\Pr[u \text{ is informed}] \leq |I_m(t)| \cdot d / \Big(n\Big(d - \log^2 n\Big)\Big) \ .$$

Therefore, the probability that either a previously used stub is opened or that the target has already been informed can be bounded for sufficiently large $n$ by $\log^{-3} n$. We apply union bounds and conclude that each random walk end informs a set of size $\sqrt{\log n}$ after $1/2 \log\log n$ steps with probability $1 - \log^{-2} n$.

**Both cases.** We apply Chernoff bounds on the number of random walks that do not manage to build up a sufficiently large informed set using broadcasting. In the first case all random walks are clearly uncorrelated, since they live within their own safe area. For the second case, we analyze the random walks one after another as individual trials in our experiment. Whenever a random walk fails to spread its message, we completely remove the entire random walk for our analysis. We therefore have probabilities that are negatively correlated which allows us to apply Chernoff bounds.

Let $X'_i$ denote an indicator random variable for a random walk $w_i$ defined as

$$X'_i = \begin{cases} 1 & \text{if the random walk } w_i \text{ fails broadcasting} \\ 0 & \text{otherwise.} \end{cases}$$

Let furthermore $X' = \sum_{i=1}^{W} X'_i$ denote the random variable for the number of random walks that fail during the broadcasting steps with expected value $\mathrm{E}[X'] \leq W/\log^2 n$ where $W$ is the total number of random walks. We show that $X'$ is concentrated around the expected value using Chernoff bounds.

$$\Pr\left[X' \geq \left(1 + \frac{1}{\log n}\right)\mathrm{E}[X']\right] \leq e^{-\frac{\mathrm{E}[X']}{3\log^2 n}} \leq n^{-\omega(1)}$$

Since this result holds with high probability, we have a set of informed nodes of size $|I_m(r+1)| = \Theta(|I_m(r)| \cdot \sqrt{\log n})$ and thus the claim holds. □

From Lemma 15 we obtain that the set of informed vertices grows in each round by a factor of at least $\Theta(\sqrt{\log n})$ as long as the number of informed vertices is in $\mathrm{O}\left(n \cdot 2^{-\log n/\log\log n}\right)$, with high probability. Assume that the exact factor for the growth in each round is $a\sqrt{\log n}$ where $a$ denotes a constant. Then, the number of informed nodes that can be reached in Phase II is at most

$$\begin{aligned} &\left(a\sqrt{\log n}\right)^{4\log n/\log\log n} \\ &= \left(a^2\right)^{2\log n/\log\log n} \cdot \left(\left(\sqrt{\log n}\right)^{2\log n/\log\log n}\right)^2 \\ &= \left(a^2 \cdot \sqrt{\log n}\right)^{2\log n/\log\log n} \cdot \left(\sqrt{\log n}\right)^{2\log n/\log\log n} \\ &\gg n \cdot 2^{-\log n/\log\log n} \quad . \end{aligned}$$

We apply a union bound over all messages and conclude we reach the bound on the number of informed vertices for Phase II after at most $4\log n/\log\log n$ rounds with high probability.

## 6.3 Phase III – Broadcast

In the last phase we use a simple push-pull broadcasting procedure to inform the remaining uninformed nodes. Once $\Omega(n/2^{\log n/\log\log n})$ nodes are informed after the second phase, within $O(\log n/\log\log n)$ additional steps at least $n/2$ nodes become informed, with high probability. Furthermore, after additional $O(\log n/\log\log n)$ steps, all nodes are informed with high probability [Els06].

**Lemma 16.** *After applying push-pull for* $O(\log n/\log\log n)$ *steps, at most* $n/\log n$ *uninformed vertices remain for every message* $m$*, with high probability. This procedure has a runtime complexity in* $O(\log n/\log\log n)$ *and an overall message complexity in* $O(n\log n/\log\log n)$*.*

*Proof.* Lemma 4 from [Els06] states that once the number of nodes possessing some message $m$ is $\Omega(n/2^{\log n/\log\log n})$, then in additional $O(\log n/\log\log n)$ steps the number of nodes informed of $m$ exceeds $n/2$. We observe that the number of informed nodes is within the bounds required in Lemma 4 from [Els06] for each message. We conclude that the set of informed vertices underlies an exponential growth with high probability. Therefore, $|I_m(t)| \geq n/2$ after additional $O(\log n/\log\log n)$ steps, using $O(n\log n/\log\log n)$ messages. Furthermore, we apply Lemma 5 from [Els06], which states that after additional $O(\log\log n)$ steps it holds that for the uninformed set $|H(t)| \leq n/\log n$ with high probability. Since both, Lemma 4 and Lemma 5 from [Els06] hold with high probability $1 - o(n^{-2})$ we use union bound over all messages and conclude that these results hold for all messages with high probability.

Note that the two lemmas from [Els06] are stated w.r.t. Erdős-Rényi random graphs. The same proofs, however, lead to the same statements for the configuration model. □

**Lemma 17.** *After* $O(\log n/\log\log n)$ *steps, every remaining uninformed node is informed of message* $m$ *with high probability.*

The proof of Lemma 17 is similar to Lemma 5 and Lemma 6 from [Els06]. Our adapted version is as follows.

*Proof.* After performing the mixing steps during the random walk phase, we can assume that each message is distributed uniformly at random nodes. From Lemma 16 we deduce that each node opens at most a number of connections in $O(\log n/\log\log n)$ after the last mixing phase, whereas each node has at least $\log^{2+\epsilon} n$ communication stubs. Additionally, we consider in the following phase $O(\log n/\log\log n)$ `pull` steps. Each node can open up to $O(\log n/\log\log n)$ additional connections during this phase and incoming connections from uninformed nodes can be bounded by the same expression following a balls-into-bins argument. We denote the number of opened stubs as $S$ with $S = O(\log n)$ and conclude that we still have at least $\log^{2+\epsilon} n - S = \Omega\left(\log^{2+\epsilon} n\right)$ *free* connection stubs available which are not correlated to the message distribution process of message $m$ in any way.

In each step, a node opens a connection to a randomly chosen neighbor and therefore chooses a *wasted* communication stub with probability at most $a \log n/(d \cdot \log\log n)$ where $a$ is a constant.

If a free stub is chosen, the corresponding communication partner is informed of message $m$ with probability at least $|I_m(t)| \cdot (d - S)/(n \cdot d)$. Therefore, any uninformed node $v$ *remains possibly uninformed*, that is, either uses an already wasted communication stub or connects to an uninformed partner, with the following probability.

$$\begin{aligned}
p' &= \Pr[v \text{ remains possibly uninformed}] \\
&= 1 - \Pr[v \text{ is definitely informed}] \\
&\leq 1 - \Pr[v \text{ chooses a free stub to } u] \cdot \Pr[u \in I_m(t)] \\
&\leq 1 - \left(1 - \frac{c}{\log n \cdot \log\log n}\right) \cdot \left(\left(1 - \frac{|H_m(t)|}{n}\right)\frac{d-S}{d}\right)
\end{aligned}$$

We apply $H_m(t) \leq n/\log n$ and obtain

$$\begin{aligned}
p' &\leq 1 - \left(1 - \frac{c}{\log n \cdot \log\log n}\right) \cdot \left((1 - 1/\log n)\frac{d-S}{d}\right) \\
&\leq \frac{c}{\log n \cdot \log\log n} + \left(1 - \frac{c}{\log n \cdot \log\log n}\right)\frac{d-S}{\log n \cdot d} \\
&< \log^{-c} n
\end{aligned}$$

for a suitable constant $c$. Therefore, the probability that an arbitrary node remains uninformed after $4 \log n/(c \cdot \log\log n)$ steps can be bounded by

$$\Pr[v \text{ remains uninformed}] \leq \left(\frac{1}{\log n}\right)^{\frac{4\log n}{\log\log n}} = \frac{1}{n^4} \; . \qquad \square$$

**Lemma 18.** *After the broadcast phase, every node is informed of every message with high probability.*

*Proof.* We use union bound on the results of Lemma 17 over all $n$ messages and over all $n$ nodes. Thus after the broadcast phase each node is informed of every message with probability at least $1 - n^{-2}$. □

*Proof of Theorem 1.* The theorem follows from the proofs of the correctness of the individual phases, Lemma 2 for the distribution phase, Lemma 13, Lemma 14, and Lemma 15 for the random walks phase, and Lemma 18 for the broadcast phase. □

7

# Memory Model

In this chapter we consider the $G(n,p)$ graph, in which an edge between two nodes exists with probability $p$, independently, and assume that the nodes have a constant size memory. That is, the nodes can store up to four different links they called on in the past, and they are also able to avoid these links as well as to reuse them in a certain time step. More formally, we assume that each node $v \in V$ has a list $l_v$ of length four. The entry $l_v[i]$ contains a link address which is connected on the other end to a fixed node $u$. Whenever node $v$ calls on $l_v[i]$ in a step, it opens a communication channel to $u$. From now on, we will not distinguish between the address stored in $l_v[i]$ and the node $u$ associated with this address. As assumed in the previous chapters, such a channel can be used for bi-directional communication in that step. Furthermore, $v$ is also able to avoid the addresses stored in $l_v$, by calling on a neighbor chosen uniformly at random from $N(v) \setminus \cup_{i=0}^{3}\{l_v[i]\}$, where $N(v)$ denotes the set of neighbors of $v$. This additional operation is denoted `open-avoid` in Algorithm 7.2 and Algorithm 7.1. Note that the approach of avoiding a few previously contacted neighbors was also considered in the analysis of the communication overhead produced by randomized broadcasting [BEF08, ES08] and in the analysis of the run time of push-pull protocols in the preferential attachment model [DFF11]. Clearly, the list $l_v$ may also be empty, or contain less than 4 addresses.

The algorithm we develop is similar to the one in [BCEG10] for complete graphs. However, in [BCEG10] the protocol just uses the fact that in the random phone call model the nodes of a complete graph do not contact the same neighbor twice with high probability. This cannot be assumed here. Furthermore, to obtain a communication infrastructure for gathering information at a so-called leader, we use some specific structural property of random graphs which was not necessary in complete graphs. There, we built an infrastructure by using communication paths in an efficient broadcasting network obtained by performing broadcasting once. Here, we need to analyze the structure of random graphs in relation with the behavior of our algorithm.

## 7.1 Leader Election

```
Algorithm LeaderElection($G$)
    at each node $v$ do in parallel
        with probability $\log^2 n/n$ do
            $v$ becomes active;
            open-avoid();
            push($ID_v$);
            close();
    for $t = 1$ to $\log n + \rho \log\log n$ do
        at each node $v$ do in parallel
            if $v$ has incoming messages $m$ then
                $v$ becomes active;
            Let $i_v(t)$ be the smallest identifier that $v$ received so far;
            if $v$ is active then
                open-avoid();
                push($i_v$);
                close();
    for $t = 1$ to $\rho \log\log n$ do
        at each node $v$ do in parallel
            open-avoid();
            $i_v \leftarrow \min\{i_v,$ receive()$\}$;
            close();
    at each node $v$ do in parallel
        if $ID_v = i_v$ then
            $v$ becomes the leader;
```

**Algorithm 7.1:** distributed leader-election algorithm

In our main algorithm, we assume that a single node is aware of its role as a leader. The other nodes, however, do not necessarily have to know the ID of this node. They just have to be aware of the fact that they are not leaders. In order to find a leader we may apply the following leader election algorithm described in Algorithm 7.1, see also [BCEG10].

Each node flips a coin, and with probability $\log^2 n/n$ it becomes a *possible* leader. We assume that every node $v$ has a unique ID denoted by $ID_v$. Each possible leader starts a broadcast, by sending its ID to some nodes chosen uniformly at random from the set of its neighbors, except the ones called in the previous three steps. Once a node receives some ID, it becomes active, and starts propagating the smallest ID it received so far. This push phase is performed for $\log n + \rho \log\log n$ steps, where $\rho > 64$ is some large constant. In the last $\rho \log\log n$ steps, the IDs of the possible leaders are spread by pull transmissions. The possible leader with the smallest ID will become the leader.

**Lemma 19.** *At the end of Algorithm 7.1, all nodes are aware of the leader, with high probability.*

*Proof.* Let us denote by $I(t)$ the set of nodes at time $t$, which have received some ID by this time step. Lemma 2.2 of [ES08] states that a message is distributed by a modified push-pull algorithm, in which each node is allowed to avoid the 3 neighbors chosen in the previous 3 steps, is distributed to $n - n/\sqrt[4]{n}$ nodes in $\log n + \rho \log\log n$ steps[1]. This implies that by this time $I(t) \geq n - n/\sqrt[4]{d}$, and the number of message transmissions is at most $\mathrm{O}(n \log\log n)$, with high probability. Furthermore, $n/\log^{\mathrm{O}(1)} n$ nodes know the leader. According to Lemma 2.7. and 2.8. from [ES08], after additional $\mathrm{O}(\log\log n)$ steps, the message is distributed to all nodes, with high probability. This implies that after this number of additional steps $I(t) = n$, with high probability, and $\Omega\left(n/\log^2 n\right)$ nodes know the leader, with high probability. Applying Lemmas 2.7. and 2.8. from [ES08] again, we obtain the lemma. □

Now we consider the robustness of the leader election algorithm. We show that by applying our algorithm, one can tolerate up to $n^{\epsilon'}$ random node failures, with high probability, where $\epsilon' < 1/4$ is a small constant. That is, during the execution of the algorithm, $n^{\epsilon'}$ nodes, chosen uniformly and independently at random, may fail at any time. The node failures are non-malicious, that is, a failed node does not communicate at all. The theorem below is stated for $p = \log^5 n/n$. However, with an extended analysis, the theorem can be generalized to any $p > \log^{2+\epsilon} n/n$.

**Lemma 20.** *In the failure model described above, at the end of Algorithm 7.1 the leader is aware of its role, and all other nodes know that they are not the leader with high probability.*

*Proof.* Here we only consider the node, which decided to become a possible leader, and has the smallest ID among such nodes. The algorithm is the same as the sequential version of the broadcast algorithm given in [ES08]. We know that within the first $(1-\epsilon')\log n - \rho \log\log n$ steps, the number of informed nodes, that is, the number of nodes receiving the ID of the node we consider, is $n^{1-\epsilon'}/\log^{2+\Omega(1)} n$, with high probability. Since $n^{\epsilon'}/n$ nodes may fail in total, independently, Chernoff bounds imply that all nodes informed within the first $(1-\epsilon')\log n - \rho \log\log n$ steps are healthy, with high probability. We also know that after $\log n + \rho \log\log n$ push steps, the number of informed nodes is $n - n/\log^{4+\Omega(1)} n$, with high probability [ES08]. On the other side, if we only consider push transmissions, the number of nodes which become informed by a message originating from a node informed after step $(1-\epsilon')\log n - \rho \log\log n$ is at most $2^{\epsilon' + 2\rho \log\log n} = n^{\epsilon'} \log^{\mathrm{O}(1)} n$. This is due to the fact that the number of informed nodes can at most double in each step. Thus, the total number of nodes, which received the message from a failed node

[1] We adapted the run time from the lemma mentioned before to our algorithm.

in the first $\log n + \rho \log\log n$ push steps, if this node would not fail, is at most $n^{2\epsilon'} \log^{O(1)} n$. The probability that one of the possible leaders is not among the nodes informed in the first $\log n + \rho \log\log n$ push steps, or is not informed due to a node failure, is $o\left(\log^{-4} n\right)$. The union bound over $O\left(\log^2 n\right)$ possible leaders implies the lemma. □

## 7.2 Gossiping Algorithm and its Analysis

The pseudocode can be found in Algorithm 7.2. We assume that at the beginning a random node acts as a leader. For an efficient and robust leader election algorithm see Algorithm 7.1. Once a leader is given, the goal is to gather all the messages at this node. First, we build an infrastructure as follows (Phase I). The leader emits a message by contacting four different nodes (one after the other), and sending them these messages. These nodes contact four different neighbors each, and send them the message. If we group four steps to one so-called long-step, then in long-step $i$, each of the nodes which received the message in the long-step before *for the first time* chooses four distinct neighbors, and sends them the message. Furthermore, each node stores the addresses of the chosen nodes. This is performed for $\log_4 n + \rho \log\log n$ long-steps, where $\rho > 64$ is some large constant. For the next $\rho \log\log n$ long-steps, all nodes, which have not received the message of the leader so far, choose 4 different neighbors in each of these long-steps, and open communication channels to these nodes, that is, communication channels are opened to all these different neighbors within one long-step, where each of these neighbors is called in exactly one step. If some node has the message of the leader in some step, then it sends this message through the incident communication channel(s) opened in that step. We call these last $\rho \log\log n$ long-steps *pull long-steps*.

In Phase II the infrastructure built in Phase I is used to send the message of each node to the leader. This is done by using the path, on which the the leader's message went to some node, to send the message of that node back to the leader. In the third phase the messages gathered by the leader are sent to all nodes the same way the leader's message was distributed in Phase I. Then, the following lemmas hold.

**Lemma 21.** *After $\log_4 n + \rho \log\log n$ long-steps at least $n/2$ nodes have the message of the leader, with high probability.*

*Proof.* Since during the whole process every node only chooses four neighbors, simple balls-into-bins arguments imply that the total number of incoming communication channels opened to some node $u$ is $O(\log n)$, with probability at least $1 - n^{-4}$ [RS98].

Let $v$ be the leader, and let its message be $m_v(0)$. We know that as long as $d = 2^{o\left(\sqrt{\log n}\right)}$, the tree spanned by the vertices at distance at most $\rho \log\log n$ from $v$ is a tree, or there are at most 4 edges which violate the tree property

```
Algorithm Memory($G$)
    Assume a leader is given.
    Phase I
        for $t = 0$ to $3$ do
            The leader performs an open-avoid and then then a push($m_v(0)$)
              operation. In each step, the leader stores in $l_v[t]$ the address of the
              node contacted in this step.
        for $t = 4$ to $4 \log_4 n + 4\rho \log\log n$ do
            Every node $v$ that received $m_v(0)$ in step $t$ *for the first time* (with
              $t = 4j + k$ and $k \in \{0, 1, 2, 3\}$) is active in step $4(j + 1), 4(j + 1) + 1$,
              $4(j + 1) + 2$, and $4(j + 1) + 3$.
            Every active node $v$ performs an open-avoid and then a
              push($m_v(0)$) operation. $v$ stores in $l_v[t \bmod 4]$ the address of the
              node contacted in the current step.
            Every active node $v$ also stores the time steps $4(j + 1)$, $4(j + 1) + 1$,
              $4(j + 1) + 2$ and $4(j + 1) + 3$ together with the neighbors it used for
              the push operations in the list $l_v$.
        for $t = 4 \log_4 n + 4\rho \log\log n + 1$ to $4 \log_4 n + 8\rho \log\log n$ do
            Every node $v$ that knows $m_v(0)$ performs pull($m_v(0)$) operation.
            Every node $v$ that does not know $m_v(0)$ performs an open-avoid and
              receives eventually ($m_v(0)$). The address of the contacted node is
              stored in $l_v[t \bmod 4]$.
            Every node $v$ that receives $m_v(0)$ for the first time in step $t$
              remembers the chosen neighbor together with $t$ in the list $l_v[0]$.
    Phase II
        $t' \leftarrow 4 \log_4 n + 8\rho \log\log n$
        for $t = 1$ to $\rho \log\log n$ do
            Every node $v$ which received the message in step $t' - t + 1$ (for the
              first time) opens a channel to the corresponding neighbor in $l_v[0]$
              and performs a push operation with all original messages it has.
        $t' \leftarrow 4 \log_4 n + 8\rho \log\log n$
        for $t = 1$ to $4 \log_4 n + 8\rho \log\log n$ do
            Every node $v$ which stores a neighbor with time step $t' - t + 1$ in its
              list $l_v$ opens a channel to that neighbor in $l_v$ and receives the
              message from that neighbor. The node at the other side performs a
              pull operation with all original messages it has.
    Phase III
        The leader broadcasts all original messages using the algorithm described
          in Phase I for message $m_v(0)$.
```

**Algorithm 7.2:** memory-based gossiping algorithm. After each step, the nodes close all channels opened in that step.

[BES14]. Thus, after $\rho \log\log n$ steps, at least $3^{\rho \log\log n - 1}$ vertices have $m_v(0)$ with high probability. If $d = 2^{\Omega(\sqrt{\log n})}$ simple probabilistic arguments imply that at least $3^{\rho \log\log n - 1}$ vertices have $m_v(0)$ with high probability.

Let now $I^+(t)$ be the set of nodes, which receive $m_v(0)$ in long-step $t$ (for the first time). Each of these nodes chooses an edge, which has already been used (as incoming edge), with probability $\mathrm{O}(\log n/d) \le 1/\log^{1+\epsilon/2} n$. Let $|I(t)| \le n/\log^2 n$ and $I^+(t) = \{v_1, \dots, v_{|I^+(t)|}\}$. Given that some $v_i$ has at least $pn(1 - \mathrm{o}(1))$ neighbors in $G$, and at most $|I(t)| + 4|I^+(t)|)p(1 + \mathrm{o}(1)) + 5 \log n$ neighbors in $I(t) \cup \{v_1, \dots, v_{i-1}\}$, the edge chosen by $v_i$ in a step of the long-step $t+1$ is connected to a node, which is in $I(t)$ or it has been contacted by some node $v_1, \dots, v_{i-1}$ in long-step $t+1$, with probability at most

$$p_I \le \frac{(|I(t)| + 4|I^+(t)|)p(1 + \mathrm{o}(1)) + 5 \log n}{pn(1 - \mathrm{o}(1))} \tag{7.1}$$

independently, see also [Els06]. Thus, we apply Chernoff bounds, and obtain that the number of newly informed nodes is

$$|I^+(t+1) \ge 4|I^+(t)| \left(1 - \frac{2}{\log^{1+\epsilon/2} n}\right) ,$$

with probability $1 - n^{-3}$. Therefore, after $\log_4 n - \mathrm{O}(\log\log n)$ steps, the number of informed nodes is larger than $n/\log^2 n$.

Now we show that within $\rho \log\log n$ steps, the number of uninformed nodes becomes less than $n/2$. As long as $|I(t)| \le n/3$, applying equation (7.1) together with standard Chernoff bounds as in the previous case, we obtain that

$$|I^+(t+1) \ge 4|I^+(t)| - |I^+(t)| \frac{5|I^+(t)(1 + \mathrm{o}(1))}{n} > 2|I^+(t)| ,$$

with probability $1 - n^{-3}$. Once $|I(t)|$ becomes lager than $n/3$, it still holds that $|I^+(t)| \ge |I(t)|$ (see above). Thus, in the next step the total number of informed nodes exceeds $n/2$, with high probability. □

The approach we use here is similar to the one used in the proof of Lemma 2.2. in [ES08]; the only difference is that in [ES08] the nodes transmitted the message in all steps, while here each node only transmits the message to 4 different neighbors chosen uniformly at random. Note that each node only opens a channel four times during these $\log_4 n + \rho \log\log n$ long-steps, which implies a message complexity of $\mathrm{O}(n)$.

**Lemma 22.** *After $\rho \log\log n$ pull long-steps, all nodes have the message of the leader with high probability.*

*Proof.* First we show that within $\rho \log\log n/2$ steps, the number of uninformed nodes decreases below $n/\sqrt[4]{d}$. The arguments are based on the proof of Lemma 2.2. from [ES08]. Let us consider some time step $t$ among these $\rho \log\log n/2$

steps. Given that all nodes have some degree $\Omega(d)$, a node chooses an incident edge not used so far (neither as outgoing nor as incoming edge) with probability $1 - \mathrm{O}(\log n/d)$. According to Lemma 1 of [Els06], this edge is connected to a node in $H(t)$ with probability at most $\mathrm{O}(p|H(t)| + \log n)/d)$, independently. Applying Chernoff bounds, we obtain that as long as $|H(t)| > n/\sqrt[4]{d}$, we have

$$|H(t+1)| \leq O\Big(|H(t)| \cdot \Big(\frac{|H(t)|}{n} + \frac{\log n}{d}\Big)\Big) \ ,$$

with high probability. Thus, after $\rho \log\log n/2$ steps, the number of uninformed nodes decreases with high probability below $n/\sqrt[4]{d}$, see also [KSSV00]. Applying now Lemmas 2.7. and 2.8. from [ES08] (for the statement of these lemmas see previous proofs), we obtain the lemma. Since only nodes of $H(t)$ open communication channels in a step, we obtain that the communication complexity produced during these pull long-steps is $\mathrm{O}(n)$, with high probability. □

**Lemma 23.** *After Phase II, the leader is aware of all messages in the network, with high probability.*

*Proof.* Let $w$ be some node, and we show by induction that the leader receives $m_w(0)$. Let $t$ be the long-step, in which $w$ receives $m_v(0)$. If $t = 1$, then $w$ is connected to $v$ in the communication tree rooted at $v$, and $v$ receives $m_w(0)$ in one of the last four steps of Phase II.

If $t > 1$, then let $w'$ denote the successor of $w$ in the communication tree rooted at $v$. That is, $w$ received $m_v(0)$ from $w'$ in long-step $t$. This implies that $w'$ either received $m_v(0)$ in pull long-step $t$ or $t-1$, or it received $m_v(0)$ in a push long-step $t-1$. If however, $w'$ obtained $m_v(0)$ in pull long-step $t$, then this happened before $w$ received the message. In both cases $w'$ will forward $m_w(0)$ to $v$ together with $m_{w'}(0)$, according to our induction hypothesis, and the lemma follows. □

**Lemma 24.** *After Phase III, gossiping is completed with high probability.*

The proof of Lemma 24 follows directly from Lemma 22. From the lemmas above, we obtain the following theorem.

**Theorem 25.** *With high probability, Algorithm 7.2 completes gossiping in* $\mathrm{O}(\log n)$ *time using* $\mathrm{O}(n)$ *message transmissions. If leader election has to be applied at the beginning, then the communication complexity is* $\mathrm{O}(n \log\log n)$.

Now we consider the robustness of our algorithm. We show that by applying our algorithm twice, independently, one can tolerate up to $n^{\epsilon'}$ random node failures, with high probability, where $\epsilon' < 1/4$. That is, during the execution of the algorithm, $n^{\epsilon'}$ nodes, chosen uniformly and independently at random, may fail at any time. The node failures are non-malicious, that is, a failed node does not communicate at all. The theorem below is stated for $p = \log^5 n/n$. However, with an extended analysis, the theorem can be generalized to any

$p > \log^{2+\epsilon} n/n$. As before, we assume that a random node acts as a leader. Since at most $n^{\epsilon'}$ random nodes fail in total, the leader fails during the execution of the algorithm with probability $n^{-\Omega(1)}$. Moreover, due to the robustness of the leader election algorithm from Section 7.1, the result of Theorem 26 also holds if leader election has to be applied to find a leader at the beginning.

**Theorem 26.** *Consider a $G(n, p)$ graph with $p = \log^5 n/n$. Assume that $f = n^{\epsilon'}$ random nodes fail according to the failure model described above, where $\epsilon' < 1/4$. If we run Algorithm 7.2 two times, independently, then at the end $n - |f|(1 + o(1))$ nodes know all messages of each other, with high probability.*

*Proof.* To analyze the process, we assume that all the failed nodes fail directly after Phase I and before Phase II. This ensures that they are recorded as communication partners for a number of nodes, but these failed nodes are not able to forward a number of messages in Phase II to the leader. Let us denote the two trees, which are constructed in Phase I of the two runs of the algorithm, by $T_1$ and $T_2$, respectively. First we show that with probability $1 - o(1)$ there is no path from a failed node to another failed node in any of these trees. Let us first build one of the trees, say $T_1$. Obviously, at distance at most $(1 - \epsilon') \log_4 n - \rho \log\log n$ from the root, there will be less than $n^{1-\epsilon'}/\log^2 n$ nodes. Thus, with probability $(1 - n^{\epsilon'}/n)^{n^{1-\epsilon'}/\log^2 n} = 1 - o(1)$, no node will fail among these $n^{1-\epsilon'}/\log^2 n$ many nodes. This implies that all the descendants of a failed node will have a distance of at most $\epsilon' \log_4 n + O(\log\log n)$ to this node. Then, the number of descendants of a failed node is $n^{\epsilon'} \log^{O(\log\log n)} n$, given that the largest degree is $\log^{O(1)} n$. As above, we obtain that none of the failed nodes is a descendant of another failed node with probability $1 - o(1)$.

For simplicity we assume that each failed node participates in at least one push long-step, that is, it contacts 4 neighbors and forwards the message of the leader (of $T_1$ and $T_2$, respectively) to these neighbors. Now we consider the following process. For each failed node $v$, we run the push-phase for $\epsilon' \log_4 n + O(\log\log n)$ long-steps. The other nodes do not participate in this push phase, unless they are descendants of such a failed node during these $\epsilon' \log_4 n + O(\log\log n)$ long-steps. That is, if a node is contacted in some long-step $i$, then it will contact 4 neighbors in long-step $i + 1$; in long-step 1 only the failed nodes are allowed to contact 4 neighbors. Then, we add to each node $w \neq v$ in the generated tree rooted at $v$ all nodes being at distance at most $\rho \log\log n$ from $w$. Clearly, the number of nodes in such a tree rooted at $v$ together with all the nodes added to it is $n^{\epsilon'} \log^{O(\log\log n)} n$. This is then repeated a second time. The nodes attached to $v$ in the first run are called the descendants of $v$ in $T_1$ in the following. Accordingly, the corresponding nodes in the second run are called the descendants of $v$ in $T_2$.

We consider now two cases. In the first case, let $v$ be a failed node, and assume that $v$ contacts four neighbors in $T_2$, which have not been contacted by $v$ in $T_1$. Such a failed node is called friendly. Furthermore, let $F(T_1)$ be the set of nodes which are either failed or descendants of a failed node in $T_1$. As shown above, $|F(T_1)| = n^{\epsilon'} \cdot n^{\epsilon'} \log^{\mathrm{O}(\log\log n)} n$, with probability $1 - \mathrm{o}(1)$. Let $v_i$, $i = 1, 2, \ldots$ be the descendants of $v$ in $T_2$, and denote by $A_i$ the event that $v_i \notin F(T_1)$. Then,

$$\Pr[\overline{A_i} \mid A_1 \ldots A_{i-1}] \leq \frac{\log^5 n \cdot n^{2\epsilon'} \log^{\mathrm{O}(\log\log n)} n}{n} < \frac{n^{2\epsilon'} \log^{\mathrm{O}(\log\log n)} n}{n} \quad .$$

Since $v$ has at most $n^{\epsilon'} \log^{\mathrm{O}(\log\log n)} n$ descendants, none of them belongs to $F(T_1)$, with probability at most $n^{3\epsilon'} \log^{\mathrm{O}(\log\log n)} n/n$. Thus, the expected number of descendants of friendly failed nodes in $F_1 \cap F_2$, is $\mathrm{o}(1)$, as long as $\epsilon' < 1/4$.

In the second case, we denote by $NF_1$ the set of nodes, which are direct descendants of non-friendly failed nodes. That is, $NF_1$ are the nodes which are contacted by non-friendly failed nodes in step 1 of the process described above. Since $p = \log^5 n/n$, a failed node is non-friendly with probability $\mathrm{O}\Big(1/\log^5 n\Big)$. Using standard Chernoff bounds, we have $|NF_1| = \mathrm{O}\Big(|f|/\log^5 n\Big)$, with high probability. Let now $NF_2$ denote the set of nodes which are either contacted by the nodes $NF_1$ in a push long-step of the original process in $T_1$ as well as in $T_2$, or contact a node in $NF_1$ in a pull long-step of the original process in both, $T_1$ and $T_2$. Similarly, $NF_{i+1}$ denotes the set of nodes which are either contacted by the nodes $NF_i$ in a push long-step of the original process in $T_1$ as well as in $T_2$, or contact a node in $NF_i$ in a pull long-step of the original process in both $T_1$ and $T_2$. We show that $|NF_{i+1}| < |NF_i|$ with high probability, and for any non-friendly failed node $v$ there is no descendant of $v$ in $NF_i$ with probability $1 - \mathrm{o}(1)$. The second result implies that $|NF_{\rho \log n}| = 0$ with high probability, if $\rho$ is large enough. The first result implies then that $\sum_{i=1}^{\rho \log n} |NF_i| < \mathrm{O}\Big(|f|/\log^3 n\Big)$, with high probability.

To show the first result we compute the expected value $E[NF_{i+1}]$ given $NF_i$. Clearly, a node contacted by a node of $NF_i$ in $T_1$ is contacted in $T_2$ as well with probability $\mathrm{O}\Big(1/\log^5 n\Big)$. Similarly, a node which contacted a node of $NF_i$ in $T_1$ contacts the same node in $T_2$ with probability $\mathrm{O}\Big(1/\log^5 n\Big)$. Simple balls-into-bins arguments imply that the number of nodes, which may contact the same node, is at most $\mathrm{O}(\log n/\log\log n)$ [HR90]. Applying now the method of bounded differences, we have $|NF_{i+1}| < |NF_i|$ with high probability, as long as $NF_i$ is large enough.

The arguments above imply that if the number of descendants of a non-friendly failed node in $NF_i$ is at least $\rho \log n$ for some $\rho$ large enough, then the number of descendants in $NF_{i+1}$ does not increase, with high probability. Furthermore, as long as the number of these descendants $NF_i$ is $\mathrm{O}(\log n)$, then there will be no descendants in $NF_{i+1}$ with probability $1 - \mathrm{o}(1)$, and the statement follows. Summarizing, $\sum_{i=1}^{\infty} NF_i = \mathrm{O}\Big(|f|/\log^3 n\Big)$ with high probability, which concludes the proof. □

# 8 Empirical Analysis

We implemented our algorithms using the C++ programming language and ran simulations for various graph sizes and node failure probabilities using four 64 core machines equipped with 512 GB to 1 TB memory running on Linux. The underlying communication network was implemented as an Erdős-Rényi random graph with $p = \log^2 n/n$. We measured the number of steps, the average number of messages sent per node, and the robustness of our algorithms.

## 8.1 Communication Complexity

The main result from Chapter 6 shows that it is possible to reduce the number of messages sent per node by increasing the run time. This effect can also be observed in Figure 8.1, where the communication overhead of three different methods is compared. The plot shows the average number of messages sent per node using a simple push-pull-approach, Algorithm 6.1, and Algorithm 7.2. In the simple push-pull-approach, every node opens in each step a communication channel to a randomly selected neighbor, and each node transmits all its messages through all open channels incident to it. This is done until all nodes receive all initial messages. Formally, the simple protocol is specified in Algorithm 8.1.

```
for $t = 1$ to $O(\log n)$ do
    at each node $v$ do in parallel
        open();
        pushpull($m_v$);
        $m_v \leftarrow m_v \cup$ receive();
        close();
```

**Algorithm 8.1:** a simple push-pull algorithm to perform randomized gossiping

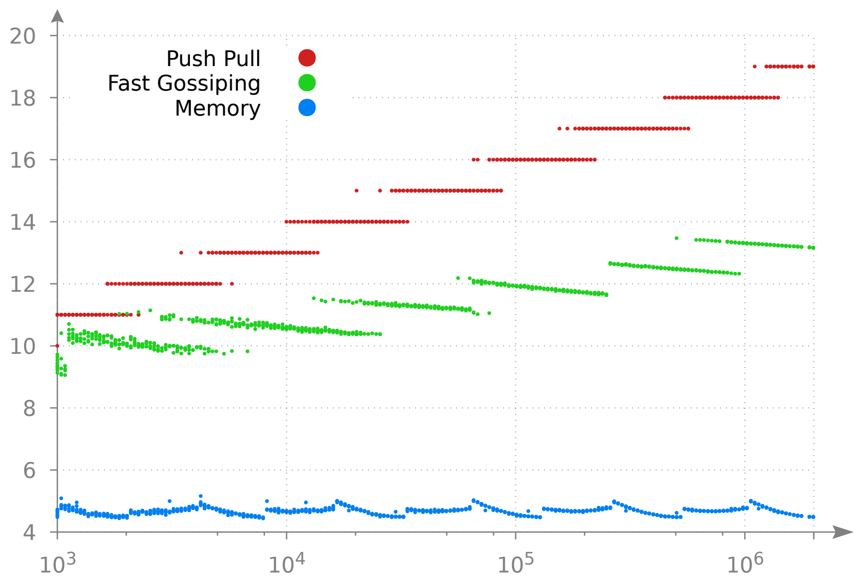


**Figure 8.1:** comparison of the communication overhead of the gossiping methods. The $x$-axis shows the graph size, the $y$-axis the average number of messages sent per node.

Figure 8.1 shows an increasing gap between the message complexity of Algorithm 6.1 and the simple push-pull approach. Furthermore, the data show that the number of messages sent per node in Algorithm 7.2 is bounded by 5. According to the descriptions of the algorithms, each phase runs for a certain number of steps. The parameters were tuned as described in Table 8.1 to obtain meaningful results.

The fact that the number of steps is a discrete value also explains the discontinuities that can be observed in the plot. In the case of the simple push-pull-approach, these jumps clearly happen whenever an additional step is required to finish the procedure. Note, that since in this approach each node communicates in every round, the number of messages per node corresponds to the number of rounds.

In the case of Algorithm 6.1, we do not only observe these jumps, but also a reduction of the number of messages per node between the jumps. Let us consider such a set of test runs between two jumps. Within such an interval, the number of random walk steps as well as broadcasting steps remain the same while $n$ increases. The number of random walks, however, is not fixed. Since each node starts a random walk with a probability of $1/\log n$, the *relative* number of random walks decreases and thus also the average number of messages per node (see also Figure 8.3). This shows the impact of the random walk phase on the message complexity.

The last phase of each algorithm was run until the entire graph was informed, even though the nodes do not have this type of global knowledge. From our data we observe that the resulting number of steps is concentrated (that is,

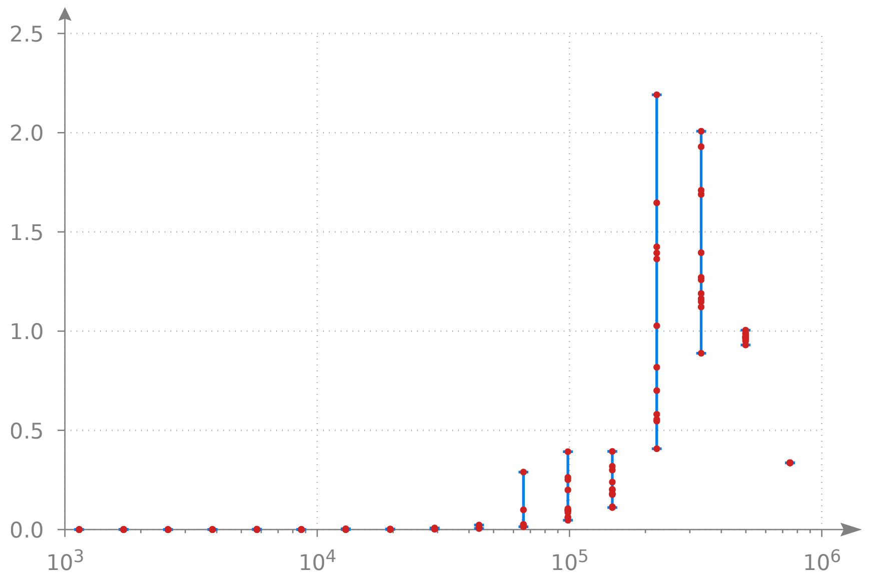


**Figure 8.2:** relative number of additional node failures in the memory model with a graph size of 1,000,000. The $x$-axis shows the number of nodes marked *failed* $F$, the $y$-axis the ratio of additional uninformed nodes to $F$.

| Phase | Limit | Value |
|---|---|---|
| | Algorithm 6.1 | |
| I | number of steps | $\lceil 1.2 \cdot \log\log n \rceil$ |
| II | number of rounds | $\lceil \log n / \log\log n \rceil$ |
| II | random walk probability | $1.0 / \log n$ |
| II | number of random walk steps | $\lceil \log n / \log\log n + 2 \rceil$ |
| II | number of broadcast steps | $\lceil 0.5 \cdot \log\log n \rceil$ |
| | Algorithm 7.2 | |
| I | first loop, number of steps (rounded to a multiple of 4) | $2.0 \cdot \log n$ |
| I | second loop, number of steps | $\lfloor 2.0 \cdot \log\log n \rfloor$ |
| II | number of steps | corresponds to Phase I |
| III | number of push steps | $\lfloor \log n \rfloor$ |

**Table 8.1:** the actual constants used in the simulation

for the same $n$ the number of steps to complete only differs by at most 1 throughout all the simulations). Furthermore, no jumps of size 2 are observed in the plot. Thus, overestimating the obtained run time by 1 step would have been sufficient to complete gossiping in all of our test runs.

## 8.2 Robustness of the Memory Model

To gain empirical insights into the behavior of the memory-based approach described in Chapter 7 under the assumption of node failures, we implemented nodes that are marked as failed. These nodes simply do not store any incoming message and refuse to transmit messages to other nodes.

The plot in Figure 8.2 shows the results of simulations on an Erdős-Rényi random graph consisting of 1,000,000 nodes with an expected node degree of $\log^2 n \approx 400$. Our simulation of Algorithm 7.2 constructed 3 message distribution trees, independently. Afterwards we marked $F$ nodes chosen uniformly at random as failed. The nodes were deactivated before Phase II. The $x$-axis in Figure 8.2 shows this number of nodes $F$. In the simulation, we determined the number of initial messages that have been lost in addition to the messages of the $F$ marked nodes. Figure 8.2 shows on the $y$-axis the ratio of the lost messages of healthy nodes over $F$. That is, zero indicates that no additional initial message was lost, whereas 2.0 indicates that for every failed node the initial messages of at least two additional healthy nodes were not present in any tree root after Phase II.

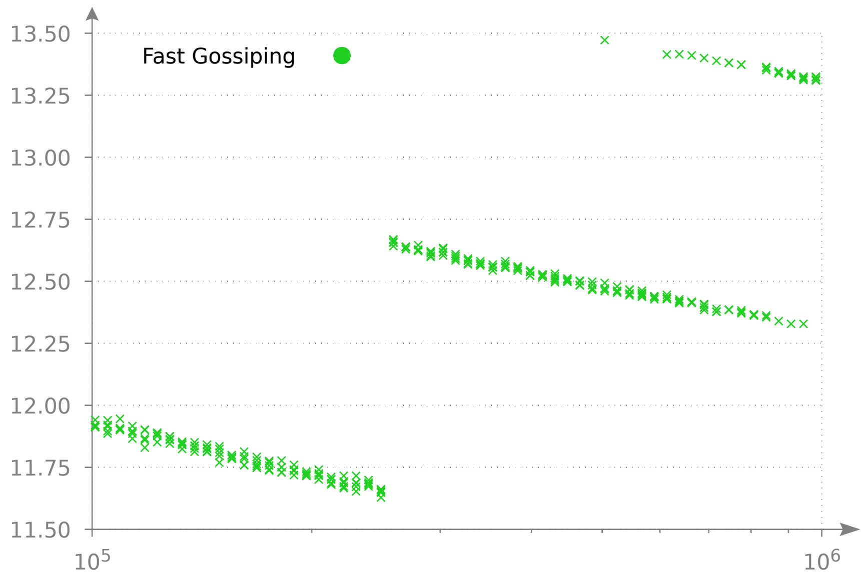
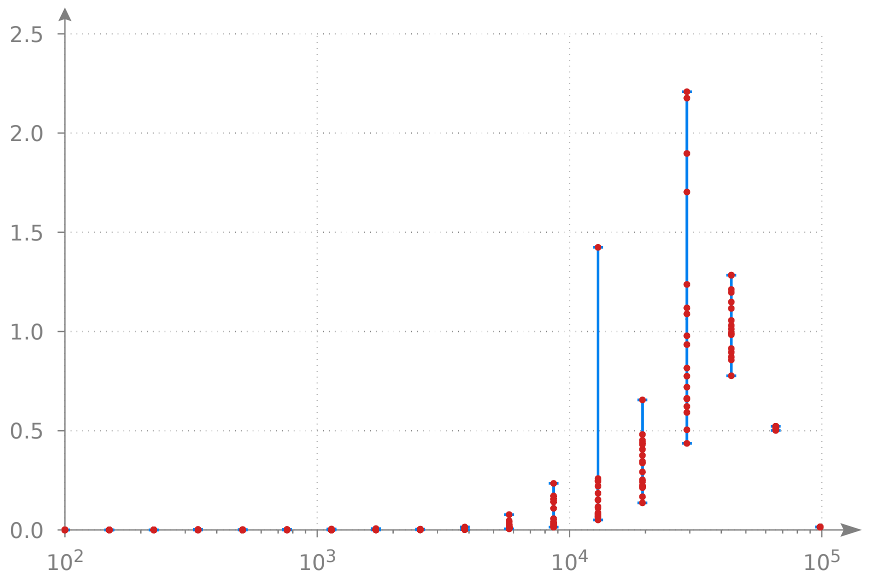


**Figure 8.3:** a more detailed view of the data presented in Figure 8.1 for Algorithm 6.1. The $x$-axis shows the graph size, the $y$-axis the number of messages sent per node.

Further plots showing additional graph sizes and various levels of detail can be found in Figure 8.4. The corresponding simulations were run on graphs of size 100,000 and 500,000 nodes, respectively. They visualize the results of the same type of simulation as presented in Figure 8.2.

Even more details can be obtained from the plots shown in Figure 8.5, where we ran our simulation with a higher resolution. That is, we ran a series of at least 5 tests per number of failed nodes. The number of failed nodes was chosen from the set $\{0, 100, 200, 300, \dots\}$. We used graphs of two different sizes in these 6 plots. The left column shows the results for a graph consisting of 100,000 nodes and the right column for 500,000 nodes. The $x$-axis shows the number of failed nodes, the $y$-axis shows the percentage of runs in which more than a certain number $T$ of additional nodes failed. This number is $T = 0$ for the top row, $T = 10$ for the middle row and $T = 100$ for the bottom row. For example, this tells us that on a graph of size 100,000 more than 4000 nodes could fail and still the number of additional uninformed nodes was less than 100 in all test runs.



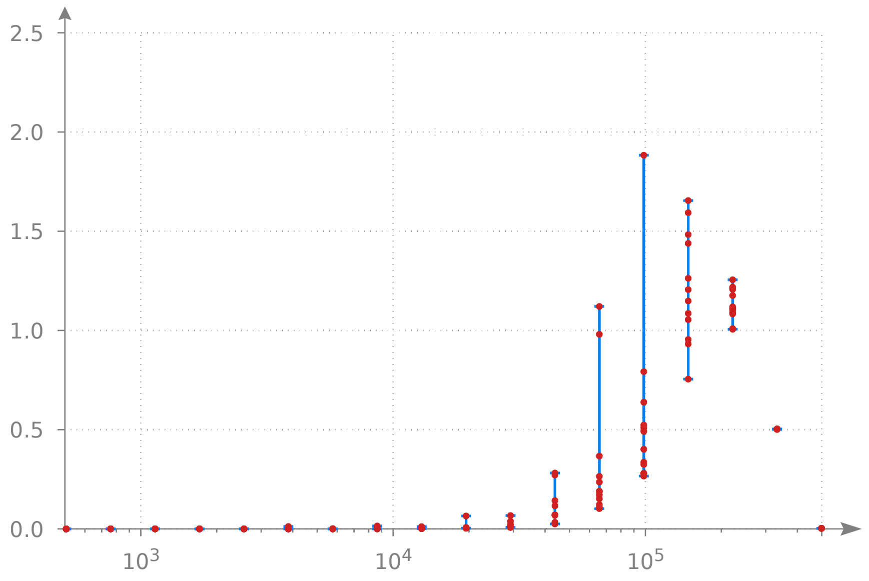


**Figure 8.4:** relative number of additional node failures in the memory model with graphs of sizes 100,000 (top) and 500,000 (bottom). The $x$-axis shows the number of failed nodes $F$, the $y$-axis the ratio of additional uninformed nodes to $F$.

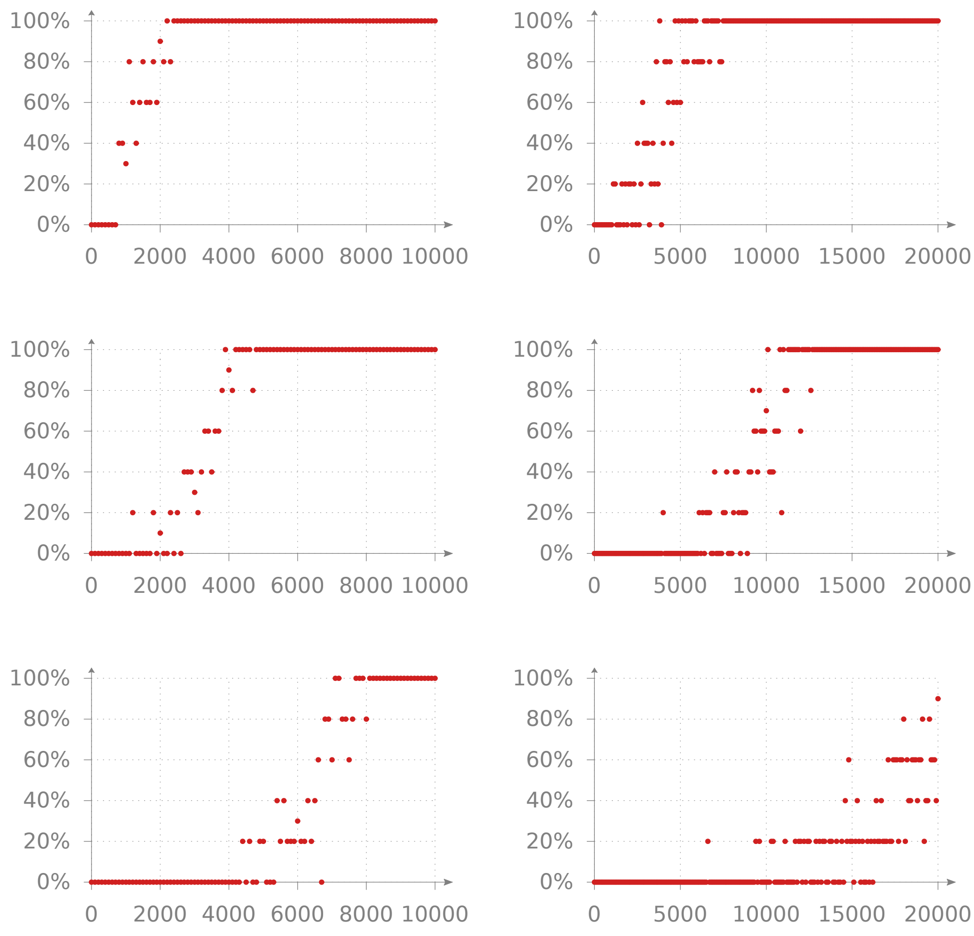


**Figure 8.5:** detailed plot showing the robustness of Algorithm 7.2 on graphs of sizes 100,000 (left column) and 500,000 (right column). The $x$-axis shows the number of failed nodes, the $y$-axis shows the percentage of runs in which more than $T$ additional numbers remained uninformed. In the top $T = 0$, in the middle $T = 10$ and in the bottom $T = 100$.

A

# Additional Lemmas

For the sake of completeness, we state additional lemmas that we used for our analysis in this appendix.

## A.1 Additional Lemmas from [Els06]

For some $u, v$ let $A_{u,v}$ denote the event that $u$ and $v$ are connected by an edge, and let $A_{u,v,l}$ denote the event that $u$ and $v$ share an edge **and** $u$ chooses $v$ in step $l$ (according to the random phone call model). In the next lemma, we deal with the distribution of the neighbors of a node $u$ in a graph $G(n,p)$, after it has chosen $t$ neighbors, uniformly at random, in $t = \mathrm{O}(\log n)$ consecutive steps. In particular, we show that the probability of $u$ being connected with some node $v$, not chosen within these $t$ steps, is not substantially modified after $\mathrm{O}(\log n)$ steps.

**Lemma 1 from** [Els06]**.** *Let $V = \{v_1, \dots, v_n\}$ be a set of $n$ nodes and let every pair of nodes $v_i, v_j$ be connected with probability $p$, independently, where $p \geq \log^\delta n/n$ for some constant $\delta > 2$. If $t = \mathrm{O}(\log n)$, $u, v \in V$, and*

$$A(U_0, U_1, U_2) = \bigwedge_{\substack{0<l\leq t \\ (v_i,v_j,l)\in U_0}} A_{v_i,v_j,l} \bigwedge_{(v_{i'},v_{j'})\in U_1} A_{v_{i'},v_{j'}} \bigwedge_{(v_{i''},v_{j''})\in U_2} \overline{A_{(v_{i''},v_{j''})}} \ ,$$

*for some $U_0 \subset V \times V \times \{0, \dots, t\}$ and $U_1, U_2 \subset V \times V$, then it holds that*

$$\Pr[(u,v) \in E \mid A(U_0, U_1, U_2)] = p(1 \pm \mathrm{O}(t/d)) \ ,$$

*for any $U_0, U_1, U_2$ satisfying the following properties:*

- $|U_0 \cap \{(v_i, v_j, l) | v_j \in V\}| = 1$ *for any* $v_i \in V$ *and* $l \in \{0, \dots, t\}$,
- $|U_1 \cap \{(u, u') | u' \in V\}| = \Omega(d)$ *and* $|U_1 \cap \{(v, v') | v' \in V\}| = \Omega(d)$,
- $(u, v) \notin U_1 \cup U_2$, *and* $(u, v, i) \notin U_0$ *for any* $i$.

## A.2 Additional Lemmas from [ES08]

**Lemma 2.2 from** [ES08] (adapted version)**.** *Let Algorithm 7.2 be executed on the graph $G(n,p)$ of size $n$, where $p > \log^{2+\Omega(2)} n/n$ and $\rho$ is a properly chosen (large) constant. If $t = \log n + \frac{\rho}{2} \log\log n$, then $|H(t)| \leq n/\sqrt[4]{d}$ and the number of transmissions after $t$ time steps is bounded by $\mathrm{O}(n \log\log n)$. Additionally, if $t = \log n + \frac{3\rho}{8} \log\log n$, we have $|H(t)| \geq n/\sqrt[4]{d}$.*

**Lemma 2.7 from** [ES08] (adapted version)**.** *Let $|H(t)| \in [\log^q n, n/\sqrt[4]{d}]$ be the number of uninformed nodes in $G(n,p)$ at some time $t = \mathrm{O}(\log n)$, where $q$ is a large constant, and let Algorithm 7.2 be executed on this graph. Then, $|H(t + 3\rho \log\log n/8)| \leq \log^q n$, with high probability, provided that $\rho$ is large enough.*

**Lemma 2.8 from** [ES08] (adapted version)**.** *Let $|H(t)| \leq \log^q n$ be the number of uninformed nodes in $G(n,p)$ at time $t = \mathrm{O}(\log n)$, and let Algorithm 7.2 be executed on this graph. Then within additional $\rho \log\log n/8$ steps all nodes in the graph will be informed, with high probability, whenever $\rho$ is large enough.*

# Part II

# Load Balancing

Hoda Akbari, Petra Berenbrink, Robert Elsässer, and Dominik Kaaser: *Discrete Load Balancing in Heterogeneous Networks with a Focus on Second-Order Diffusion.* In *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015, pages 497–506. DOI: 10.1109/ICDCS.2015.57.

# 9

# Load Balancing

Load balancing is a fundamental task in many parallel and distributed applications. Often there are significant differences in the amount of work load generated on the processors of a parallel machine, which have to be balanced in order to obtain a substantial benefit w.r.t. the runtime of a parallel computation. One of the most prominent examples are so-called finite element simulations [FWM94].

In the load balancing problem we are given an interconnection network and a number of load items which are arbitrarily distributed over the nodes of the network. The goal is to redistribute the items such that at the end each node has (almost) the same load. To achieve this goal, nodes are only allowed to communicate with their direct neighbors. We assume that each node has access to a global clock, and the algorithm works in synchronous rounds.

A prominent class of load balancing algorithms are so-called diffusion schemes [DFM99]. In these algorithms, the nodes are allowed to balance their load with all their neighbors simultaneously in a round. As already said in the introduction, we distinguish between *continuous* and *discrete* settings. In the continuous case it is assumed that the load can be split into arbitrarily small pieces. Although often not realistic, this assumption is very helpful for analyzing these algorithms [DFM99]. Discrete load balancing algorithms, on the other hand, assume that tasks are atomic units of load, called *tokens*. Hence, two adjacent nodes cannot balance their load any way they want; only integral amounts of load can be transferred. As a consequence, discrete diffusion algorithms are usually not able to balance the load completely [EMS06, ABS16].

Two fundamental diffusion type algorithms are the first order scheme (FOS) and the second order scheme (SOS) [MGS98]. In FOS the amount of load that nodes send to their neighbors in a step only depends on their current load difference. In SOS the flow over an edge is a function of the current load difference between its incident nodes and the load that was sent in the previous round. Note that SOS can lead to negative load at some nodes if the loads of the nodes are not sufficient to fulfill the calculated demand of all edges. There are tight bounds on the worst-case convergence time of both, FOS and SOS, in the continuous case [DFM99]. In general, for the optimal choice of parameters SOS converges much faster than FOS.

The common approach for analyzing discrete diffusion algorithms is to consider a closely related continuous version of the algorithm and to bound the load deviation between load vectors of the two processes ([RSW98]). To explain the approach we need a couple of definitions first. We assume that the network is modeled by an undirected graph $G = (V, E)$, where $V = \{1, \dots, n\}$ represents the set of processors and the edges in $E$ describe (physical or virtual) connections between them. A total of $m$ identical load items are distributed over the nodes. We use a vector $x = (x_1, \dots, x_n)$ to indicate the amount of load assigned to every node. In the *heterogeneous* network model the nodes may have different speeds $(s_1, \dots, s_n)$. The aim of a load balancing algorithms is to distribute the load proportional to the processors' speeds. Hence, the ideal load of a node $i$ is $\bar{x}_i = ms_i/s$, where $s = \sum_{i=1}^{n} s_i$. The deviation of a load vector $x$ from another load vector $x'$ is $\max_{i \le n} |x_i - x'_i|$.

In the case of the common approach mentioned above the continuous process would forward a fractional amount of load $\ell_e$ over some edge $e$, the discrete algorithm rounds $\ell_e$ to an integer $\ell'_e$. The rounding can be done deterministically or randomized, whereas randomized rounding often outperforms deterministic rounding (for example, the always round down approach [SS12]). The difference between $\ell_e$ and $\ell'_e$ is called the *rounding error*. The propagation of the rounding errors causes the two processes to deviate from each other.

## Outlook

The remainder of this part is organized as follows. In Section 9.1 we formally define the model and elaborate on related work. Then, in Section 9.2 we give an overview over the three main results from [ABEK15]. In Chapter 10, we first present the general framework for randomly rounding continuous diffusion schemes to discrete schemes from [ABEK15]. Compared to the results of [RSW98], which are only valid w.r.t. the class of homogeneous first order schemes, this framework can be used to analyze a larger class of diffusion algorithms, such as algorithms for heterogeneous networks and second order schemes. Secondly, we state bounds on the deviation between randomized second order schemes and their continuous counterparts. Additionally, we state a bound for the minimum initial load in a network that is sufficient to prevent the occurrence of negative load at a node during the execution of second order diffusion schemes. The full proofs of the results stated in Chapter 10 can be found in [ABEK14a] and in [ABEK15].

Finally, we present our empirical analysis and simulation results for various graph classes in Chapter 11, comparing the performance of FOS and SOS and giving an empiric insight into the behavior of diffusion based load balancing processes. For the simulation, we implemented a network and simulated both, FOS and SOS load balancing processes. Especially in tori, our results show a clear advantage of SOS over FOS w.r.t. the number of steps required to balance the loads. We also empirically analyze the remaining imbalance that arises in discrete load balancing schemes once the system has converged such that no node has more than a constant number of additional load tokens. We propose to switch from SOS to FOS once this threshold is reached, and our simulations show that this change of the scheme leads to a further drop of the remaining load imbalance.

## 9.1 Models

### First Order Diffusion (FOS)

The first order scheme, FOS, was independently introduced by [Cyb89] and [Boi90]. FOS in the *homogeneous* network model is defined as follows. Let $N(i)$ be the set of neighbors of node $i$ and $d_i$ be its degree. We define $x(t) = (x_1(t), \ldots, x_n(t))$ as the *load vector* at the beginning of round $t \geq 0$, where $x_i(t)$ is the load of node $i$. The amount of load transferred from node $i$ to node $j$ in round $t$ is denoted by $y_{i,j}(t)$. Then FOS is characterized by the following equations, where $\alpha_{i,j}$ is a parameter, usually $\alpha_{i,j} = 1/(\max(d_i, d_j) + 1)$.

$$\begin{aligned} y_{i,j}(t) &= \alpha_{i,j} \cdot \Big(x_i(t) - x_j(t)\Big) \\ x_i(t+1) &= x_i(t) - \sum_{j \in N(i)} \alpha_{i,j} \Big(x_i(t) - x_j(t)\Big) \end{aligned}$$

The process can be expressed by means of a *diffusion matrix* $M$, where $M_{i,i} = 1 - \sum_j \alpha_{i,j}$ and $M_{i,j} = \alpha_{i,j}$ for $j \in N(i)$. All other entries of $M$ are zero. Then

$$x(t+1) = M \cdot x(t) \ , \tag{9.1}$$

where $M$ is a symmetric doubly stochastic $n \times n$ matrix that can be viewed as the transition matrix of an ergodic Markov chain with uniform steady-state distribution. Hence, repeatedly applying the equation leads to the perfectly balanced state. Let $K$ denote the difference between the maximum and minimum load at the beginning of the process. Let $\lambda$ denote the second-largest eigenvalue (in magnitude) of $M$. Then [MGS98, RSW98] show that FOS converges in $O(\log(Kn)/(1-\lambda))$ rounds. In [RSW98] the authors introduce a framework to analyze a wide class of discrete FOS processes. This framework served as a foundation for analyzing several discrete FOS algorithms. Many of these publications consider uniform processors [BCF+15, BFH09, EM03, ES10, FGS12, FS09, GM96, MGS98, RSW98, SS12], while a few others incorporate processor speeds into the model [AB12, EMS06]. The authors of [FGS12] consider a discrete process where the continuous flow is rounded randomly.

This algorithm achieves a deviation bound of $\mathrm{O}((d \log\log n)/(1-\lambda))$. The drawback of this method is that rounding up on too many edges might result in negative load. The process of [BCF+15] avoids negative load. A node first rounds down all the flows on the adjacent edges, which leaves some surplus tokens which are randomly distributed among the neighbors. This algorithm achieves a deviation bound of $\mathrm{O}\Big(d\sqrt{\log n} + \sqrt{(d \log n \log d)/(1-\lambda)}\Big)$. In [SS12] the authors study two natural discrete diffusion-based protocols and their discrepancy bounds depend only polynomially on the maximum degree of the graph and logarithmically on $n$.

The balancing process of [ABS16] simulates a continuous process using a corresponding discrete process. In every round the discrete flow on each edge is determined such that it stays as close as possible to the total continuous flow that is sent over the edge. This process results in a deviation of $\mathrm{O}(d)$ (for a more detailed description see next section). In [ES10] the authors consider an approach that is based on random walks where tokens of overloaded nodes use a random walk to reach underloaded nodes. While this approach leads to a situation at the end, in which no node has more than a constant number of tokens above average [ES10], it needs to keep track of the load traffic the continuous scheme would produce. Moreover, the corresponding random walks of the tokens result in a huge amount of load transmissions between the nodes, which is not the case in diffusion based schemes [DFM99].

## Second Order Diffusion (SOS)

Muthukrishnan et al. [MGS98] introduce the continuous second order scheme which is based on a numerical iterative method called successive over-relaxation [GV61] and is one of the fastest diffusion load balancing algorithms. In SOS, the amount of load transmitted over each edge depends on the current load as well as the load transferred in the previous round. The only exception is the very first round in which FOS is applied. Subsequent rounds follow the equations below.

$$\begin{aligned} y_{i,j}(t) &= (\beta-1)\; y_{i,j}(t-1) + \beta\alpha_{i,j}\Big(x_i(t) - x_j(t)\Big) \qquad (9.2)\\ x_i(t+1) &= \beta \cdot \left( x_i(t) - \sum_{j\in N(i)} \alpha_{i,j}\Big(x_i(t) - x_j(t)\Big)\right) \\ &\quad + (1-\beta)\cdot x_i(t-1) \end{aligned}$$

Here, $\beta$ is independent of the iteration number $t$. From the above equations we get

$$x(t+1) = \begin{cases} M\,x(t) & \text{if } t = 0 \\ \beta \cdot M\,x(t) + (1-\beta)\cdot x(t-1) & \text{if } t > 0 \end{cases} \qquad (9.3)$$

For the process to converge, $\beta$ must be in $(0,2)$. For the optimal choice of $\beta_{\text{opt}} = 2/(1+\sqrt{1-\lambda^2})$ SOS converges in $\mathrm{O}\Big(\log(Kn)/\sqrt{1-\lambda}\Big)$ rounds [MGS98] which is in general faster than FOS; for graphs with some eigenvalue gap $(1-\lambda)^{-1} = \log^{\omega(1)} n$, the convergence time of SOS is almost quadratically faster than FOS. Unfortunately, it can happen that the total outgoing flow from a node exceeds its current load, which results in so-called *negative load*.

### Heterogeneous Networks

Continuous FOS and SOS processes in the *heterogeneous network model* were first studied in [EMP02]. In heterogeneous networks, processors have different speeds and the aim is to distribute the load proportional to their speeds. The minimum speed is 1, the maximum speed is $s_{\max}$, and $s = s_1 + \cdots + s_n$. Let the diagonal matrix $S$ be defined by $S_{i,i} = s_i$. Then the heterogeneous FOS/SOS processes are defined as before, see equation (9.1) and equation (9.3)), respectively, except the diffusion matrix is now $M = I - LS^{-1}$ where $L$ is the normalized Laplacian matrix of the graph [EMP02]. In [AB12], the authors analyze a discrete FOS for homogeneous networks. In [EMP02] the authors show that continuous FOS/SOS processes converge in $\mathrm{O}(\log(Kns_{\max})/(1-\lambda))$ and $\mathrm{O}\Big(\log(Kns_{\max})/\sqrt{1-\lambda}\Big)$ rounds, respectively. In [EMS06], the authors consider a discrete version of SOS too. They show that the euclidean distance between the discrete and continuous load vectors in the discrete version is $\mathrm{O}(d \cdot \sqrt{n \cdot s_{\max}}/(1-\lambda))$.

## 9.2 New Results from [ABEK15]

**Result I.** In [ABEK15], we presented a general framework for rounding continuous diffusion schemes to discrete schemes. Our approach estimates the error between a continuous diffusion scheme and the rounded discrete version first, similar to [RSW98]. Then we combine that error term with martingales techniques similar to the ones used in [BCF+15] to bound the deviation between the continuous scheme and a discrete scheme based on randomized rounding. Note that the results in [RSW98] are only valid for a class of homogeneous first order schemes and [BCF+15] analyzes a fixed first order diffusion scheme with a specific transition matrix. The error estimation introduced in [ABEK15] allows to derive results for a larger class of diffusion algorithms (see Definition 4) in heterogeneous networks, including SOS.

In the homogeneous case the bounds from [ABEK15] are the same as the best results for FOS. The bound is worse than the $\mathrm{O}(d)$ bound from [ABS16]. In [ABEK15], we bound the deviation of a class of very natural and stateless algorithms. That is, the amount of load that is forwarded over an edge in step $t$ only depends on the load at the beginning of step $t$ and the amount that was sent in step $t-1$. The approach of [ABS16] is not stateless as it simulates the continuous process. They take the difference of the cumulative load that was sent by the continuous process and the cumulative load that was sent by the discrete process so far into account.

**Result II.** In the second result from [ABEK15], we showed that randomized SOS has a deviation (after the balancing time of continuous SOS) of $\mathrm{O}\Big(d \cdot \log s_{\max} \cdot \sqrt{\log n}/(1-\lambda)^{3/4}\Big)$, where $\lambda$ is the second largest eigenvalue of $M$ and $s_{\max}$ is the maximum speed. Note that, assuming optimal $\beta$, the runtime of SOS is in most cases much better than the runtime of FOS, more precisely, $\mathrm{O}\Big(\log(Kn)/\sqrt{1-\lambda}\Big)$ compared to $\mathrm{O}(\log(Kn)/(1-\lambda))$.

**Result III.** Finally, we showed in [ABEK15] that the continuous second order scheme with optimal $\beta$ will not generate negative load if at time $t = 0$ the minimum load of every node is at least $\mathrm{O}\Big(\sqrt{n} \cdot \Delta/\sqrt{1-\lambda}\Big)$, where $\Delta$ is the difference between the initial maximum load and the average load. For discrete SOS and graphs with proper eigenvalue gap we showed a bound of $\mathrm{O}\Big((\sqrt{n} \cdot \Delta(0) + d^2)/\sqrt{1-\lambda}\Big)$.

# 10 Framework for First Order and Second Order Diffusion Schemes

In this chapter we state the results from [ABEK15] where we first generalized the framework of Rabani et al. [RSW98] to a wider class of processes, see Section 10.1, in order to obtain an estimation of the deviation of the discrete process from its continuous version. The estimation is valid as long as the continuous process is *linear* according to Definition 4. In [RSW98] the deviation is expressed in terms of the diffusion matrix. In [ABEK15], however, we presented an analysis from a different perspective which allowed us to obtain essentially the same deviation formula for a larger class of processes. Our analysis can be applied to second order processes and heterogeneous models as well. In Section 10.2 we present the framework from [ABEK15] to transform a continuous load balancing process $C$ into a discrete process $R(C)$ using randomized rounding.

For simplicity we initially consider only first order schemes. Then, in Section 10.3 and Section 10.4 we generalize the framework to second order schemes.

## 10.1 First Order Diffusion Schemes

For a load balancing process $A$, we use $x_i^A(t)$ to denote the load of a node $i$ at the beginning of the round $t$, and $x^A(t) = (x_1^A(t), \dots, x_n^A(t))$. For $j \in N(i)$, we define $y_{i,j}^A(t)$ as the amount of load sent from $i$ to $j$ in round $t$. Observe that this value is negative if load items are transferred from $j$ to $i$. In this definition, $N(i)$ represents the set of neighbors of node $i$. Then $y^A(t)$ is the matrix with $y_{i,j}^A(t)$ as its entry in row $i$ and column $j$. Note that each balancing process $A$ can be regarded as a function that, given the current state of the network, determines for every edge $e$ and round $t$ the amount of load that has to be transferred over $e$ in $t$. Hence, we can regard $y^A(t)$ as the result of applying a function $A$ on the load vector, that is, $y^A(t) = A(x^A(t))$. Based on these observations we formally define the discrete load balancing process as follows.

**Definition 3.** *Let $C$ be a continuous process. A process $D$ is said to be a discrete version of $C$ with rounding scheme $R_D$ if for every vector $\mathbf{x}$, we have $D(\mathbf{x}) = R_D(C(\mathbf{x}))$ where $R_D$ is a function that rounds each entry of the matrix to an integer.*

A straight forward example for such a rounding scheme would be the rule to always round down. Alternatively, one could decide randomly between rounding up or down. In Section 10.2, we will describe the more elaborate algorithm called *randomized rounding* from [ABEK15] that can be applied to a wide range of continuous load balancing schemes.

Note that any load balancing process must fulfill the property of load conservation, that is, the total load present in the system at a round $t$ must not diverge from the initial total load. Furthermore, the analysis performed in [ABEK15] requires that the process exhibits a *linearity* property according to the following definition.

**Definition 4** (Linearity)**.** *A diffusion process $A$ is said to be* linear *if for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$ and $a, b \in \mathbb{R}$ we have $A(a\mathbf{x} + b\mathbf{x}') = a \cdot A(\mathbf{x}) + b \cdot A(\mathbf{x}')$.*

**Lemma 27.** *Both, FOS and SOS defined in Section 9.1 are linear.*

For the following definition, we will use $\hat{\mathbf{i}}$ to denote the unit vector of length $n$ which has 1 as its $i$-th entry and 0 for all other entries.

**Definition 5** (Contributions)**.** *Let $\mathbf{x}$ and $\mathbf{x}'$ be the load vectors obtained from applying $C$ for $t$ rounds on $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$, respectively. For two fixed nodes $i$ and $k$ and $j \in N(i)$ the* contribution *of edge $(i, j)$ on node $k$ after $t$ rounds is defined as*

$$\mathcal{C}^C_{k,i\to j}(t) = \mathbf{x}_k - \mathbf{x}'_k \quad .$$

The next lemma provides a general form of the FOS deviation formula of [RSW98] which serves as a basis for analyzing several discrete FOS processes. Let $C$ be a continuous process and $D$ its discrete version. Let furthermore be the *rounding error* defined as $e_{i,j}(t) = \hat{Y}_{i,j}(t) - y^D_{i,j}(t)$, where $\hat{Y}(t)$ represents $C(x^D(t))$.

**Lemma 28.** *Consider a linear diffusion process $C$ and its discrete version $D$ with an arbitrary rounding scheme. Then, for an arbitrary node $k$ and round $t$ we have*

$$x^D_k(t) - x^C_k(t) = \sum_{s=1}^{t} \sum_{\{i,j\}\in E} e_{i,j}(t-s)\, \mathcal{C}^C_{k,i\to j}(s) \quad .$$

## 10.2 Randomized First Order Scheme

In this section we give an overview over the analysis of the randomized rounding scheme for a general class of continuous load balancing algorithms from [ABEK15]. The technique is based on the results by Berenbrink et al. [BCF+15] where a fixed discrete FOS process using randomized rounding is analyzed for for homogeneous $d$-regular graphs. Their algorithm is based on a continuous process in which every node sends a $1/(d+1)$-fraction of its load to each neighbor. Initially, the discrete algorithm rounds $x_i/(d+1)$ down if it is not an integer. This leaves $(d+1) \cdot \lfloor x_i/(d+1) \rfloor$ surplus tokens on node $i$, which they call *excess* tokens. The excess tokens are then distributed by sending the tokens to neighbors which are uniformly sampled without replacement.

In [ABEK15] we applied this technique in a much more general way. We introduced a randomized framework that converts a general class of continuous processes to their discrete versions using randomized rounding.

### The Randomized Rounding Algorithm

For $a \in \mathbb{R}$ we will use $\{a\}$ to denote the fractional part $a - \lfloor a \rfloor$. Let $i$ be an arbitrary but fixed node and let $\hat{Y}(t)$ represent $C(x^D(t))$. For each edge $e = \{i, j\}$ let the corresponding $\hat{Y}_{i,j}(t)$ be the load that would be sent over $e$ by the continuous process $C$. The rounding scheme works as follows. First, it rounds $\hat{Y}_{i,j}(t)$ down for all the edges. This leaves $r = \sum_{j:\hat{Y}_{i,j}(t)\geq 0}\{\hat{Y}_{i,j}(t)\}$ excess tokens on node $i$. Then it takes $\lceil r \rceil$ additional tokens and sends each of them out with a probability of $r/\lceil r \rceil$, independently. With the remaining probability the excess tokens remain on node $i$. The tokens which do not remain on $i$ are sent to a neighbor $j$ with a probability of $\{\hat{Y}_{i,j}(t)\}/r$. Formally, this algorithm is specified in Algorithm 11.1 in Chapter 11.

The deviation bound from [ABEK15] is expressed in terms of the *refined local divergence* $\Upsilon^C(G)$, which is a function of both the algorithm and the graph, defined as follows.

$$\Upsilon^C(G) = \max_{k\in V}\left(\sum_{s=0}^{\infty}\sum_{i=1}^{n}\max_{j\in N(i)}\left(\mathcal{C}^C_{k,i\to j}(s)\right)^2\right)^{1/2}$$

Observe that $\Upsilon^C(G)$ is a generalization of the refined local divergence $\Upsilon(G)$ introduced in [BCF+15]. Based on this refined local divergence we showed in [ABEK15] the following result.

**Lemma 29.** *Let $C$ be a continuous FOS and let $R = R(C)$ be a discrete FOS using our randomized rounding algorithm. In an arbitrary round $t$ we have with high probability*

$$\left|X^R_k(t) - x^C_k(t)\right| = \mathrm{O}\left(\Upsilon^C(G)\cdot\sqrt{d\log n}\right) .$$

The proof of Lemma 29 relies on the fact that FOS is a linear process, see Lemma 27, and therefore a similar analysis as in [BCF+15] can be conducted. The difference is that in [ABEK15] we use the contributions $\mathcal{C}^{C}_{k,i\to j}(t)$ instead of the diffusion matrix.

Using Lemma 29 one can also obtain concrete results for randomized FOS processes as stated in the following corollaries. The following result holds for the homogeneous case and a special class of algorithms where $\alpha_{i,j} = 1/(\gamma d)$. Recall that $d$ is the maximum degree. The same result was already shown in [SS12].

**Corollary 30.** *Assume $s_1 = \cdots = s_n$ and $\alpha_{i,j} = 1/(\gamma d)$. Let $C$ be a continuous FOS process and let $R = R(C)$ be a discrete FOS process based on the randomized rounding algorithm applied on $C$. Then*

*(1)* $\Upsilon^{\mathrm{C}}(G) = \mathrm{O}\left(\sqrt{\gamma d/(2-2/\gamma)}\right)$ .

*(2) For any round $t$ we have with high probability*

$$\left|x^{R}_{k}(t) - x^{\mathrm{C}}_{k}(t)\right| = \mathrm{O}\left(\sqrt{\frac{\gamma d}{2-2/\gamma}} \cdot \sqrt{d\log n}\right) .$$

In [SS12] the authors applied a potential function in order to estimate $\Upsilon^{C}(G)$. This proof relies heavily on the fact that the transition probabilities are uniform over all edges, which is not the case for the heterogeneous model or when $\alpha_{i,j}$ depends on $d_i$ and $d_j$. The next result from [ABEK15] is more general and applies to these cases as well.

**Theorem 31.** *Let $C$ be a continuous FOS process and let $R = R(C)$ be a discrete FOS process based on the randomized rounding algorithm applied on $C$. Then*

*(1)* $\Upsilon^{\mathrm{C}}(G) = \mathrm{O}\left(\sqrt{d \cdot \log s_{\max}/(1-\lambda)}\right)$ .

*(2) For any round $t$ we have with high probability*

$$\left|x^{R}_{k}(t) - x^{\mathrm{C}}_{k}(t)\right| = \mathrm{O}\left(d \cdot \sqrt{\frac{\log n \cdot \log s_{\max}}{1-\lambda}}\right) .$$

## 10.3 Second Order Diffusion Schemes

In this section we state the results from [ABEK15] for SOS, where we showed that after some slight adjustments the framework from Section 10.1 can be applied to second order processes on heterogeneous networks as well. Recall that for a second order process $C$ the flow $y^{C}(t)$ is determined based on $x^{C}(t)$ and $y^{C}(t-1)$. More formally, $y^{C}(t) = C(x^{C}(t), y^{C}(t-1))$. Thus, Definition 6 and Definition 7 for SOS from [ABEK15] for *linearity* and *contributions*, respectively, also incorporate $y^{C}(t-1)$. We again use $\hat{\mathbf{i}}$ to denote the unit vector of length $n$ which has 1 as its $i$-th entry and 0 for all other entries.

**Definition 6** (Linearity)**.** *A diffusion process $A$ is said to be* linear *if for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^{n\times n}$ and $a, b \in \mathbb{R}$ we have $A(a\mathbf{x} + b\mathbf{x}', a\mathbf{y} + b\mathbf{y}') = aA(\mathbf{x}, \mathbf{y}) + bA(\mathbf{x}', \mathbf{y}')$.*

**Definition 7** (Contributions)**.** *Let $\mathbf{x}(0) = \mathbf{x}'(0) = \hat{\mathbf{i}}$, $\mathbf{y}(0) = \mathbf{0}_{n\times n}$ and let $\mathbf{y}'(0)$ be all zero except $\mathbf{y}'_{i,j}(0) = 1$, such that $\mathbf{x}(1) = \hat{\mathbf{i}}$ and $\mathbf{x}'(1) = \hat{\mathbf{j}}$. Let $\mathbf{x}(t+1)$ and $\mathbf{x}'(t+1)$ be the load vectors obtained from applying $C$ for $t$ rounds on $(\mathbf{x}(1), \mathbf{y}(0))$ and $(\mathbf{x}'(1), \mathbf{y}'(0))$, respectively. Then the* contribution *of the edge $(i, j)$ on a node $k$ after $t$ rounds is defined as*

$$\mathcal{C}^C_{k,i\to j}(t) = \mathbf{x}_k(t) - \mathbf{x}'_k(t) \ .$$

In [ABEK15], the contributions are expressed based on a sequence of matrices $Q(t)$ defined below, whose role in error propagation is similar to that of the diffusion matrix in FOS. In the corresponding proofs from [ABEK15], we first gave an upper bound in terms of the contributions $\mathcal{C}^C_{k,i\to j}(t)$. Then, in order to obtain more concrete bounds, we estimated and upper bounded $\mathcal{C}^C_{k,i\to j}(t)$.

$$Q(t) = \begin{cases} \mathbf{I} & \text{if } t = 0 \\ \beta \cdot M & \text{if } t = 1 \\ \beta \cdot M\, Q(t-1) + (1-\beta) \cdot Q(t-2) & \text{if } t \geq 2 \end{cases}$$

We now state the bounds on the deviation between continuous SOS and its rounded version from [ABEK15]. Note that the authors of [EMS06] show the following similar bound on the deviation.

$$\left\| x^{D(\mathrm{SOS})}(t) - x^{\mathrm{SOS}}(t) \right\|_2 = \mathrm{O}(d\sqrt{n s_{\max}}/(1-\lambda))$$

Note that the bound on the deviation of FOS, $\mathrm{O}\left(d\sqrt{s_{\max}\log n/(1-\lambda)}\right)$, is smaller.

**Lemma 32.** *Consider a discrete SOS process $D = D(\mathrm{SOS})$ with optimal $\beta$ and a rounding scheme that rounds a fractional value to either its floor or its ceiling. Then for arbitrary $t \geq 0$ we have*

$$\left| x^D_k(t) - x^{\mathrm{SOS}}_k(t) \right| = \mathrm{O}(d\sqrt{n s_{\max}}/(1-\lambda)) \ .$$

## 10.4 Randomized Second Order Scheme

In [ABEK15] we argued that the proof of Lemma 29 holds for the more general definitions of linearity and contributions for SOS. The following lemma was used to show the next theorem similarly to FOS.

**Lemma 33.** *In the setting of Section 10.3 for an arbitrary round $t$ we have with high probability*

$$\left| X^R_k(t) - x^C_k(t) \right| = \mathrm{O}\left(\Upsilon^C(G) \cdot \sqrt{d \log n}\right)$$

In the following theorem from [ABEK15] we bounded the deviation between continuous and discrete SOS using the randomized rounding algorithm from Section 10.2.

**Theorem 34.** *Let $R = R(\mathrm{SOS})$ be a discrete SOS process with optimal $\beta$ obtained using our randomized rounding algorithm. Then*

*(1)* $\Upsilon^{\mathrm{SOS}}(G) = \mathrm{O}\left(\sqrt{d} \cdot \log s_{\max}/(1-\lambda)^{3/4}\right)$ .

*(2) The deviation of $R$ from the continuous SOS in an arbitrary round $t$ is with high probability*

$$\left|x_k^R(t) - x_k^{\mathrm{SOS}}(t)\right| = \mathrm{O}\left(\frac{d \cdot \log s_{\max} \cdot \sqrt{\log n}}{(1-\lambda)^{3/4}}\right) .$$

## 10.5 Negative Load in Second Order Schemes

In second order schemes, it might happen that nodes do not have enough load to satisfy all their neighbors' demand. We refer to this situation as *negative load*. Naturally, one might ask by how much a node's load may become negative. In [ABEK15], we studied the minimum amount of load that nodes need in order to prevent this event. In the following we state a bound on the minimum load of every node that holds during the whole balancing process. More precisely, if every processor has such a minimum load at the beginning of the balancing process, there will be no processor with negative load. Hence, these bounds can also be regarded as bounds on the minimum load of every processor that suffices to avoid negative load.

Let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ be the balanced load vector. We define $\Delta(t) = \|x(t) - \bar{x}\|_\infty$ and $\Phi(t) = \|x(t) - \bar{x}\|_2$, where $\|\cdot\|$ is the norm operator. Then the following observation estimates the load at the end of every step.

**Observation 35.** *In continuous SOS with $\beta = \beta_{opt}$ we have*

$$x(t) \geq -\sqrt{n} \cdot \Delta(0) .$$

In [ABEK15], we noted that the load during a single balancing step can be lower than the bound given in Observation 35, since in this observation we only consider snapshots of the network at the end of each round. It might be possible that a node has to send more load items to some of its neighbors than it has at the beginning of round $t$, but still its load remains positive at the end of round $t$. This can happen if it also receives many load items from other neighbors in round $t$. To study the negative load issue it is helpful to divide every round into two distinct steps, where in the first step all nodes send out their outgoing flows. In the second step, the nodes receive incoming flows sent by their neighbors in the first step. At the end of the first step, all the outgoing flows are sent out but no incoming flow is yet received. To prevent negative load the load of every node has to be non-negative at this point. In [ABEK15], we call this state the *transient state* and use $\breve{x}_i(t)$ to denote the load in the transient state. Note that we always have $\breve{x}_i(t) \leq x_i(t)$ and $\breve{x}_i(t) \leq x_i(t+1)$. The following lemma provides a lower bound on $\breve{x}_i(t)$.

**Lemma 36.** *In a continuous SOS process with optimal $\beta = \beta_{opt}$ we have*

$$\breve{x}_i(t) \geq -\operatorname{O}\left(\sqrt{n} \cdot \Delta(0)/\sqrt{1-\lambda}\right) .$$

The following result from [ABEK15] shows that the asymptotic lower bound obtained in Observation 35 also holds for the randomized discrete second-order process $R = R(SOS)$ when $s_{\max}$ is polynomial in $n$ and $d/(1-\lambda)^{3/4} = \operatorname{O}(n^{0.5-\varepsilon})$ for constant $\varepsilon > 0$. These properties hold, for example, in tori with four or more dimensions, hypercubes, and expanders. In this case we can get the following lower bound.

**Theorem 37.** *In a discrete SOS process $R = R(\mathrm{SOS})$ with $\beta = \beta_{opt}$, $s_{\max}$ polynomial in $n$, and $d/(1-\lambda)^{3/4} = \operatorname{O}(n^{0.5-\varepsilon})$ for some $\varepsilon > 0$, we have*

$$\breve{x}_i^R(t) \geq -\operatorname{O}\left(\frac{\sqrt{n} \cdot \Delta(0) + d^2}{\sqrt{1-\lambda}}\right) .$$

# 11 Simulation Results

In this chapter we present empirical results for several balancing algorithms. We simulated discrete versions of both, first order and second order balancing schemes, where we use *randomized rounding* as described in Section 10.2 for the discretization. We also give a formal definition of the randomized rounding algorithm in Algorithm 11.1. Our main goal is to see under which circumstances SOS outperforms FOS.

```
Algorithm RandomizedRounding(continuous scheme $C$, load vector $x$)
    /* start with the continuous flow */
    let $\hat{Y}(t) \leftarrow C(x^D(t))$;
    at each node $i$ do
        let $r \leftarrow 0$;
        /* send out integral flows */
        for each edge $e = (i,j)$ where $\hat{Y}_{i,j}(t) > 0$ do
            /* $\hat{Y}_{i,j}(t)$ is the load that would be sent over edge $e$
               by the continuous process $C$ */
            send $\lfloor \hat{Y}_{i,j}(t) \rfloor$ tokens to node $j$;
            /* Recall that $\{a\} = a - \lfloor a \rfloor$ for any $a \in \mathbb{R}$ */
            $r \leftarrow r + \{\hat{Y}_{i,j}(t)\}$;        /* collect excess load */
        /* send out excess load randomly */
        for excess token 1 to $\lceil r \rceil$ do
            with probability $r/\lceil r \rceil$ do
                for each edge $e = (i,j)$ let $p_j \leftarrow \{\hat{Y}_{i,j}(t)\}/r$;
                sample neighbor $k$ according to probabilities $p_j$;
                send token to node $k$;
            else
                the token remains at node $i$;
```

**Algorithm 11.1:** The randomized rounding algorithm generates discrete flows from continuous diffusion schemes.

| Graph | Size | Parameter $\beta$ |
|---|---|---|
| Two-Dimensional Torus | $n = 1000 \times 1000$ | 1.9920836447 |
| Two-Dimensional Torus | $n = 100 \times 100$ | 1.9235874877 |
| Random Graph (CM) | $n = 10^6$, $d = \lfloor \log_2 n \rfloor$ | 1.0651965147 |
| Random Geometric Graph | $n = 10^4$, $r = \sqrt[4]{\log n}$ | 1.9554636334 |
| Hypercube | $n = 2^{20}$ | 1.4026054847 |

**Table 11.1:** graph classes and parameters used in the simulation

We consider different networks which are based on various graph classes. A complete list of all graph types and parameters used for the simulation can be obtained from Table 11.1. Our simulation tool is highly modularized and supports various load balancing schemes and rounding procedures. It can be used to simulate the load balancing process using multiple threads on a shared-memory machine. To fully utilize the capability of modern CPUs we used OpenMP to generate code that performs suitable instructions in parallel. The simulation was implemented using the C$^{++}$ programming language. Our tests were conducted on an Intel Core i7 machine with 4 cores and 8 GB system memory. Some simulations of denser graph classes were also run on machines equipped with 64 AMD Opteron cores and 1 TB memory.

If not stated otherwise, we initialized our system by assigning a load of $1000 \cdot n$ load tokens to a fixed node $v_0$, where $n$ is the number of nodes of the network, and the load of all other nodes was set to zero. Our data plotted in Figure 11.2, however, indicate that the amount of initial load does only have limited impact on the behavior of the simulation, especially once the system has converged.

We use the following metrics to measure the quality of the load distribution.

1. **Maximal local load difference.** This is the maximum load difference between the nodes connected by an edge. That is, the maximum local load difference for given load vectors $x(t)$ in a round $t$ is defined as

$$\phi_{\text{local}}(x(t)) = \max_{\{u,v\} \in E} \{|x_u(t) - x_v(t)|\} .$$

2. **Maximum load.** This is the maximum load of any node minus the average load $\overline{x}$.

$$\phi_{\text{global}}(x(t)) = \Delta(t) = \max_{v \in V}\{x_v(t)\} - \overline{x}$$

3. **Potential based on 2-norm.** We compute the value of the potential function $\phi_t$ proposed by Muthukrishnan et al. [MGS98] which is defined as

$$\phi_t = \phi(x(t)) = \sum_{v \in V} (x_v(t) - \overline{x})^2$$

In our plots we divided this potential by $n$.

4. **Impact of eigenvectors on load.** We initially compute the eigenvectors of the diffusion matrix and solve in each round $t$ the the linear system $V \cdot a = x(t)$, which is defined over the orthonormal matrix of $n$ eigenvectors $V$ and the load vector $x(t)$. We then identify the *leading* eigenvector, that is, the eigenvector with the largest $|a_i|$. The coefficients $a_i$ for $i = 2, \ldots, n$ describe together with the eigenvectors the load imbalance completely [GV61]. Observe that the coefficient $a_i$ in round $t$ multiplied with the corresponding eigenvalue $\mu_i$ yields the coefficient in the following round $t + 1$. Therefore, the largest coefficient governs the convergence rate in that step.
5. **Remaining imbalance.** This is the remaining imbalance of the converged system (see [ES10]), that is, the number of tokens above average once this number starts to fluctuate and does not visibly improve any more. This imbalance does not occur in continuous systems and is due to the applied rounding in discrete systems.

In this first section we focus on the torus. For results w.r.t. other graph classes see Section 11.2.

## 11.1 Results for the Torus

Our main results are shown in Figure 11.1, where we plotted the simulation results using the second order scheme with randomized rounding in a two-dimensional torus consisting of $1000 \times 1000$ nodes and an average load of 1000. As in all following plots, the $x$-axis represents the number of rounds. The plot shows the maximum load minus the average load, the maximum local load difference, and the potential function $\phi_t$ on the $y$-axis. As a comparison, a simulation run using only first order scheme is shown as well.

It is known that the second order scheme is faster than the first order scheme w.r.t. the convergence time of the load balancing system in graphs with a suitable eigenvalue gap. However, our simulations indicate that for SOS the remaining maximal load difference does not drop below a certain threshold. Therefore, we implemented the following approach to decrease the load differences even further. First we perform a number of steps using the fast second order scheme. Then, every node synchronously switches to first order scheme. We considered two different scenarios. In the first case we switched to FOS early after 2500 SOS steps. This number of steps corresponds roughly to the end of a phase of exponential decay in the potential function. In the second case we switched to FOS rather late at 3000 steps, allowing the system to run for a few hundred additional steps using the second order scheme.

In both cases we observed a significant drop in both, the local and the global load differences. That is, the values for the load differences do not drop below 10 when using SOS. Once the simulation is switched to FOS, the maximum local load difference converges to a value of 4 and the maximum load minus the

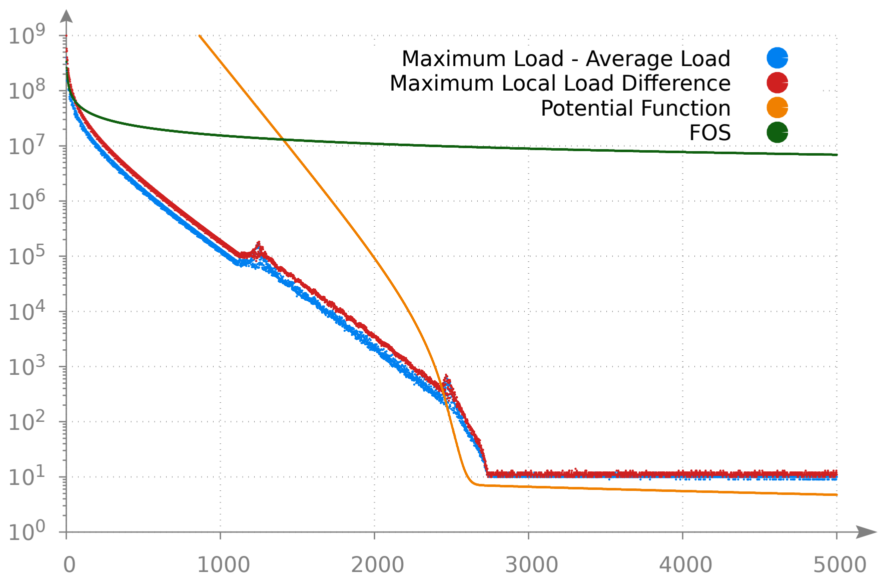


**Figure 11.1:** The maximum load minus the average load is plotted in blue, the maximum local load difference in red and the potential function $\phi_t$ in yellow, using SOS on a two-dimensional torus size $1000 \times 1000$. As a comparison, the green line shows the maximum load minus the average load using FOS.

average load drops to 7. This is shown in Figure 11.4, and a direct comparison is shown in Figure 11.5.

In the left plot in Figure 11.4 we furthermore observe that the load differences continue to diminish for about 200 steps (during steps 2500 to 2700) when we switch to FOS after 2500 steps. When we switch to FOS after 3000 steps (right plot in Figure 11.4) a drop can still be observed, however, the resulting load differences remain at a low level. To explain this behavior of the load balancing procedure we analyzed the impact of the eigenvectors of the diffusion matrix on the load balancing process. Recall that the diffusion matrix $M = (M_{ij})$ is defined as

$$M_{ij} = \begin{cases} \alpha_{ij} & \text{if } i \neq j \\ 1 - \sum_{i \neq j} \alpha_{ij} & \text{if } i = j \end{cases}$$

with $\alpha_{ij} = 1/(\max\{\deg(i), \deg(j)\} + 1)$ if node $i$ is adjacent to node $j$ and 0 otherwise.

We used the LAPACK library [ABB+99] to compute the eigenvalues and corresponding eigenvectors of $M$. The same library was then used to solve the set of linear systems

$$V \cdot \mathfrak{a} = W$$

for a matrix of coefficients $\mathfrak{a}$, where $V$ denotes a matrix of eigenvectors of $M$ and $W = (x(t))$ consists of row vectors $x(t)$ as defined in Section 9.1 containing the loads of the system at every round $t$. The resulting coefficients in $\mathfrak{a}$ give the impact of the corresponding eigenvectors in each round on the load. The results are shown for the torus of size $100 \times 100$ in the two plots of Figure 11.7. The first plot shows the maximum of these coefficients. In the simulation run corresponding to this plot we observed that starting roughly after 100 rounds this leading eigenvector corresponds to $\mathfrak{a}_4$ up until roughly round 700. After that time there is no clear leading eigenvector. This can be observed from the right plot in the same figure, where the currently leading coefficient is plotted for each round.

It seems reasonable to switch from SOS to FOS once the impact of the leading eigenvector drops below some threshold. This information, however, requires a global view on the load balancing network and therefore cannot be used in a distributed approach. In real-world applications also the trade-off between a remaining imbalance and the time required to balance the loads must be considered. We therefore investigate the effect of the time step when switching from SOS to FOS.

In Figure 11.8 we plotted the maximum load minus the average load for second order scheme and for an adaptive approach where we switched to FOS after a number of SOS rounds. The impact of the leading eigenvector (and the loss thereof) explains the data shown in Figure 11.8. Our data indicate that once the impact of the leading eigenvector drops below a certain threshold in a round $R$, there is no difference in the behavior of the system when switching to FOS in some consecutive round $r \geq R$. Independently of the round $R$, however, we observe a significant drop in the maximum load.

Note that the maximum local load difference seems to be a good indicator for switching from SOS to FOS. Furthermore this local property is also available in a distributed system with only limited global knowledge.

In Figure 11.1 we also observe strong discontinuities of the local and global maximum load differences which occur approximately every 1200 to 1300 steps. To explain these discontinuities we visualized the load balancing process on the two-dimensional torus in Figure 11.9 as follows. We rendered a raster graphic of size $1000 \times 1000$ pixels per round. In the graphic each pixel represents a node of the torus such that neighboring pixels are connected in the network and border-pixels are connected in a periodic manner. We now set the pixels' colors to correspond to the nodes' loads, that is, a pixel is shaded bright if its load is close to the average load and dark otherwise. In the visualization shown in Figure 11.9 the initial load is placed at the node with ID 0, which corresponds to the top-left pixel. Since the border-pixels *wrap around*, the loads spread in circles from all four corners, forming the *wavefronts* in the graphic. Our visualizations now indicate that the discontinuities in the local load differences and the maximum load occur whenever these wavefronts collapse at the center of the graphic, that is, when the center node gets load for the first time. This is a consequence of the second order scheme since nodes continue to push loads towards the center pixel, even though this pixel may already have a load above average. Note that these discontinuities also occur in the idealized scheme and for smaller tori, see Figure 11.6 and Figure 11.8, respectively.

We furthermore rendered a video of the load balancing process (available online, see [ABEK14b]) which shows the behavior of the system in an intuitive way and thus helps understanding these discontinuities. Further visualizations in Figure 11.11 show the impact of the first order scheme. That is, after applying FOS steps the rendered image becomes more *smooth*, in contrast to the SOS steps where our visualization shows a significant amount of noise.

To gain further insights we also implemented a simulation of the idealized load balancing procedure where loads can be split up in arbitrary small portions and any real fraction of load can be transmitted. This simulation is based on double precision floating point variables that represent the current load at a node. Therefore, a quantification takes place which introduces an error. However, we observed that in our setup the total error over all loads is small and thus can be neglected. A comparison of the idealized and discrete processes can be found in Figure 11.6.

## 11.2 Further Networks

For random regular graphs constructed using the configuration model [Wor99] and the hypercube we observe only a limited improvement of SOS compared to FOS, see Figure 11.12 and Figure 11.13, respectively. That is, the number of steps required to balance the loads up to some additive constant is only slightly larger when using FOS instead of SOS. For random graphs the remaining imbalance is the same for both FOS and SOS. For the hypercube our results indicate that the remaining imbalance using FOS is by one smaller than in the case of the SOS process. Hence, our data only show a negligible difference between FOS and SOS in these graphs. This can be related to the second largest eigenvalue of the diffusion matrix, which is $(2 + o(1))/\sqrt{d}$ for random graphs and $1 - 2/(\log n + 1)$ for hypercubes (compared to approximately $1 - \pi^2/n$ for the torus) [CDS80]. Note that the spectral gap is also reflected in the corresponding values for $\beta$ in Table 11.1.

The random geometric graphs were generated by assigning each node a coordinate pair in the range $[0, \sqrt{n}]^2$ uniformly at random and connecting nodes $v_i$ and $v_j$ if and only if $d(v_i, v_j) \leq \sqrt[4]{\log n}$, where $d$ denotes the euclidean distance. Remaining small isolated components were connected to the closest neighbor in the largest component of the graph. Even though we observe a less pronounced potential drop in random geometric graphs, the behavior of FOS and SOS in these graphs is very similar to the behavior in the torus graphs, see Figure 11.14 and Figure 11.15.

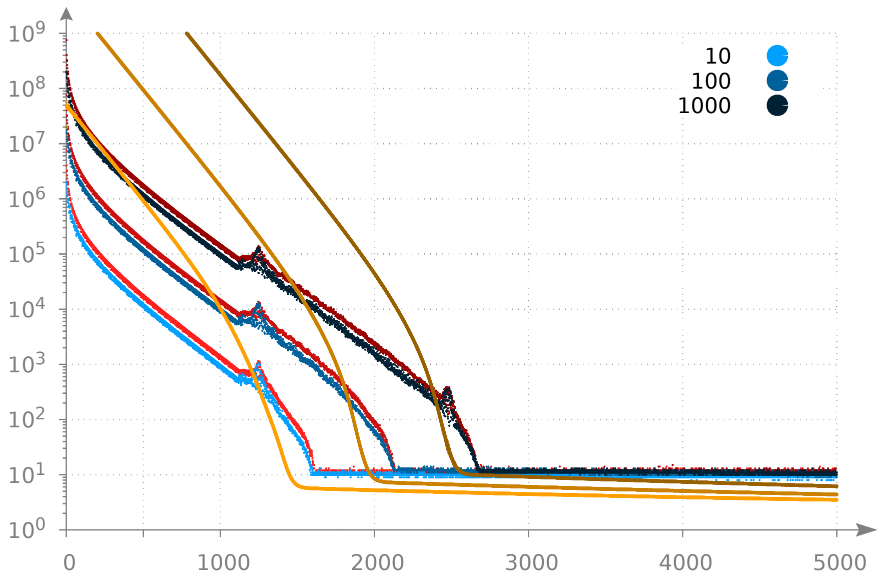


**Figure 11.2:** The plot shows the maximum load minus the average load, the maximum local load difference and the potential function on a two-dimensional torus of size $1000 \times 1000$. Three different initial loads were used with average loads of 10, 100, and 1000, colored from light to dark.

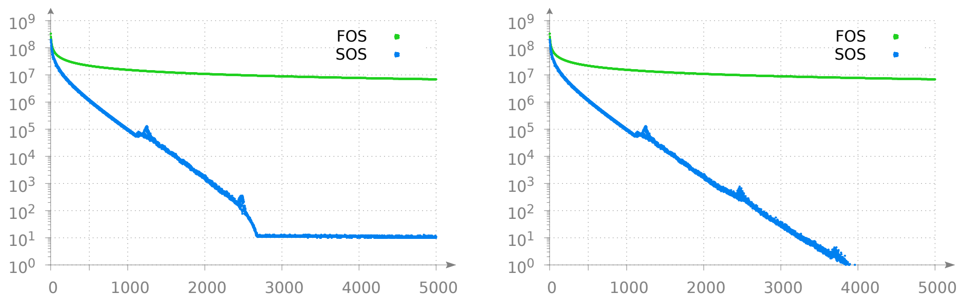


**Figure 11.3:** a comparison of the maximum load minus the average load using SOS (blue) and FOS (green) on a two-dimensional torus of size $1000 \times 1000$. The first plot shows discrete loads and randomized rounding, the second plot shows an idealized scheme.

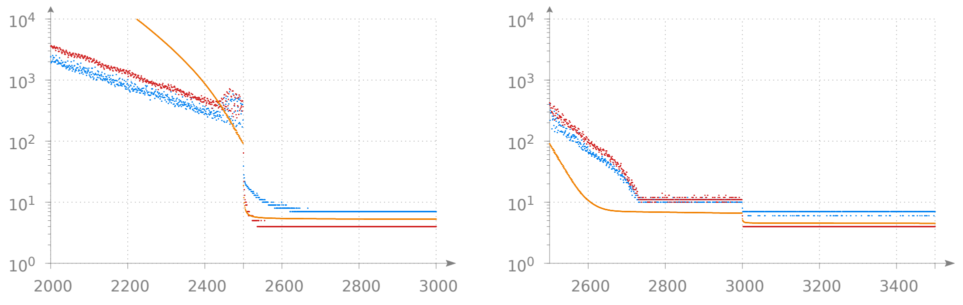


**Figure 11.4:** The plots show the maximal local difference in red, the maximal load minus the average load in blue, and the potential function $\phi_t$ in yellow. The simulation switches from second order scheme to first order scheme in the left and the right plot after 2500 and 3000 rounds, respectively.

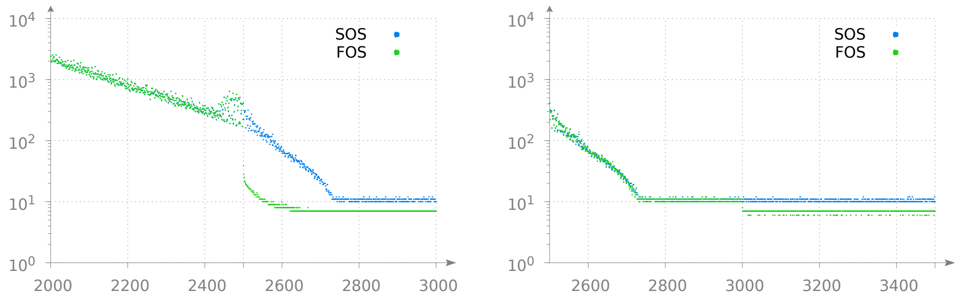


**Figure 11.5:** The plots show a direct comparison of the same data presented in Figure 11.4. The blue data points show the maximal load minus the average load using only a SOS approach while the green data points show the maximal load minus the average load when switching to FOS. Again, the switch has been conducted after 2500 steps in the left and 3000 steps in the right plot.

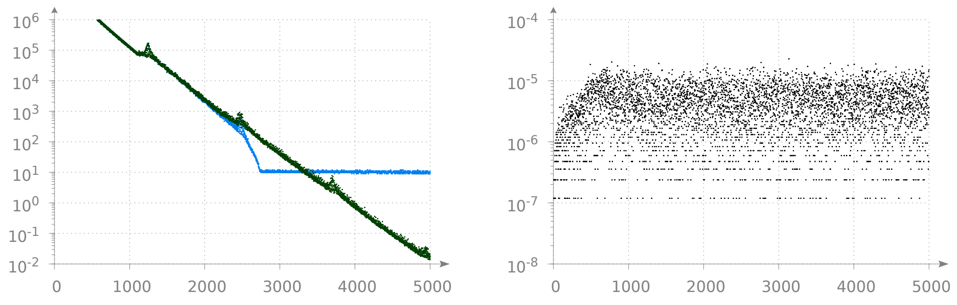


**Figure 11.6:** a comparison of the idealized second order scheme in green with a SOS using randomized rounding in blue. The idealized version is based on IEEE754 double precision floating point values as loads. The data points show the maximum load of the system minus the average load. The right plot shows the absolute value of the total load in the system at round $t$ minus the initial total load, that is, the absolute error.

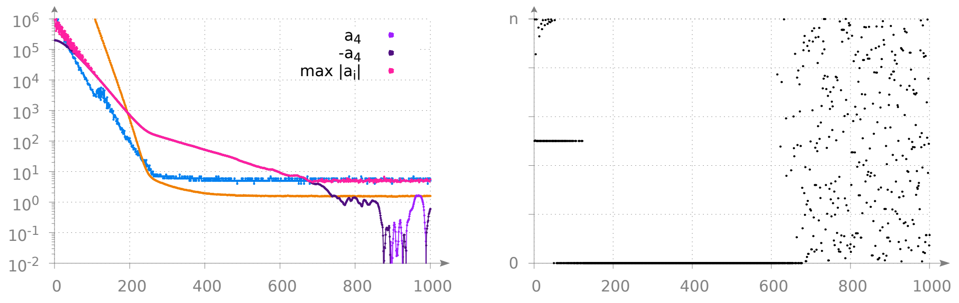


**Figure 11.7:** The left plot shows the impact of eigenvectors on the load on a two-dimensional torus of size 100 × 100. The maximum over all coefficients, $\max_i\{|\mathfrak{a}_i|\}$, is shown along with $\mathfrak{a}_4$. In the right plot the currently leading coefficient is shown, that is, a black point indicates that in the given round ($x$-axis) the corresponding eigenvector ($y$-axis) has maximal impact.

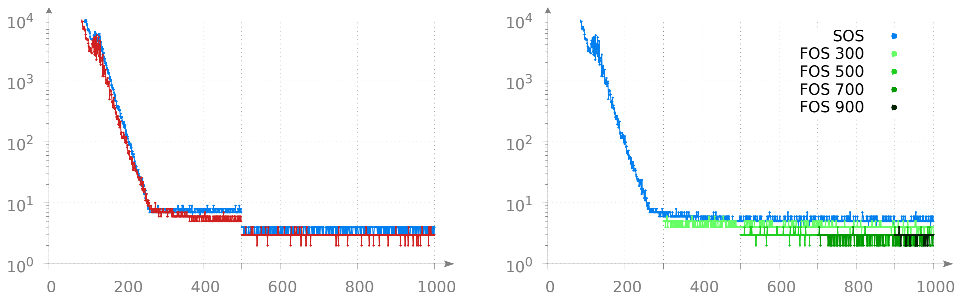


**Figure 11.8:** a plot showing the effect of switching from SOS to FOS on a two-dimensional torus of size 100 × 100. The left plot shows the maximum load minus the average load in blue and the maximal local load difference in red. After 500 SOS rounds the process switches to a FOS approach. In the right plot various time steps to switch from SOS to SOS are used. All data points show the current maximum load minus the average load.

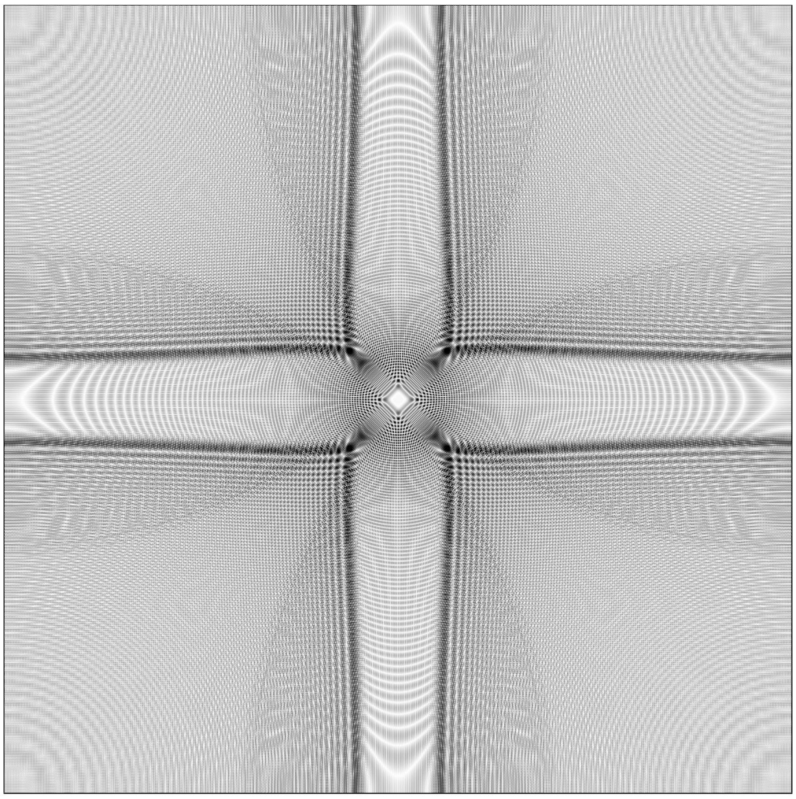


**Figure 11.9:** a visualization of the load balancing network (a two-dimensional torus of size $1000 \times 1000$) after 1100 steps. Each pixel corresponds to one node which has edges to its 4-neighborhood and is shaded such that a light pixel has a load close to the average load and a dark pixel a load close to either the maximum or minimum load of the system. Further time steps are rendered in Figure 11.10.

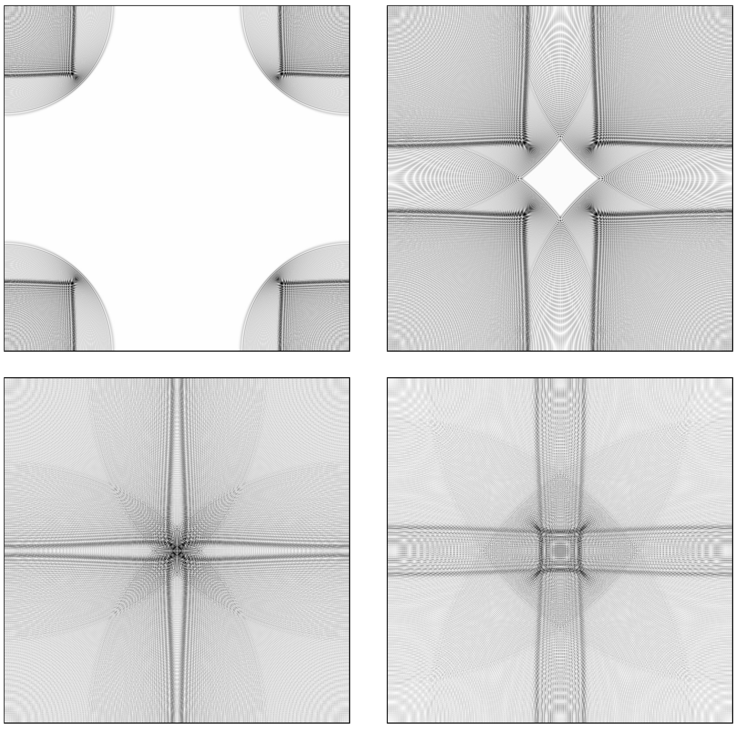


**Figure 11.10:** The figure shows the same visualization as Figure 11.9, rendered after 500, 1000, 1200, and 1400 steps. The network is modeled as a two-dimensional torus. Each pixel corresponds to one node which has edges to its 4-neighborhood. All pixels are shaded in an adaptive way, that is, a light gray or white pixel indicates a load close to the average load and a dark gray or black pixel indicates a load close to either the maximum or minimum load of the system.

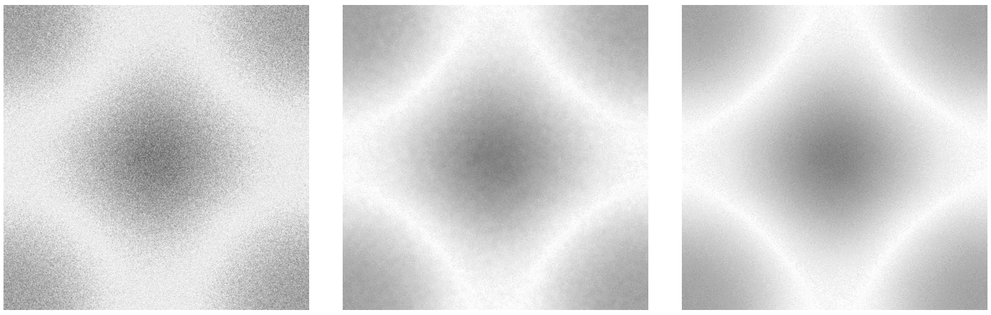


**Figure 11.11:** Above figures show the same load balancing network as Figure 11.9. A pixel colored white indicates a node with optimal load, a pixel colored black corresponds to a load that is more than 10 units away from the optimal value. Observe that in none of the above images such a load (which exceeds the average load by more than 10 tokens) occurs. In the center region of the left image there are several pixels which have load at least 9, whereas in the right image the maximum load exceeds the average load by at most 7. The first visualization has been rendered after 3000 SOS steps. The second image and the third image show the same network after additional 100 and 1000 FOS steps, respectively.

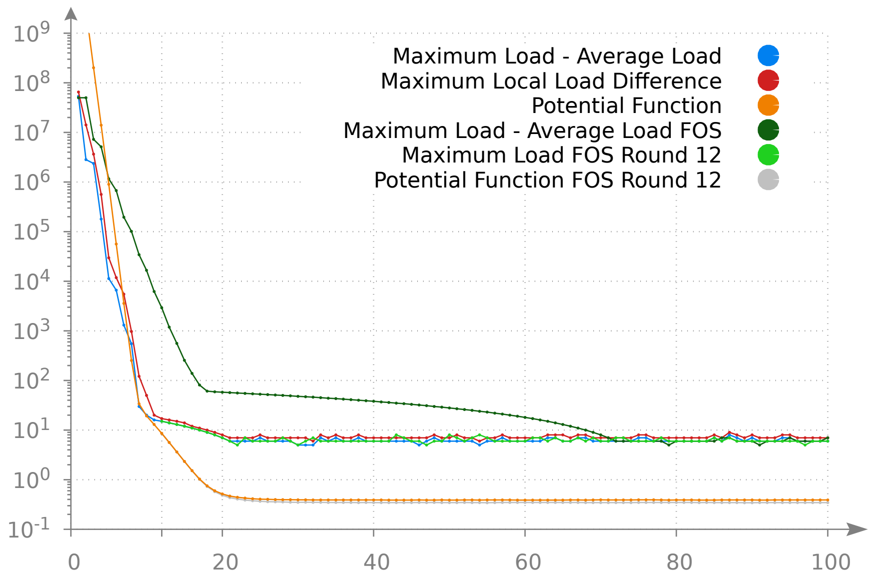


**Figure 11.12:** load balancing simulation on a random graph in the configuration model of size $10^6$ nodes with $d = 19$

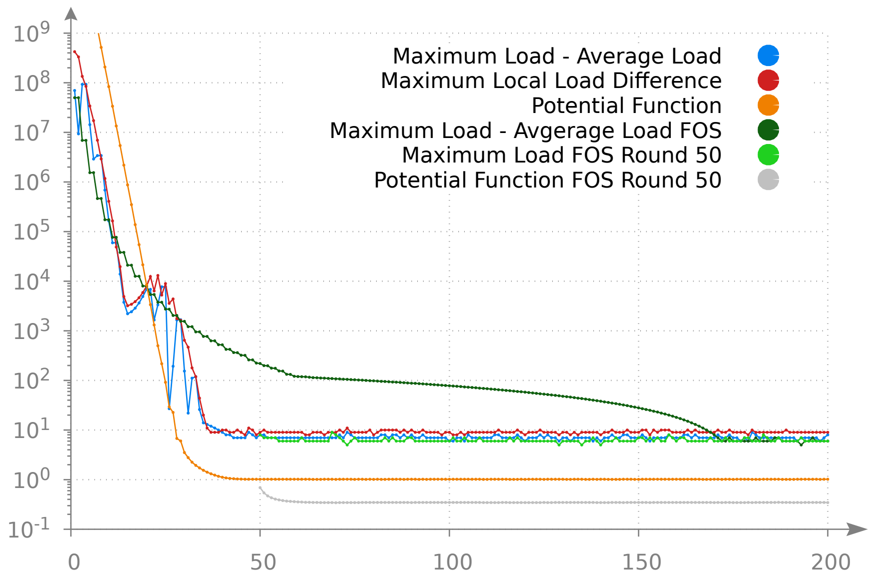


**Figure 11.13:** load balancing simulation on a hypercube with $n = 2^{20}$ nodes. The green data points show the effect of switching to FOS after 32 steps.

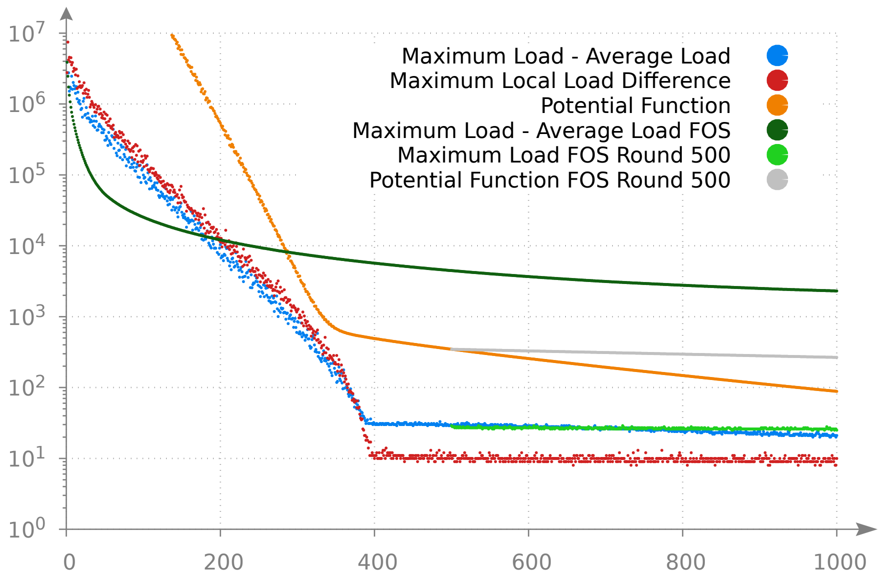


**Figure 11.14:** load balancing simulation on a random geometric graph with 10.000 nodes in $[0, \sqrt{n}]^2$ with connectivity radius $\sqrt{\log n}$.

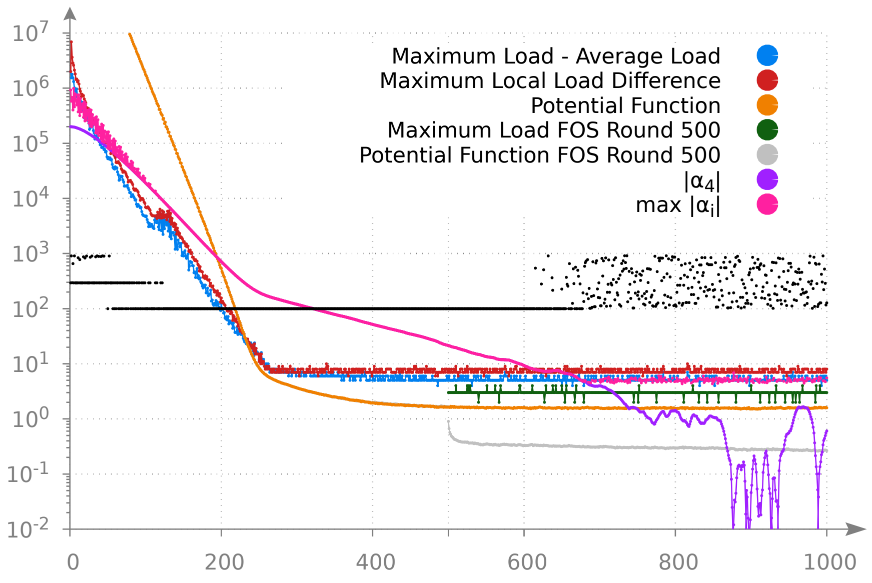


**Figure 11.15:** load balancing simulation on a two-dimensional torus of size $100 \times 100$. The purple line shows the maximum coefficient $\max\{|\alpha_i|\}$ for the impact of the eigenvectors on the load. This coefficient is $\alpha_4$, starting approximately in round 100 and up to approximately round 700. The black dots also shown in this plot in the range $[10^2, 10^3]$ represent the leading coefficient, where $\alpha_1$ is plotted with a value of $10^2$ and $\alpha_n$ is plotted with a value of $10^3$, with a linear scale between them. Observe that after approximately 700 rounds no single eigenvector can be identified that has a leading impact on the load.

# Part III

# The Deterministic Majority Voting Process

Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale: *Brief Announcement: On the Voting Time of the Deterministic Majority Process.* In *Proceedings of the 29th International Symposium on Distributed Computing (DISC)*, 2015.

Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale: *On the Voting Time of the Deterministic Majority Process.* In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2016. DOI: 10.4230/LIPIcs.MFCS.2016.55.

# 12 Introduction

Distributed voting is a fundamental problem in distributed computing. We are given a network of players modeled as a graph. Each player in the network starts with one initial opinion out of a set of possible opinions. Then the voting process runs either synchronously in discrete rounds or asynchronously according to some activation mechanism. During these rounds in the synchronous case, or upon activation in the asynchronous case, the players are allowed to communicate with their direct neighbors in the network with the main goal to eventually agree on one of the initial opinions. If all nodes agree on one opinion, we say this opinion *wins* and the process *converges*. Usually, voting algorithms are required to be simple, fault-tolerant, and easy to implement [HP01, Joh89].

In this part, we study the *deterministic binary majority process* which is defined as follows. We are given a graph $G = (V, E)$ where each node has one of two possible opinions. The process runs synchronously in discrete rounds where each node in every round computes and adopts the majority opinion among all of its neighbors. It is known that this process always converges to a two-periodic state. The *convergence time* of a given graph for a given initial opinion assignment is the time required until this two-periodic state is reached. In this work we improve the bounds on the convergence time for given initial opinions and then we analyze the *voting time* of the process, which is the maximum convergence time over all possible initial opinion assignments.

In distributed computing, various variants of the majority process are used in fault-tolerant distributed consensus algorithms. In the analysis of structures of large networks, the deterministic binary majority process has widespread applications in the study of so-called *influence networks* [FKW13]. Early applications can be found in distributed databases [Gif79]. Further fields include sensor networks [BTV09], the analysis of opinions in social networks [MT14], social behavior in game theory [DP94], chemical reaction networks [Dot14], neural and automata networks [GM90], and cells' behavior in biology

[CC12]. Variants of the deterministic binary majority process have been used in the area of distributed community detection [RAK07, KPS13, CG10]. In this context, the proposed community detection protocols exhibit a convergence time which can be bounded by the voting time of the deterministic binary majority process.

Among its many probabilistic variants that have been previously considered, plenty of work concerns *randomized voting* where in each step every node is allowed to contact a random sample of its neighbors and updates its current opinion according to the majority opinion in that sample [AF02, BMPS04, CEOR13, DW83, HP01, HL75, LN07, Lig85, Mal14, Oli12].

In an algorithmic game theoretic setting, the deterministic binary majority process can be seen as the simplest *discrete preference games* [CKO13]. In this game theoretic perspective, the existence of *monopolies* has been investigated [ACF+15]. A monopoly in a graph is a set of nodes which start with the same opinion and cause all other nodes to eventually adopt this opinion. In the distributed computing area, a lot of research has been done to find small monopolies, see for example [Pel02]. It has also been shown that there exist families of graphs with constant-size monopolies [Ber01]. More recently, classes of graphs which do not have small monopolies have been investigated [Pel14].

Many of these results relate to the voting time of the deterministic binary majority process. It was proven independently by Goles and Olivos [GO80], and Poljak and Sůra [PS83] with the same potential function argument that the deterministic binary majority process always converges to a two-periodic state. They later (independently) refined and generalized the potential function argument in several directions [Gol89, GO88, GFP85, PT86]. Their proof was popularized in the *Puzzled* columns of Communications of the ACM [Win08a, Win08b]. Recently, the same problem has been studied on infinite graphs w.r.t. a given probability distribution on the initial opinion assignments [BCO+14]. In [TT15], the authors provide a bound on the number of times a node in a given bounded-degree graph changes its opinion. Both [BCO+14] and [TT15] also investigate the probability that in the two-periodic state all nodes hold the same opinion.

As for the maximum time it takes for the process to converge over all initial opinion assignments, Frischknecht et al. [FKW13] note that the potential argument by Goles et al. [GO80, PS83, Win08b] can be used to prove an $O(|E|)$ upper bound. They furthermore show that this upper bound is tight in general, by designing a class of graphs in which the deterministic binary majority process takes at least $\Omega(|V|^2)$ rounds to converge from a given initial opinion assignment. This construction has later been extended to prove lower bounds for weighted and multi-edges graphs by Keller et al. [KPW14].

Once the process converges to the two-periodic state, each node stays either with its own opinion or changes its opinion in every round. A lot of attention has been given to the opinions to which the deterministic binary majority process converges. However, regarding the voting time, besides the $O(|E|)$

upper bound that follows from the result by Goles et al. [GO80, PS83, Win08b], no further upper bound on the voting time that holds for any initial opinion assignment has been proved. Still, one can observe that in many graphs the voting time is much smaller than $\mathrm{O}(|E|)$. For example, the voting time of the complete graph is one.

We show that for the deterministic binary majority process the question whether the voting time is greater than a given number is NP-hard. While for many generalizations of the deterministic binary majority process many decision problems are known to be NP-hard, at the best of our knowledge this is the first NP-hardness proof that does not require any additional mechanisms besides the bare majority rule of the deterministic binary majority process. However, it is possible to obtain upper bounds on the voting time which can be computed in linear time. A module of a graph is a subset of vertices $S$ such that for each pair of nodes $u, v \in S$ it holds that $N(u) \setminus S = N(v) \setminus S$, where $N(u)$ denotes the set of neighbors of a node $u$. By carefully exploiting the structure of the potential function by Goles et al. we leverage the particular behavior that certain modules, which we call *families*, exhibit and prove that the voting time of a graph can be bounded by the voting time of a smaller graph that can be constructed in linear time by contracting suitable vertices.

We obtain a new upper bound that asymptotically improves the previous $\mathrm{O}(|E|)$ bound on graph classes which are characterized by a high number of modules that are either cliques or independent sets. An example for such graphs is the Turán graph $T(n, r)$, formed by partitioning a set of $n$ vertices into $r$ subsets of (almost) equal sizes and connecting two vertices by an edge whenever they belong to different subsets. For the convergence time of the Turán graph $T(n, r)$ we obtain an $\mathrm{O}(r^2)$ bound, compared to the previously best known bound of $\mathrm{O}(n^2)$. Also, for the convergence time of full $d$-ary trees we get an $\mathrm{O}(|V|/d)$ bound, compared to $\mathrm{O}(|V|)$ originating from the $\mathrm{O}(|E|)$ bounds. Further examples include the clique and the star graph, for which our bound gives a constant $\mathrm{O}(1)$ convergence time. Our bound relies on a well-known graph contraction technique based on identifying *equivalent nodes*. This technique is used in other related disciplines as well, including parallel and distributed computing. See, for example, the notion of *identical nodes* in the work by Sarıyüce et al. [SSKÇ13].

## 12.1 Preliminaries

We are given a graph $G = (V, E)$ and an initial opinion assignment defined as follows.

**Definition 8.** *An opinion assignment $f_t$ in round $t \geq 0$ is a function $f_t : V \to \{0, 1\}$ which assigns for each $v \in V$ one out of two possible opinions. We will also denote opinion $1$ as* white *and opinion $0$ as* black*. The opinion assignment at time $t = 0$ is called* initial opinion assignment*.*

The deterministic binary majority process can be defined as follows. Let $v$ be an arbitrary but fixed vertex and $N(v)$ the set of neighbors of $v$. To compute $f_{t+1}(v)$ the node $v$ computes the majority opinion of all of its neighbors in $N(v)$. In the case of a tie the node behaves *lazily*, that is, $v$ stays with its own opinion. Otherwise, there is a *clear majority* and the node adopts the majority opinion. This leads to the following definition.

**Definition 9.** *Let $G = (V, E)$ be a graph and let $f_0$ be an initial opinion assignment such that $f_0 : V \to \{0, 1\}$. The deterministic binary majority process is the series of opinion assignments that satisfy the rule*

$$f_{t+1}(v) = \begin{cases} 0 & \text{if } |\{u \in N(v) : f_t(u) = 0\}| > |\{u \in N(v) : f_t(u) = 1\}| \\ 1 & \text{if } |\{u \in N(v) : f_t(u) = 0\}| < |\{u \in N(v) : f_t(u) = 1\}| \\ f_t(v) & \text{otherwise.} \end{cases}$$

Note that the pair $(G, f_0)$ completely determines the behavior of the system according to the majority process. We now define the main object of this work, the voting time.

**Definition 10.** *Given a graph $G = (V, E)$ and any initial opinion assignment $f_0$ on $V$, the* convergence time *$\mathfrak{T}$ of the majority process on $G$ w.r.t. $f_0$ is $\mathfrak{T} = \mathfrak{T}(G, f_0) = \min\{t : \forall v \ f_{t+2}(v) = f_t(v)\}$. The* voting time *of $G$ is defined as $\max\limits_{f_0 \in \{0,1\}^V} \mathfrak{T}(G, f_0)$.*

Observe that $\mathfrak{T}$ is indeed the number of steps until the process converges to a two-periodic state. This holds since the process is completely determined by the current opinion assignment. Thus $f_{t+2}(v) = f_t(v)$ also implies that $f_{t+3}(v) = f_{t+1}(v)$ for all nodes $v$.

In the following we assume without loss of generality that $G$ is connected. For disconnected graphs the deterministic binary majority process runs independently in each connected component. Therefore, the resulting upper bounds on the voting time can be replaced by the maximum over the corresponding bounds in the individual connected components of $G$.

## 12.2 Our Contribution

First we define the *voting time decision problem* VTDP and show in Chapter 13 that it is NP-complete.

**Definition 11** (voting time decision problem VTDP)**.** *For a given graph $G$ and an integer $k$, is there an assignment of initial opinions such that the voting time of $G$ is at least $k$?*

**Theorem 38.** *Given a general simple graph $G$,* VTDP *is NP-complete.*

In Chapter 14 we extend known approaches to derive upper bounds on the voting time for general graphs, which are tight up to a constant of 1. In Section 14.2, we identify the following subsets of nodes that play a crucial role in determining the voting time of the deterministic binary majority process.

**Definition 12.** *A set of nodes $S$ is called a family if and only if for all pairs of nodes $u, v \in S$ we have $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. We say that a family $S$ is* proper *if $|S| > 1$.*

The set of families of a graph forms a partition of the nodes into equivalence classes. Our main contribution is a proof that the voting time of the deterministic binary majority process is bounded by that of a new graph obtained by contracting its families into one or two nodes, as stated in the following theorem.

**Definition 13.** *Given a graph $G = (V, E)$, its asymmetric graph $G^{\Delta} = (V^{\Delta}, E^{\Delta})$ is the subgraph of $G$ induced by the subset $V^{\Delta} \subseteq V$ constructed by contracting every family of odd-degree non-adjacent nodes to one node, and any other proper family to two nodes.*

Let in the following $V_{\text{even}}$ be the set of even-degree vertices in $V$ and, analogously, let $V_{\text{odd}}$ be the set of odd-degree vertices. Based on above definition of $G^{\Delta}$, we give the following bound on the voting time.

**Theorem 39.** *Given any initial opinion assignment on a graph $G = (V, E)$, the voting time of the deterministic binary majority process is at most*

$$1 + \min\left\{ |E^{\Delta}| - \frac{|V^{\Delta}_{odd}|}{2}, \frac{|E^{\Delta}|}{2} + \frac{|V^{\Delta}_{even}|}{4} + \frac{7}{4} \cdot |V^{\Delta}| \right\} .$$

*Furthermore, this bound can be computed in $\mathrm{O}(|E|)$ time.*

As mentioned before, this bound becomes $\mathrm{O}(r^2)$ for the Turán graph $T(n, r)$ and $\mathrm{O}(|V|/d)$ for $d$-ary trees.

Finally, in Chapter 15 we give some insight into further interesting computational properties of the deterministic binary majority process. For example, we disprove a monotonicity of the convergence time w.r.t. the potential function and argue that the voting time is not, at least straightforwardly, bounded by the diameter of the graph.

# 13 NP-Completeness

If it was possible to efficiently compute the voting time, there would have been not much interest in investigating good upper bounds for it. In this section, we show that this is unlikely to be the case. We prove Theorem 38 by reducing 3SAT to the voting time decision problem. Given $\Phi \in$ 3SAT, we construct a graph $G = G(\Phi)$ such that the deterministic binary majority process on $G$ simulates the evaluation of $\Phi$. The graph $G$ consists of $h$ layers where $h = 3 + 4 \cdot n$. The first layer represents an assignment of the variables in $\Phi$, the remaining layers represent $\Phi$ and ensure that the assignment of variables in $\Phi$ is valid. We will show that if $\Phi$ is satisfiable, then there exists an initial assignment of opinions for which the convergence time is exactly $h + 1$. If, however, $\Phi$ is not satisfiable, then any assignment of opinions will result in a convergence time strictly less than $h + 1$.

We now give the formal proof.

## 13.1 Reduction

Let $\Phi \in$ 3SAT be a Boolean formula in 3-conjunctive normal form. Let $n$ be the number of variables of $\Phi$. Let $m$ be the number of clauses of $\Phi$. The Boolean formula is of the form $\Phi = (l_{1,1} \vee l_{1,2} \vee l_{1,3}) \wedge \cdots \wedge (l_{m,1} \vee l_{m,2} \vee l_{m,3})$, where $l_{i,j} \in \{x_1, \overline{x}_1, x_2, \overline{x}_2, \cdots, x_n, \overline{x}_n\}$ is a literal for $1 \leq i \leq m$ and $1 \leq j \leq 3$.

We construct a graph $G$ to simulate the evaluation of $\Phi$ as follows. Let $\ell = 10 \cdot (m + n) + 1$. The graph consists of several layers. On the first layer, we place so-called literal cliques of size $\ell$, and on the layers above we place the gates. In our reduction, we use OR-gates, an AND-gate, and 2/3-gates. Each gate consists of one or several nodes. Additionally, we have two so-called *mega-cliques* $K_{\text{white}}$ and $K_{\text{black}}$ of size $\ell$.

Let $g$ be an arbitrary but fixed gate. We denote a node on a layer below $g$ which does not belong to $g$ but is connected to $g$ as *input node* to $g$. Additionally, we will denote a node that belongs to $g$ and is connected to another gate on a layer above $g$ as *output node* of $g$.

In the following, we assume that opinion 1, white, corresponds to Boolean TRUE and 0, black, corresponds to FALSE. The main idea of the construction is to show that an *activation signal* is transmitted from the bottom up through all layers. If the current assignment of opinions on the literal cliques corresponds to a satisfying assignment of Boolean values to $\Phi$, then the process requires $h+1$ steps. The main purpose of the OR-gates and the AND-gate is to evaluate $\Phi$. The 2/3-gates check whether the opinion assignment to literal nodes is valid. That is, we need to enforce that the corresponding literal nodes for $x_i$ and $\overline{x}_i$ are of opposite colors for every variable $x_i$ of $\Phi$. If either this condition is violated and variables $x_i$ exist for which $x_i = \overline{x}_i$ or the current assignment of opinions on the literal cliques does not corresponds to a satisfying assignment of Boolean values to $\Phi$, the construction enforces that the process stops prematurely after strictly fewer than $h+1$ steps.

## Layer 1: Literal Cliques.

We represent each variable $x_i$ with two cliques, one for $x_i$ and one for $\overline{x}_i$. Each clique has a size of $\ell$ which is defined above. Note that $\ell$ is odd. Additionally, we distinguish three so-called *representative nodes* in each of these cliques. Furthermore, we add two cliques of size $\ell$ to the graph which we call mega-cliques. Intuitively, these mega-cliques represent the Boolean values TRUE and FALSE. We will show that they cannot have the same color in order to achieve a long convergence time. The mega-cliques are used in all other gates.

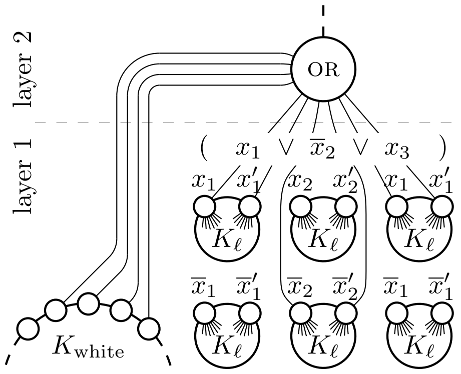

layer 2
layer 1
OR
$( \; x_1 \vee \overline{x}_2 \vee x_3 \; )$
$x_1$ $x_1'$
$\overline{x}_2$ $x_2'$
$x_1$ $x_1'$
$K_\ell$
$\overline{x}_1$ $\overline{x}_1'$
$\overline{x}_2$ $\overline{x}_2'$
$\overline{x}_1$ $\overline{x}_1'$
$K_{\text{white}}$

**Figure 13.1:** literal nodes representing variables and OR-gate encoding the formula

## Layer 2: Parallel OR-Gates.

The OR-gates are placed on layer 2 and consist of one node $v$ which is also the output node. There is one OR-gate for every clause. Fix a clause $(l_{j,1} \vee l_{j,2} \vee l_{j,3})$. Input nodes are three pairs of nodes $(v_1, v_1')$, $(v_2, v_2')$, and $(v_3, v_3')$, where $(v_1, v_1')$ are two representative nodes of the literal clique for $l_{j,1}$, $(v_2, v_2')$ are representatives of $l_{j,2}$, and $(v_3, v_3')$ are representatives of $l_{j,3}$. That is, for each literal in the clause we connect the OR-gate on layer 2 to two of the three representative nodes of the corresponding literal clique on layer 1. The output node $v$ is additionally connected to 4 nodes of the $K_{\text{white}}$ mega-clique. Intuitively, we use the OR-gates to verify that for each clause at least one literal is TRUE. All clauses are evaluated simultaneously using an OR-gate for each clause. The OR-gate is shown in Figure 13.1.

### Layer 3: AND-Gate.

There is exactly one AND-gate on layer 3. This AND-gate consists of one output node denoted $u_0$, which has the following input nodes. It is connected to every output node of the OR-gates on layer 2 and to $m - 2$ distinct nodes of the $K_{\text{black}}$ mega-clique. Intuitively, the AND-gate is used to verify that every clause is satisfied. It is shown in Figure 13.2.

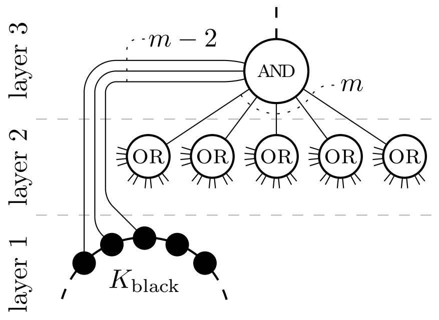


**Figure 13.2:** AND-gate to verify that each clause is satisfied

### Layers 4 to 3 + 4n: 2/3-Gates.

The 2/3-gates consist of a path $v_1$, $v_2$, $v_3$, and $v_4$. Each node of this path is connected to two distinct nodes of the $K_{\text{white}}$. The output node of the gate is $v_4$. The node $v_1$ of the first 2/3-gate on layer 4 is connected to the AND-gate on layer 3. The node $v_1$ of each of the following 2/3-gates is connected to the node $v_4$ of the previous 2/3-gate. Additionally, the input node of the $i$-th 2/3-gate is connected to three distinct nodes of the literal clique representing $x_i$ and to three distinct nodes of the literal clique representing $\overline{x}_i$ on layer 1. The output node of the final 2/3-gate is connected to $K_{\text{black}}$. An example is shown in Figure 13.3. The 2/3-gates are used to verify that we do not have variables $x_i$ in $\Phi$ for which the literal cliques of $x_i$ and $\overline{x}_i$ have the same color. Observe that 2/3-gates span over 4 layers, and we have $n$ such 2/3-gates.

Literal cliques, OR-gates, and the AND-gate use only one layer, and 2/3-gates span over 4 layers. Therefore, the total number of layers is $h = 3 + 4 \cdot n$, which results from one layer for the literal cliques, one layer for the OR-gates, one layer for the AND-gate, and $4 \cdot n$ layers containing $n$ concatenated 2/3-gates.

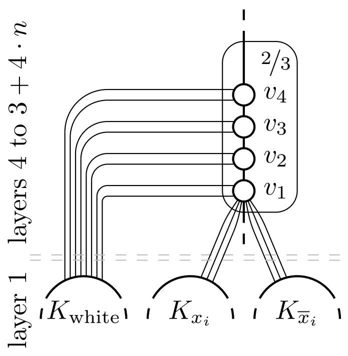


**Figure 13.3:** 2/3-gate to verify that the literal cliques have valid colors assigned

A detailed example for such a graph $G$ is given at the end of this chapter in Figure 13.5. Based on above description of $G$ we prove the following lemmas, which are then used to show Theorem 38.

**Lemma 40.** *If $\Phi$ is satisfiable, then there exists an assignment of opinions such that the convergence time in $G$ is at least $h+1$.*

To show Lemma 40, we construct an initial opinion assignment for which the gates change from black to white one layer after the other, assuming $\Phi$ is satisfiable. The full proof is as follows.

*Proof.* Let $A = (a_1, \dots, a_n)$ be an assignment of Boolean values to the $n$ variables in $\Phi$ which satisfies $\Phi$. We need to show that there exists an opinion assignment on $G$ for which the deterministic binary majority process requires at least $h+1$ steps to converge. In the following, we construct such an opinion assignment.

Let $f_A$ be an initial opinion assignment in the graph $G$ that represents $A$ by initializing the nodes in the literal cliques on layer 1 according to the assignment $A$ as follows. For each literal $x_i$ or $\overline{x}_i$, $a_i$ assigns either TRUE or FALSE to the literal. We denote a literal $x_i$ or $\overline{x}_i$ which is assigned TRUE as *positive* and literals which are assigned FALSE as *negative*. For the positive literal cliques, we assign the color black to $\lfloor L/2 \rfloor$ nodes including the representative nodes of the clique. The remaining $\lceil L/2 \rceil$ nodes, which do not have any other connections except within the literal clique, are colored white. Negative literal cliques are colored entirely black. Furthermore, we initialize all nodes of the $K_{\text{white}}$ and the $K_{\text{black}}$ with white and black, respectively. All other nodes, the paths $v_1$ to $v_4$ in the 2/3-gates, the output nodes of the OR-gates, and the output node of the AND-gate, are colored black.

The process now behaves as follows.

1. In the first step, all black nodes in every positive literal clique except the representative nodes turn white, since they have $\lceil L/2 \rceil$ white neighbors and only $\lfloor L/2 \rfloor - 1$ black neighbors.

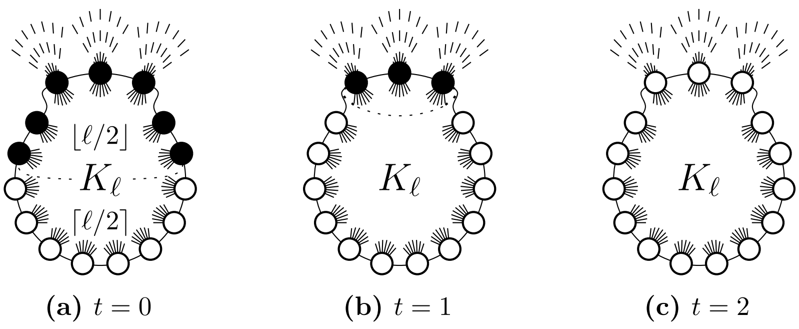


**Figure 13.4:** The figure shows the behavior of the cliques on layer 1. The three top nodes are the representatives.

2. The representative nodes of the literal cliques will turn white in the following step. This behavior of the cliques on layer 1 is shown in Figure 13.4.
3. Additionally to the neighbors in $K_{\text{white}}$, all OR-gates on layer 2 will have at least two white input nodes from representing at least one positive literal clique, since $A$ satisfies $\Phi$. Therefore, the OR-gates will turn white in step 3.
4. Once all OR-gates become white, the AND-gate has a total of $m$ white input nodes that form a clear majority against the $m - 2$ edges to black nodes in $K_{\text{black}}$ and the 1 edge to the black node of the first 2/3-gate. Therefore, the AND-gate turns white in step 4.
5. In the following $4 \cdot n$ steps, node after node and gate after gate the 2/3-gates turn white. Once all nodes of the 2/3 gates have turned white, the process stops.

Summing up over all of the above steps, the convergence time of the process w.r.t. the initial opinion assignment $f_A$ is exactly $\mathfrak{T}(G(\Phi), f_A) = 4+4\cdot n = h+1$. Therefore, the voting time in $G(\Phi)$ for a satisfiable $\Phi$ is at least $h + 1$, which yields the lemma. □

It remains to show that if $\Phi$ is not satisfiable, then the voting time in $G$ is strictly less than $h + 1$. Recall that the voting time is the maximum of the convergence time over all possible initial opinion assignments.

**Lemma 41.** *If $\Phi$ is not satisfiable, then there is no assignment of opinions such that the convergence time in $G$ is at least $h + 1$.*

Before we prove this lemma, we establish several auxiliary lemmas which require the following definitions. Let $u_0$ denote the output node of the AND-gate. Consider the graph $G'$ induced by the node of the AND-gate and the nodes of the 2/3-gates. Let $u_i$ be the node at distance $i$ to $u_0$ in $G'$. We observe that $G'$ is a path $u_0, \ldots, u_\kappa$ consisting of the $4 \cdot n + 1$ top layers of the graph $G$. Consequently, $\kappa = 4 \cdot n$ and $u_i$ is the $i$-th node on this path.

**Definition 14** (Stable Time)**.** *We define the* stable time $s(v)$ *for any node* $v \in V$ *to be the first time step such that* $v$ *does not change its opinion in any subsequent time step* $t' > s(v)$ *over all possible initial configurations. That is,*

$$s(v) = \min\Big\{t : \forall f_0 \in \{0,1\}^V \ \forall t' \geq t \quad f_{t'}(v) = f_t(v)\Big\} \ .$$

*Accordingly, let for any subset* $V' \subseteq V$ *be* $s(V')$ *defined as* $s(V') = \max\{s(v) : v \in V'\}$*.*

In the following, let $V_K$ be the set of nodes of all cliques in $G(\Phi)$, that is, the nodes contained in the literal cliques and in the mega-cliques on layer 1. Furthermore, let $V_{K^{\mathrm{R}}}$ be the set of representatives of the cliques and $V_{K^-} = V_K \setminus V_{K^{\mathrm{R}}}$. That is, every clique $K$ on layer 1 consists of $K^- \cup K^{\mathrm{R}}$. Finally, let $V_{\mathrm{OR}}$ be the set of all output nodes of OR-gates. The following lemma shows that the layers become stable one after the other.

**Lemma 42.** *It takes at most* 3 *time steps for the layers* 1 *and* 2 *consisting of literal cliques and* OR*-gates to become stable. Precisely, we have*

*(i)* $s(V_{K^-}) = 1$,

*(ii)* $s(V_{K^{\mathrm{R}}}) = 2$, *and*

*(iii)* $s(V_{\mathrm{OR}}) = 3$.

*Proof.* The lower bounds for all three claims follow from the initial opinion assignment $f_A$ described in the proof of Lemma 40. We now show the upper bounds. In the following, let $f_0$ be an arbitrary but fixed initial opinion assignment.

(i) Let $K$ be an arbitrary but fixed clique and let $c \in \{0,1\}$ be the majority color among the nodes of $K$. Let furthermore $K^-$ be the set of clique nodes that do not have connections to any other node except within the clique, that is, $K^-$ contains all clique nodes except representatives. Note that all nodes in $K^-$ only have connections to all other nodes in $K$. Since $K$ is odd and $c$ is the majority color in $K$, each node $v \in K^-$ with $f_0(v) = c$ will have at least $\lfloor \ell/2 \rfloor$ neighbors out of a total of $\ell - 1$ neighbors colored $c$. Therefore, each node $v \in K^-$ with $f_0(v) = c$ will keep its color $c$ such that $f_1(v) = c$. However, all other nodes $v' \in K^-$ with $f_0(v') \neq c$ will change their opinion to $c$, since they have at least $\lceil \ell/2 \rceil$ neighbors out of a total of $\ell - 1$ neighbors colored $c$, such that $f_1(v') = c$.

   By construction and by the size of the clique, $\ell$, all nodes $v \in K^-$ have more neighbors in $K^-$ than in $V \setminus K^-$. Therefore, for all consecutive steps $t' \geq 1$, we have for any $v \in K^-$ that $f_{t'}(v) = c$. This holds for all cliques on layer 1, and thus $s(V_{K^-}) \leq 1$.

(ii) Let $K$ be an arbitrary but fixed clique and let $v \in K^{\mathrm{R}}$ be a representative node of $K$. By construction, $v$ has a majority of its neighbors in $K^-$ and hence from (i) we derive $s(v) \leq 2$. Therefore, $s(V_{K^{\mathrm{R}}}) \leq 2$. We also observe that all nodes in $K^{\mathrm{R}}$ have the same color after the second step, since the nodes in $K^-$ become monochromatic in the first step and these nodes dominate the behavior of the nodes in $K^{\mathrm{R}}$.

(iii) Let $v$ be the output node of an arbitrary but fixed OR-gate in $V_{\mathrm{OR}}$. We observe that all neighbors of $v$ except for one neighbor (the node of the AND-gate $u_0$) are stable for any time step $t' \geq 2$. By (ii), at time 2 all representatives of any literal $x_i$ have the same color and $K_{\text{black}}$ is stable. Therefore, at time 2 an even number of neighbors of $v$ are black and an even number is white. Since the total number of neighbors of $v$ is 10, we observe that $u_0$ cannot influence $f_{t'}(v)$ for $t' \geq 2$. Moreover, by (i) and (ii) we have at time $t' \geq 2$ that the majority of neighbors having color $c$ does not change and therefore $v$ becomes stable at time 3. Thus $s(V_{\mathrm{OR}}) \leq 3$ holds. □

The above lemma gives bounds on the stable time of layers 1 and 2. In the following, we argue that whenever a node changes its opinion in any step $t$ after time step 3, it will not change its color in any subsequent time step $t' \geq t$ any more. We therefore define the so-called *activation time* of a node $v \in G'$ as follows.

**Definition 15** (Activation Time). *Let $c$ be the color of the $K_{black}$ mega-clique at time 2 and let $f_0$ be an arbitrary but fixed initial opinion assignment. We define the* activation time *of a node $v \in G'$ to be the first time step after time step 3 in which the node $v$ adopts opinion $c$. That is, $a(v) = \min\{t \geq 3 : f_t(v) = c\}$. If $v$ does not change its color after time step 3 we write $a(v) = 3$.*

We now use the above definition to state the following lemma, which describes that every node $u_i \in G'$ with $i \geq 1$ changes its color at most once after time step 3. Note that this covers the nodes of the 2/3-gates.

**Lemma 43.** *Let $f_0$ be an arbitrary but fixed initial opinion assignment. Let $t$ be the activation time w.r.t. $f_0$ of the node $u_i \in G'$ with $i \geq 1$ such that $t = a(u_i)$. Then for all $t' \geq t$ we have $f_{t'}(u_i) = f_t(u_i)$.*

*Proof.* By Lemma 42, all nodes $u \in V_{K^{\mathrm{R}}}$ are stable at $t' \geq 2$. We now distinguish two cases.

**Case 1:** $i \mod 4 \neq 1$. Observe that $u_i$ can only change its color at time $t = a(u_i)$, if it had a different color than $K_{\text{white}}$ in the previous round. This holds, since every node $u_i$ with $i \mod 4 \neq 1$ has the same number of connections to $K_{\text{white}}$ than to nodes in $V \setminus K_{\text{white}}$. Since furthermore the process behaves lazy, any node $u_i$ which has the same color as $K_{\text{white}}$ cannot change its opinion back to the opposite color any more.

**Case 2:** $i \mod 4 = 1$. The node $u_i$ is a $v_1$ node of the $j$-th 2/3-gate with $j = \lceil i/4 \rceil$. Therefore it is connected to three representatives of each literal clique for $x_j$ and $\overline{x}_j$. The literal representatives of $x_j$ and $\overline{x}_j$ are stable at time

$t' \geq 2$. Now if $x_j$ and $\overline{x}_j$ have the same color $c$, then $u_i$ has $6 > |N(u_i)|/2$ edges to nodes of color $c$. Therefore, the node does not change its color any more after time step 3. That is, we have $a(u_i) = 3$ and also $f_{t'}(u_i) = c$ for any consecutive time step $t' \geq 3$. If, however, $x_j$ and $\overline{x}_j$ do not have the same color, these edge *cancel* each other out and the color of node $u_i$ is determined by $u_{i-1}$, $u_{i+1}$, and $K_{\text{white}}$. Therefore, the same argument as in the first case holds. □

In the following we examine the behavior of layer 3 which contains only the AND-gate. Recall that $u_0$ is the output node of the AND-gate. The next lemma describes the following fact. The AND-gate $u_0$ can only change its color in a time step $t \geq 4$ if $u_1$ changed its color in time step $t-1$. After this change at time $t$, the node $u_0$ cannot change its color again.

**Lemma 44.** *Let $f_0$ be an arbitrary but fixed initial opinion assignment and let furthermore $t$ be the round after node $u_1$ has been activated such that $t = a(u_1) + 1$. For all consecutive rounds $t' \geq t$ we have $f_{t'}(u_0) = f_t(u_0)$. That is, the* AND*-gate does not change its opinion any more once the node $u_1$ has become stable.*

*Proof.* Note that $t$ is at least 4 by definition of the activation time. Let $c$ be the color of $K_{\text{black}}$ and $\overline{c} = 1 - c$ the opposite color of $c$. If at most $m-2$ of the OR-gates have color $\overline{c}$, then the node of the AND-gate has at least $2 + (m-2) > |N(u_0)|/2$ neighbors which will be colored $c$ for all $t \geq 3$ and therefore the AND-gate will be colored $c$ for every $t' \geq 4$.

If, however, $m-1$ of the OR-gates have color $\overline{c}$, only one OR-gate has not been activated and has color $c$. Thus the node of the AND-gate $u_0$ has on layer 1 and layer 2 a total of $m-1$ neighbors of color $c$ and also a total of $m-1$ neighbors of color $\overline{c}$. That is, these neighbors *cancel* each other out. By Lemma 42 the cliques and gates on layers 1 and 2 do not change their color for any $t' \geq 4$. Therefore, the node $u_0$ can only be influenced by $u_1$ and the color of $u_0$ at time $t$ is the color of $u_1$ at time $t-1$ for any $t \geq 4$. By Lemma 43 we know that $u_1$ may change its opinion only once in a round $t = a(u_1) \geq 3$ and therefore for any round $t' \geq t+1$ we have $f_{t'}(u_0) = f_t(u_0)$.

Finally, if $m$ of the OR-gates are colored $\overline{c}$, then $u_0$ has $m > |N(u_0)|/2$ neighbors of color $\overline{c}$ and since by Lemma 42 these $m$ neighbors do not change their color for $t \geq 4$ we have $f_t(u_0) = \overline{c}$ for all $t \geq 4$. Thus, in all cases the claim follows. □

The following lemma implies that in order to reach a convergence time of $h+1$ the gates on the path $u_0, \ldots, u_\kappa$ in $G'$ have to activate one after the other starting with $u_0$ at time 4. Recall that $\kappa = 4 \cdot n$.

**Lemma 45.** *Let $f_0$ be an arbitrary but fixed initial opinion assignment and let $u_i \in G'$ be a node with $0 \leq i \leq \kappa$. If $a(u_i) < i + 4$ w.r.t. $f_0$, then $\mathfrak{T}(G(\Phi), f_0) < h + 1$.*

*Proof.* By Lemma 42 all nodes of $V_K$ and $V_{\text{OR}}$ are stable after time step 2 and 3, respectively. From Lemma 43 we observe that every node of $u_1, \ldots, u_k$ with $k = 4 \cdot n$ can only change its color once after time step 3. Note that from Lemma 44 we conclude that if $u_1$ changes its color at time $t$ then the AND-gate does not change its color for any $t' \geq t + 1$.

We now consider the *inner* nodes of the path $u_j$ for which $1 \leq j < k$. In order for a node $u_j$ to change its color at time $t > 3$, one of the neighboring nodes $u_{j-1}$ or $u_{j+1}$ must have changed its color at time $t - 1$. This follows, since according to Lemma 42 all other neighbors of the node $u_j$ are already stable after 2 steps. Now if a node $u_j$ changes its opinion, one of the neighbors of $u_j$ must have changed its opinion in the previous round. This can only be either $u_{j-1}$ or $u_{j+1}$ (or both), since all other neighbors of $u_j$ are already stable.

Since all nodes $u_1, \ldots, u_k$ of the path in $G'$ can only change their color once and since $u_0$ becomes stable one time step after $u_1$ changes its color, the convergence time of the graph $G(\Phi)$ is dominated by the behavior of the path. That is, in order to achieve a *long* convergence time, the path must change its color one node after the other, resulting in a convergence time in $\Omega(n)$. Observe that this can only happen if the entire path has a different color than the $K_{\text{white}}$ after the second step. As soon as one of the path nodes is assigned the same opinion as the $K_{\text{white}}$ mega-clique, the entire path will be activated too early and the process stops prematurely.

Now in order to have a convergence time of $h + 1$, the path in $G'$, $u_0, \ldots, u_k$, must activate from $u_0$ over $u_1$ up to $u_k$ or in the reverse direction from $u_k$ over $u_{k-1}$ down to $u_0$. We now argue that activating from $u_k$ down to $u_0$ cannot yield a convergence time of at least $h + 1$.

Note that all neighbors of $u_k$ except for $u_{k-1}$ are stable at any time step $t \geq 3$. Therefore, $u_k$ either has the same fixed opinion as the $K_{\text{white}}$ and thus $a(u_k) = 3$, or $u_k$ has an activation time $a(u_k) = a(u_{k-1}) + 1$. Now in the first case, $a(u_k) = 3$, the convergence time is bounded by $3 + k = 3 + 4n < h + 1$, since the path becomes stable one node after the other starting with the node $u_k$. That is, the resulting convergence time is strictly less than $h + 1$. In the second case, $a(u_k) \geq 4$, we note that $a(u_k) = a(u_{k-1}) + 1$ and thus the path cannot activate from $u_k$ down to $u_0$.

We conclude that in order to have a convergence time of $h + 1$ the nodes must activate from $u_0$ to $u_k$ starting with node $u_0$ in time step 4 such that $a(u_0) = 4$. Therefore, $a(u_i)$ must be $i + 4$ to have a convergence time of $h + 1$ which shows the lemma. $\square$

In the following two lemmas, we enforce that initial opinion assignments which do not represent valid assignments of Boolean values to literal cliques result in premature termination of the deterministic binary majority process in $G(\Phi)$. An assignment is called *illegal* if there exist literal cliques such that the majority of $x_i$ and the majority of $\overline{x}_i$ have the same initial color.

**Lemma 46.** *For any illegal initial opinion assignment $f_I$ to $G(\Phi)$, the convergence time $\mathfrak{T}(G(\Phi), f_I)$ is strictly less than $h+1$.*

*Proof.* In the following we use $K^{\mathrm{R}}(x_i)$ and $K^{\mathrm{R}}(\overline{x}_i)$ to denote the representative nodes of the literal cliques for $x_i$ and $\overline{x}_i$. Note that by Lemma 42 these representative nodes are stable at time 2. Now assume both cliques have color $c$ after the second step.

Let $u$ be the first node $v_1$ of the $i$-th 2/3-gate. By the construction of $G(\Phi)$, $u$ is connected to 6 representative nodes of literal cliques which all share the same color $c$. Since the representative nodes are stable after 2 steps, $u$ will also have color $c$ for every time step $t' \geq 3$. That is, $a(u) = 3$ and thus by Lemma 45 the convergence time is less than $h+1$. □

**Lemma 47.** *If after two time steps $K_{white}$ and $K_{black}$ have the same color, the process stops after strictly less than $h+1$ steps.*

*Proof.* Let $c$ be the color of both mega-cliques after two time steps. Note that from Lemma 42 we conclude that both cliques are stable at time 2. Therefore $u_k$ activates at most at time 3, that is, $a(u_k) = 3$. By induction, one can show that $u_i$ will activate at most at time $3 + k - i$. Hence $u_1$ becomes activated at most at time $t = 3 + k - 1 < h$ and $u_0$ at most at time $t = 3 + k$ which is strictly less than $h+1$. Since by Lemma 42 all other nodes are also stable at time $3 + k < h + 1$ the claim follows. □

We now combine above lemmas and prove Lemma 41.

*Proof of Lemma 41.* In the following we assume that $K_{\text{white}}$ and $K_{\text{black}}$ have opposite colors after the second step, since otherwise the convergence time is less than $h+1$ as shown in Lemma 47. W.l.o.g., assume $K_{\text{white}}$ is colored white and $K_{\text{black}}$ is colored black. Furthermore, we assume that the assignment is legal, since otherwise the convergence time is less than $h+1$ as shown in Lemma 46. Finally, we also assume that $u_1, \ldots, u_k$ are initially black, since otherwise the convergence time is less than $h+1$ as shown in Lemma 45. Note that this especially covers the node $u_1$ which we assume to be black at time 4, since otherwise again the convergence time is less than $h+1$ according to Lemma 45.

According to the assumption of Lemma 41, $\Phi$ is not satisfiable. That is, for every possible assignment of Boolean values to the variables in $\Phi$, there exists a clause $(l_1 \vee l_2 \vee l_3)$ where all literals $l_1$, $l_2$, and $l_3$ are FALSE. Therefore, for any legal initial opinion assignment $f_0$ in $G(\Phi)$, the representative nodes of the corresponding literal cliques will be black at time 2. Consequently, the OR-gate corresponding to that clause will be stable with color black at time 3.

This implies that the AND-gate is black as long as $u_4$ is black since at least $(m-2)+1+1 > |N(u_0)|/2$ neighbors are black. Since the AND-gate is black, we can only have $a(u_1) = 5$ if $a(u_2) = 4$. According to Lemma 45, this results in a convergence time strictly less than $h + 1$. Note that if $f_3(u_1) = 1$, then $u_2$ will be activated at time 4 and again by Lemma 45 this yields that the convergence time is less than $h + 1$. □

Finally, we combine Lemma 40 and Lemma 41 to show Theorem 38 as follows. For a full example of a reduction, see Figure 13.5 on page 122.

*Proof of Theorem 38.* It is easy to see that VTDP is in NP. Furthermore, we can polynomially reduce 3SAT to VTDP. The correctness proof of the reduction follows from Lemma 40 and Lemma 41. Therefore we conclude that VTDP is NP-complete. □

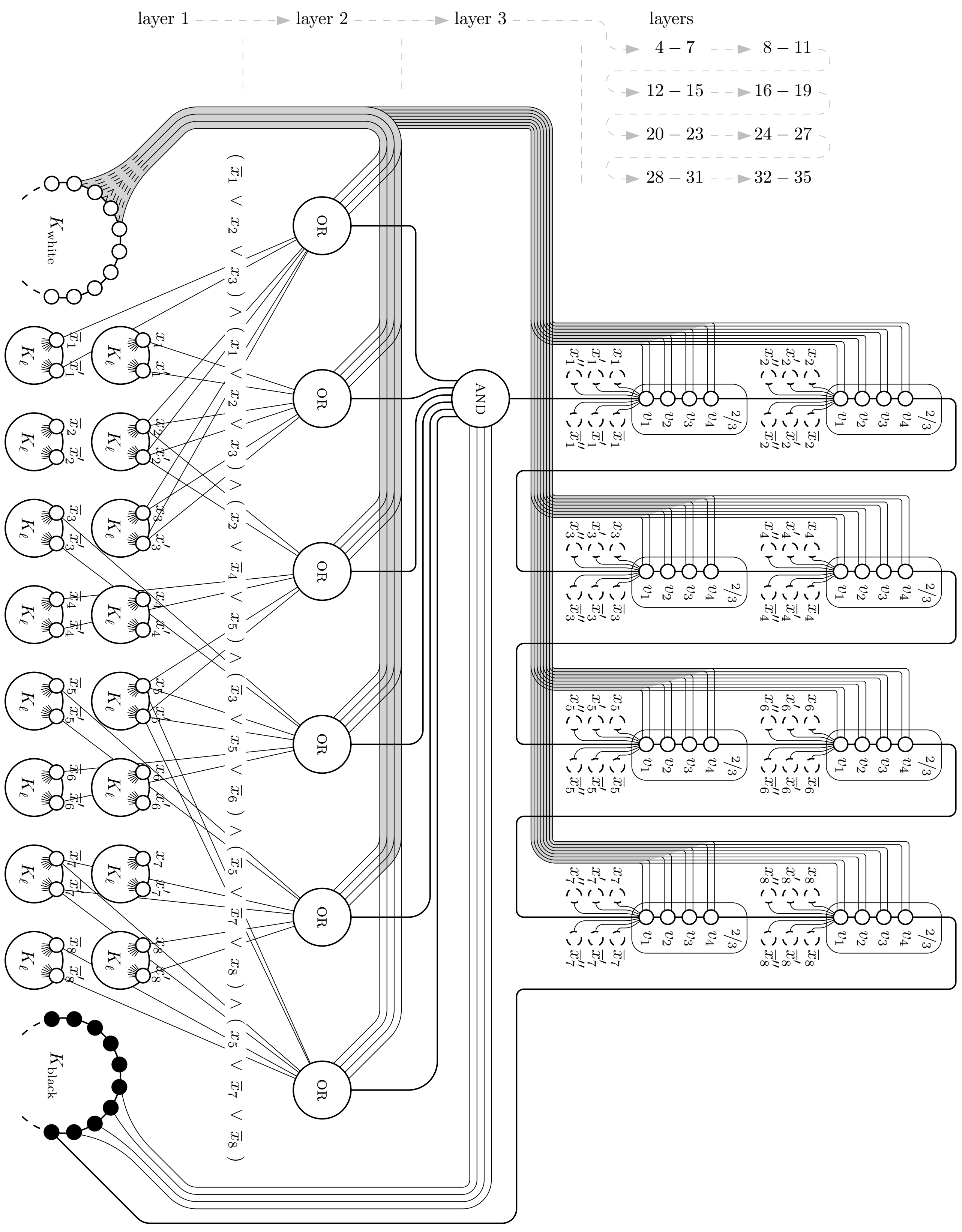

layer 1
layer 2
layer 3
layers
4 – 7
8 – 11
12 – 15
16 – 19
20 – 23
24 – 27
28 – 31
32 – 35
$K_{\text{white}}$
$K_{\text{black}}$
$K_\ell$
$(\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x}_4 \vee x_5) \wedge (\overline{x}_3 \vee x_5 \vee \overline{x}_6) \wedge (\overline{x}_5 \vee \overline{x}_7 \vee x_8) \wedge (x_5 \vee \overline{x}_7 \vee \overline{x}_8)$
OR
AND
$v_1$
$v_2$
$v_3$
$v_4$
2/3

**Figure 13.5:** a full example for a reduction

14

# Bounds on the Voting Time

Since the voting time decision problem VTDP is NP hard, we cannot hope to calculate the voting time of a graph efficiently. Nevertheless, in this chapter we show that it is possible to obtain non-trivial upper bounds on the voting time that are easy to compute. This section is dedicated to proving our upper bound on the voting time, Theorem 39. The main contribution of this theorem is the influence of symmetry which is studied in Section 14.2.

We start by giving a formal version of the potential function argument [GO80, PS83] as conceived in [Win08b]. In the following we assume that each edge in $\{x, y\} \in E$ can be replaced by two directed edges $(x, y)$ and $(y, x)$. The main idea is based on so-called *bad arrows* defined as follows.

**Definition 16.** *Let $G = (V, E)$ be a graph with initial opinion assignment $f_0$. Let $v$ denote an arbitrary but fixed node and $u \in N(v)$ a neighbor of $v$. Let $t$ denote an arbitrary but fixed round. The directed edge $(v, u)$ is called* bad arrow *if and only if the opinion of $u$ in round $t + 1$ differs from the opinion of $v$ in round $t$. We will also denote the bad arrows which have their tail at round $t = 0$ as* initial bad arrows.

Intuitively, each of these directed edges $(v, u)$ can be seen as *advice* given from $v$ to $u$ in the voting process. In the case of a bad arrow the advice was not followed by $u$ since it has a different opinion in the following round than $v$. Observe that each bad arrow is incident at exactly two nodes and thus we say it is *outgoing* in the node at its tail and incoming in the node at its head. An example of such a bad arrow can be seen in Figure 14.1.

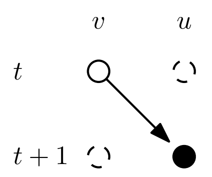
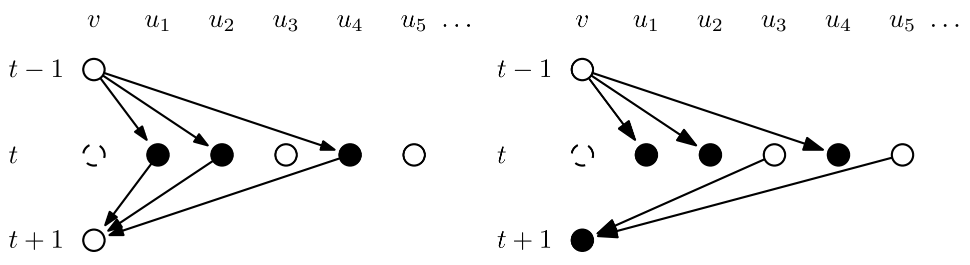


**Figure 14.1:** *bad arrow* from node $v$ to node $u$ in round $t$

**Theorem 48.** *Let $G = (V, E)$ be a graph which contains only vertices of odd degree. The voting time of the deterministic binary majority process on $G$ is at most $1 + W_{bad}$ where $W_{bad}$ is an upper bound on the number of initial bad*



**(a)** Case 1: $f_{t+1}(v) = f_{t-1}(v)$ **(b)** Case 2: $f_{t+1}(v) \neq f_{t-1}(v)$

**Figure 14.2:** Above figures show examples for the two cases. In the first case, the number of outgoing bad arrows from $v$ in round $t-1$ equals the number of incoming bad arrows at $v$ in round $t+1$. In the second case the node $v$ has color black in round $t+1$ due to a majority for black in round $t$. Therefore the number of incoming bad arrows at $v$ in round $t+1$ is strictly smaller than the number of outgoing bad arrows from node $v$ in round $t-1$.

*arrows for any initial opinion assignment on $G$. In particular, the voting time of $G$ is at most $2 \cdot |E| + 1$.*

*Proof.* The idea of the proof is to define a potential function $\phi_t$ that is strictly monotonically decreasing over the time. Let $f_0$ be any initial opinion assignment. The potential function $\phi_t$ is simply the number of bad arrows defined in Definition 16, that is,

$$\phi_t = \phi_t(G, f_t) = |\{(v,u) \in E : f_{t+1}(u) \neq f_t(v)\}| \ .$$

Let $v$ denote an arbitrary but fixed node. To show that $\phi_t$ indeed is a strictly monotonically decreasing potential function as long as $t \leq \mathfrak{T}(G, f_0)$ we distinguish the following two cases.

**Case 1.** The node $v$ has the same opinion in round $t+1$ as in round $t-1$, that is, $f_{t+1}(v) = f_{t-1}(v)$.

For each neighbor $u$ of $v$ that has a different opinion in round $t$ than $v$ in round $t-1$, there is a bad arrow from $v$ to $u$. We denote the number of these outgoing bad arrows leaving round $t-1$ as $m_{t-1}(v)$, that is,

$$m_{t-1}(v) := |\{u \in N(v) \mid f_t(u) \neq f_{t-1}(v)\}| \ .$$

There is an incoming bad arrow at node $v$ in round $t+1$ from each neighbor that has a different opinion in round $t$. Let $n_{t+1}(v)$ be this number, that is,

$$n_{t+1}(v) := |\{u \in N(v) \mid f_t(u) \neq f_{t+1}(v)\}| \ .$$

Now recall that $v$ has the same opinion in round $t+1$ as in round $t-1$. Thus, the number of incoming bad arrows at node $v$ in round $t+1$ is the same as the number of bad arrows leaving node $v$ in round $t-1$, which gives us

$$n_{t+1}(v) = m_{t-1}(v) \ . \tag{14.1}$$

An example for this case is shown in Figure 14.2a.

**Case 2.** The node $v$ has a different opinion in round $t+1$ than in round $t-1$, that is, $f_{t+1}(v) \neq f_{t-1}(v)$.

Let $m_{t-1}(v)$ and $n_{t+1}(v)$ be defined as above. Since $v$ changed its opinion after round $t-1$, either in step $t$ or in step $t+1$, there is an incoming bad arrow at node $v$ in round $t+1$ for every neighbor of $v$ that did not have an incoming bad arrow in round $t$. Now the key is that node $v$ can only have its current opinion in round $t+1$ if there is a clear majority in round $t$ in favor of this opinion among all of its neighbors. Observe that this is where the odd degrees mentioned in the problem statement [Win08a] indeed play a role. Since every node has odd degree, there is always a clear majority among its neighbors and no tie between opinions can ever occur. Now if there is a clear majority in round $t$, the number of incoming bad arrows at node $v$ in round $t+1$ will be strictly smaller than the number of outgoing bad arrows at node $v$ in round $t-1$, that is,

$$n_{t+1}(v) < m_{t-1}(v) \quad . \tag{14.2}$$

An example for this case is shown in Figure 14.2b.

**Both cases.** We take the sum over all outgoing bad arrows leaving the nodes in round $t-1$ and obtain $M_{t-1} = \sum_{v \in V} m_{t-1}(v)$. Analogously, we take the sum over all incoming bad arrows in round $t+1$ which gives us $N_{t+1} = \sum_{v \in V} n_{t+1}(v)$. However, since each bad arrow is incident in exactly two nodes, we conclude that the sum over all incoming bad arrows in round $t+1$ is the same as the sum over all outgoing bad arrows in round $t$. This gives us

$$M_t = N_{t+1} = \phi_t \quad . \tag{14.3}$$

If the deterministic binary majority process has reached a two-periodic state in round $t$, from (14.1) and (14.3) we get

$$\phi_t = N_{t+1} = M_t = \sum_{v \in V} m_t(v) = \sum_{v \in V} n_{t+2}(v) = N_{t+2} = \phi_{t+1} \quad .$$

Now assume that the deterministic binary majority process has not yet reached a two-periodic state in round $t$. That is, at least one node has a different opinion in round $t+1$ than it had in round $t-1$. Then from (14.2) and (14.3) we get

$$\phi_t = N_{t+1} = M_t = \sum_{v \in V} m_t(v) > \sum_{v \in V} n_{t+2}(v) = N_{t+2} = \phi_{t+1}$$

which proves that the voting time of the deterministic binary majority process on $G$ is bounded from above by the initial number of bad arrows.

In particular, since there can be a bad arrow only between ordered pairs of adjacent nodes, the initial number of bad arrows is bounded by $2 \cdot |E|$. Together with the observation that above argument can only be applied after the first step this implies that

$$\mathfrak{T} \leq 2 \cdot |E| + 1 \quad . \qquad \square$$

Note that in Theorem 48 it is assumed that all nodes of the graph have odd-degree. In the following we show how to remove this assumption.

**Definition 17.** *Let $G = (V, E)$ be a graph. The graph $G^* = (V, E^*)$ is the graph obtained by adding a self loop to every node of even degree in $G$. More formally,*

$$E^* = E \ \cup \bigcup_{v \in V_{\text{even}}} (v, v) \ .$$

From the definition it follows that $|E^*| = |E| + |V_{\text{even}}|$.

**Theorem 49.** *The voting time of the deterministic binary majority process on any graph $G = (V, E)$ is at most $1 + W_{bad}$, where $W_{bad}$ is an upper bound on the number of initial bad arrows in $G^*$.*

*Proof.* For every node $v \in V$ the sequence of opinions, $(f_t(v))$, is exactly the same for the deterministic binary majority process in $G$ as for the deterministic binary majority process in $G^*$. Indeed, every odd-degree node has the same neighborhood in both, $G$ and $G^*$, thus the process is the same for these nodes. Now consider an arbitrary even-degree node $v$ and fix a round $t$. If in $G$ there is a tie in round $t$, $v$ behaves lazily in $G$ and keeps its own opinion at round $t$. In $G^*$, the node $v$ considers its own opinion and thus also stays with its own opinion. If on the other hand there is a clear majority in $G$, this majority has a winning margin of at least 2, since $v$ has even degree. Thus, the impact of the self loop can be neglected and again $v$ behaves the same in $G^*$ as in $G$.

We can thus bound the voting time of $G$ by applying Theorem 48 to the odd-degree graph $G^*$. □

Observe that while the number of bad arrows is used in the potential function, the convergence time is, however, not monotone w.r.t. the number of initial bad arrows. This is argued in full detail in Chapter 15.

The upper bound on the voting time considered in [KPW14] follows from the $2 \cdot |E|$ upper bound on the number of bad arrows of Theorem 48. Clearly, this result can be improved by a factor of 2 by simply applying the observation that the number of initial bad arrows in $G^*$ is at most $|E| - |V_{odd}|/2$ as shown in the proof of the following lemma.

**Lemma 50.** *Let $G = (V, E)$ be a graph. The number of initial bad arrows in $G^*$ is at most $|E| - |V_{odd}|/2$.*

*Proof.* From the definition of the deterministic binary majority process we conclude that only less than half of a node's neighbors could have had a different opinion at time $t = 0$, since otherwise the node would have changed its own opinion. Formally, for any $v \in V$ it holds that

$$\sum_{u \in N(v)} [f_1(v) \neq f_0(u)] \leq \frac{|N(v)|}{2} \ .$$

Also, for odd-degree nodes the above inequality is strict. Therefore, the number of incoming bad arrows at a node at time $t = 1$ is smaller than half of its degree (strictly, for odd nodes). Thus, summing up all initial bad arrows we get $(2 \cdot |E| - |V_{\mathrm{odd}}|)/2$, which concludes the proof. □

Therefore we obtain the following corollary.

**Corollary 51.** *The voting time of the deterministic binary majority process on any graph $G = (V, E)$ is at most $1 + |E| - |V_{odd}|/2$.*

*Remark.* Corollary 51 is tight for general graphs up to an additive constant of 1. Indeed, consider a path graph with an initial opinion assignment on which the opinions alternate except for the last two nodes, which share the same opinion. See Figure 14.3 for this example. For this initial opinion assignment, the process converges in $|E| - |V_{odd}|/2$ steps.

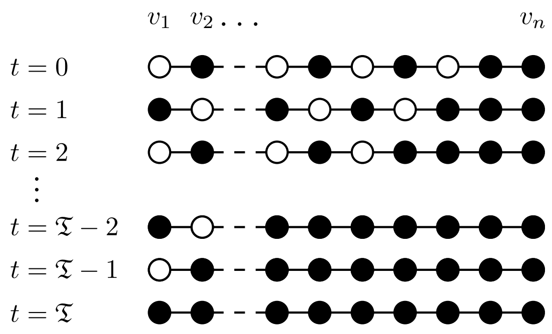


**Figure 14.3:** The figure shows an example for the tightness of the bound given in Corollary 51. The network is a path graph consisting of $|V| = n$ nodes and $|E| = n - 1$ edges. Given that $|V_{\mathrm{odd}}| = 2$, the bound given in Corollary 51 yields a voting time of at most $n - 1$. For the given initial opinion assignment, the convergence time is $n - 2$.

## 14.1 Improved Bounds for Dense Graphs

We observe that Corollary 51 is (almost) tight, and it gives us a voting time linear in the number of vertices for sparse graphs where $|E| = \mathrm{O}(|V|)$. However, for dense graphs with, e.g., $|E| = \Omega(|V|^2)$ there is room for improvement. Now the main goal in this following section is to reduce the dominant term of the voting time even further, which leads us to the following theorem.

**Theorem 52.** *Let $G = (V, E)$ be a graph. For any initial opinion assignment $f_0$ on $G$, the convergence time of the deterministic binary majority process is at most $1 + \frac{|E|}{2} + \frac{|V_{\mathrm{even}}|}{4} + \frac{7}{4} \cdot |V|$.*

To show Theorem 52 we first introduce the following definitions and auxiliary lemmas.

**Definition 18.** *An opinion assignment $f'_t$ is a $q$-*swap *of $f_t$ if for all nodes $v$*

$$f'_t(v) = f_t(v) \quad \vee \quad f'_t(v) = q \ .$$

*That is, all opinions assigned by $f'_t$ are either the original opinion assigned by $f_t$ or $q$.*

Based on this definition we can state and prove the following key lemma.

**Lemma 53** (Monotonicity)**.** *Let $f_t$ be an opinion assignment in round $t$ and $f'_t$ a $q$-swap of $f_t$. Let furthermore $v$ be a node for which $f_t(v) \neq f'_t(v)$. It holds for any time step $k \geq t$ that*

$$f_k(v) = q \implies f'_k(v) = q \ .$$

*Furthermore, any subsequent opinion assignment $f'_k$ is a $q$-swap of $f_k$.*

*Proof.* We show Lemma 53 by induction over $k$. The base case for $k = t$ is trivially true. Now suppose that Lemma 53 holds for $k \leq m$. Let $v$ be an arbitrary but fixed node for which $f_{m+1}(v) = q$. Since $f_{m+1}(v) = q$ we had a majority for $q$ among the neighbors of $v$ in the previous opinion assignment $f_m$ and according to the induction hypothesis $f'_m$ is a $q$-swap of $f_m$. Therefore, in $f'_m$ the number of nodes with opinion $q$ could have only increased, strengthening the majority for opinion $q$ even further. Thus, $f'_{m+1}(v) = q$ holds. Now assume $f'_{m+1}$ was not a $q$-swap of $f_{m+1}$. That is, there exists a node $u$ for which $f'_{m+1}(u) \neq f_{m+1}(u)$ and $f'_{m+1}(u) \neq q$. This is a contradiction to the previous statement. Together, this concludes the induction. □

In other words, Lemma 53 states that *strengthening* an opinion will never make it weaker in a subsequent round, that is, if a node ends up with opinion $q$, it also ends up with the same opinion in the $q$-swapped opinion assignment.

**Definition 19.** *An opinion assignment $f_t$ is $q$-*permanent *if $f_{t+2}$ is a $q$-swap of $f_t$.*

We now use the definition above to further bound the voting time, since the deterministic binary majority process has the property that once the process is either in a 0-permanent or 1-permanent state it will converge in a number of steps linear in $|V|$ as shown in the following lemma. Observe that a two-periodic state is both 0-permanent and 1-permanent.

**Lemma 54.** *Given a $q$-permanent opinion assignment $f_t$, the process converges in at most $2 \cdot |\{v \in V : f_t(v) \neq q\}|$ rounds.*

*Proof.* By definition, $f_{t+2}$ is a $q$-swap of $f_t$. Thus we can apply Lemma 53 and conclude that either all nodes have at time $t+2$ the same opinion as at time $t$, or some nodes have changed their opinion to $q$. That is, all nodes with opinion $q$ at time $t$ will also have opinion $q$ at time $t+2$. So there are two possibilities.

Either every two time steps at least one node switches to opinion $q$ or every node has again its former opinion and we are in a two-periodic state. Thus the process converges in at most $2 \cdot |\{v \in V : f_t(v) \neq q\}| < 2 \cdot |V|$ steps. $\square$

We will now use this result to prove an upper bound on the voting time that is better than Corollary 51 for dense graphs.

*Proof of Theorem 52.* Let $f_t$ be an opinion assignment that has not yet reached a two-periodic state at time $t$. In the proof of Lemma 54 we made the observation that there must exist a node $v \in V$ for which $f_t(v) \neq f_{t-2}(v)$. We therefore distinguish the following two cases.

**Case 1.** The opinion assignment $f_{t-2}$ is $q$-permanent.

**Case 2.** There exists *another* node $u$ with $f_{t-2}(u) \neq f_{t-2}(v)$ such that $f_t(u) \neq f_{t-2}(u)$, that is, $u$ is non-two-periodic and disagrees with $v$ at times $t$ and $t-2$.

As long as we are in case 2, by repeating the argument in case 2 of the proof of Theorem 48 we observe that the number of bad arrows drops by at least 2 in each step (one due to $v$ and another one due to $u$). According to Lemma 50, this can be the case for at most $1 + (|E| - |V_{odd}|/2)/2$ steps, since the deterministic binary majority process will converge after that time. On the other hand, if at some point we are in case 1, the process will converge in at most $2 \cdot |V|$ steps as shown in Lemma 54. Together, these two cases yield the bound

$$\begin{aligned} 1 + \frac{|E| - |V_{odd}|/2}{2} + 2 \cdot |V| &= 1 + \frac{|E|}{2} - \frac{|V_{odd}|}{4} + \frac{|V|}{4} + \frac{7}{4} \cdot |V| \\ &= 1 + \frac{|E|}{2} + \frac{|V_{\text{even}}|}{4} + \frac{7}{4} \cdot |V| \ . \end{aligned}$$ $\square$

*Remark.* One might intuitively assume that the voting time is bounded by the diameter of the network. However, this is not true, at least straightforwardly, as there exist graphs $G$ where the convergence time w.r.t. a given initial opinion assignments $f_0$ is asymptotically larger than the diameter of the network, that is, $\mathfrak{T}(G, f_0) \gg \text{diam}(G)$. Further details can be found in Chapter 15.

## 14.2 The Influence of Symmetry

We observe that the majority process is much faster on graphs that exhibit certain types of symmetry, such as the star graph, the complete graph and many other graphs in which several nodes share a common neighborhood. We investigate this feature of the process to further improve the bounds obtained so far. We recall that a set of nodes $S$ is called a *family* if and only if for all nodes $u, v \in S$ we have $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. The key fact is that these nodes of any family will behave in a similar way after the first step.

**Definition 20.** *Let* $\operatorname{fam}(u)$ *denote the family of* $u$*. We write* $u \sim v$ *if* $\operatorname{fam}(u) = \operatorname{fam}(v)$*.*

**Lemma 55.** *The relation* $\sim$ *defines an equivalence class. In particular, all nodes in the same family either form a clique or a stable set, and they all have the same degree in* $G$*.*

*Proof of Lemma 55.* Recall that $u \sim v$ is defined as $\operatorname{fam}(u) = \operatorname{fam}(v)$. Therefore, reflexivity and symmetry of $\sim$ hold trivially. It remains to show that $\sim$ is transitive, that is, $\forall u, v, w \in V \;\; u \sim v \;\wedge\; v \sim w \Rightarrow u \sim w$. By definition, we have

$$N(u) \setminus \{v\} = N(v) \setminus \{u\} \qquad \text{and} \qquad N(v) \setminus \{w\} = N(w) \setminus \{v\} \; .$$

Using the previous identities gives us

$$\begin{aligned} N(u) \setminus \{w, v\} &= (N(u) \setminus \{v\}) \setminus \{w\} = (N(v) \setminus \{u\}) \setminus \{w\} \\ &= (N(v) \setminus \{w\}) \setminus \{u\} = (N(w) \setminus \{v\}) \setminus \{u\} = N(w) \setminus \{u, v\} \end{aligned}$$

and

$$\begin{aligned} v \in N(u) &\iff u \in N(v) \iff u \in N(w) \\ &\iff w \in N(u) \iff w \in N(v) \iff v \in N(w) \; . \end{aligned} \tag{14.4}$$

That is, $v$ either belongs to both $N(u)$ and $N(w)$ or to none of them, hence (14.4) implies $N(u) \setminus \{w\} = N(w) \setminus \{u\}$. This shows transitivity of the relation $\sim$.

From the transitivity of $\sim$ it follows that all nodes in the same family either form a clique or are pairwise non-adjacent and thus form a stable set. Together with the definition

$$\operatorname{fam}(u) = \operatorname{fam}(v) \iff N(u) \setminus \{v\} = N(v) \setminus \{u\}$$

it also follows that all nodes of the same family have the same node degree. □

**Corollary 56.** *For any graph* $G$*, its asymmetric graph* $G^{\Delta}$ *is well-defined.*

*Proof.* According to Lemma 55, the set of families is a partition of the nodes of $G$. By construction of $G^{\Delta}$, every family $S$ in $G$ is replaced by one or two nodes in $G^{\Delta}$. Therefore, there is a bijection between the families in $G$ and the corresponding node or pair of nodes in $G^{\Delta}$. Hence $G^{\Delta}$ is well-defined. □

We are now ready to prove Theorem 39.

*Proof of Theorem 39.* Let $v$ and $v'$ be two nodes of the same family $\operatorname{fam}(v) = \operatorname{fam}(v')$, having the same color at time $t$. Since $v$ and $v'$ observe the same opinions in their respective neighborhood, $v$ and $v'$ will also have the same color anytime after $t$. It follows that if at some time $t$ there is a bad arrow going from $v$ to some neighbor $u$ (or from $u$ to $v$), then there will also be a bad arrow from $v'$ to $u$ (or from $u$ to $v'$). In particular, this implies that whenever the number of bad arrows adjacent to $v$ is decreased by some amount $c$, also the identical number of bad arrows adjacent to $v'$ will be decrease by the same amount $c$.

Recall the proofs of Corollary 51 and Theorem 52. An estimate of the voting time is obtained by upper bounding the number of bad arrows that can possibly disappear during the process. The main argument is the following. It suffices to only consider the bad-arrows adjacent to $v$ in $G^{\Delta}$, since the corresponding bad arrows adjacent to $v'$ will disappear whenever those adjacent to $v$ do.

Let $v$ and $v'$ be two nodes with $\mathrm{fam}(v) = \mathrm{fam}(v')$ having a different color at time $t$. We can divide every such family that contains nodes of different opinions into two sets $S_0$ and $S_1$ according to their initial opinion in the first round. Note that all nodes in either set behave identically. In particular, an adjacent bad arrow from a node $u$ to all nodes of either set disappears at the same time. Since there is bijection between the families of $G$ and the pairs of nodes and singletons of $G^{\Delta}$, and by applying Corollary 51 and Theorem 52 we can bound the voting time by bounding the bad arrows in $G^{\Delta}$. This yields the first part of the claim. Using [CH94], one can obtain the modular decomposition of $G$ in $\mathrm{O}(|E|)$ time steps. In another $\mathrm{O}(|E|)$ time steps one can select from the modular decomposition those modules that form a family, using that all nodes of a family have the same degree. Hence, $G^{\Delta}$ can be constructed in linear time. □

*Remark.* While we show for the voting time that we have $\max_f \mathfrak{T}(G^{\Delta}, f) \geq \max_f \mathfrak{T}(G, f)$, in general it is not the case that $\mathfrak{T}(G^{\Delta}, f) \geq \mathfrak{T}(G, f)$ for every opinion assignment $f$. A formal statement along with a counterexample is given in Chapter 15.

# 15

# Further Computational Properties

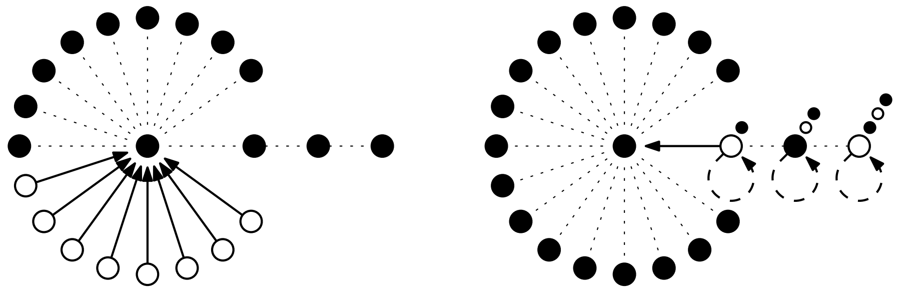
In this chapter we provide examples and prove further computational properties of the deterministic binary majority process w.r.t. the potential function of [GO80, PS83], that is, the number of *bad arrows* defined in Definition 16. While these properties are already mentioned in the previous chapter, we now give the formal statements and proofs. We show that the convergence time is not monotone w.r.t. the value of the potential function, and we investigate how many opinion assignments exhibit the same bad arrows. Overall, our results highlight the strengths and weaknesses of such a potential function approach in bounding the voting time of the deterministic binary majority process.

**Lemma 57.** *The convergence time is not monotone w.r.t. the initial number of bad arrows.*

*Proof.* Let $G$ be a graph consisting of a star graph $S_i$ with $i$ leaves that has a path graph $P_j$ of length $j$ connected to its center node such that $i > j$. We now can define two initial opinion assignments $f^{(bad)}$ and $f^{(good)}$ for which the initial number of bad arrows in $f^{(bad)}$ is greater than the initial number of bad arrows in $f^{(good)}$ but still $\mathfrak{T}(G, f^{(bad)}) < \mathfrak{T}(G, f^{(good)})$.

As $f^{(bad)}$ assignment, we color $\lceil i/2 \rceil - 1$ leaves of the star graph $S_i$ white and all other nodes, including the path $P_j$, black. As $f^{(good)}$ assignment, we color all the nodes of $S_i$ black and assign alternating opinions to the nodes of the path $P_j$. It is straightforward to verify that the described opinion assignments prove the statement. □

An example for a graph $G$ consisting of a $S_{17}$ and a $P_3$ can be seen in Figure 15.1. The example shows that even though the initial opinion assignment in Figure 15.1a has much more initial bad arrows, the deterministic binary majority process converges much faster for the opinion assignment shown in Figure 15.1b.



**(a)** initial opinion assignment $f^{(bad)}$ **(b)** initial opinion assignment $f^{(good)}$

**Figure 15.1:** The figure shows an example for the graph described in the proof of Lemma 57. It consists of a star graph $S_{17}$ joined at the center node with a path of length 3. Clearly, the initial opinion assignment $f^{(bad)}$ shown in Figure 15.1a has a total number of 8 bad arrows while the initial opinion assignment $f^{(good)}$ shown in Figure 15.1a has only one *true* bad arrow along with 3 self loop bad arrows. Still, the process will converge in only one step for $f^{(bad)}$ while it will take 3 steps for $f^{(good)}$.

Suppose that, instead of specifying the initial opinion assignment, we decide in advance what bad arrows are there. We can do that by deciding for each ordered pair $(u, v)$ for which $\{u, v\} \in E$ whether we want to have a bad arrow going from $u$ to $v$. We formalize this notion by means of the following definitions.

**Definition 21.** *Let $G = (V, E)$ be a graph and $\beta : V \times V \to \{0, 1\}$ denote a characteristic function on $V \times V$. Then $\beta$ is a bad arrows assignment on $G$ if there exists an opinion assignment $f$ on $G$ that determines $\beta$ such that $\beta$ is the indicator function of the bad arrows we have on $G$ w.r.t. the opinion assignment $f$.*

In proving upper bounds on the voting time we consider the bad arrows assignment determined by the initial opinion assignment. One may wonder whether in doing so we are *losing information*. In the following lemma we show that, given a valid bad arrows assignment, we can reconstruct the initial opinion assignment up to exchanging black and white (and up to two more possibilities in bipartite graphs).

**Lemma 58.** *Let $G$ be a connected graph and let $\beta$ be a valid bad arrows assignment on $G$. If the graph is not bipartite, there are exactly two opinion assignments, otherwise there are exactly four opinion assignments that determine $\beta$.*

*Proof.* Let $v \in V$ denote an arbitrary but fixed vertex. We now denote the set $\{v\}$ as $N_0$ and the set of direct neighbors of $v$ as $N_1$ to define the $i$-th neighborhood $N_i$ for $i \geq 2$ as

$$N_i = \left( \bigcup_{u \in N_{i-1}} N(u) \right) \setminus \left( \bigcup_{j=1}^{i-1} N_j \right) .$$

We show by an induction on $k$ that the colors of all nodes in $N_{2k}$ are determined by the color of $v$. The base-case is trivial since for $k = 0$ we have $N_0 = \{v\}$. For the induction step we observe that according to the induction hypothesis the color of each node in $N_{2k}$ is determined. We now observe that the color at time 1 of each node in $N_{2k+1}$ is determined by $\beta$ and the colors at time 0 of the nodes in $N_{2k}$. Vice versa, also the colors at time 0 of nodes in $N_{2(k+1)}$ are determined by $\beta$ and the colors at time 1 of each node in $N_{2k+1}$. This concludes the induction.
An example is shown in Figure 15.2. In this figure it is clear that $v$ and, e.g., $u_1$ must have a different color, for the following reason. Since $u_1$ does not have a bad arrow to its neighbor in $N_1$, it has the same color in the next round as this neighbor. But this neighbor's color in the next round is different to the current color of $v$ because of the bad arrow assignment.

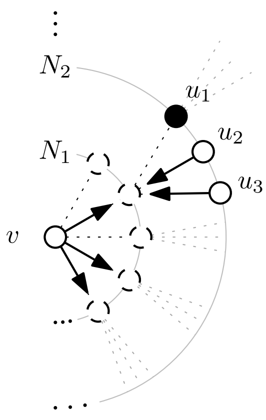


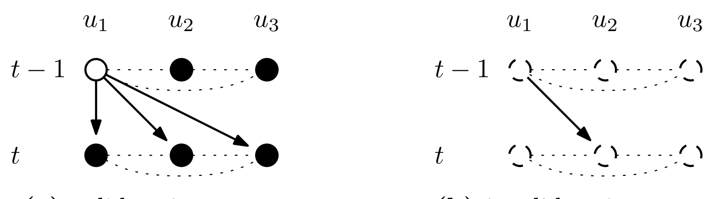
**Figure 15.2:** With a bad arrows assignment given, the opinions of each second neighborhood are uniquely determined.

Observe that from above induction the lemma follows immediately for bipartite graphs. We can fix the colors for two arbitrary nodes, one from each of the two sets of non-adjacent nodes, to determine all other nodes' colors. This gives us four possible opinion assignments for a given bad arrow assignment $\beta$. If the graph is not bipartite there must exist a cycle of odd length. The opinion assignments for all nodes of this cycle are determined by $\beta$ with the same argument as in above induction. Therefore, not only the colors of even neighborhoods $N_{2k}$ are determined, but also of odd neighborhoods $N_{2k+1}$. This leaves us with exactly two possible initial opinion assignments, which concludes the proof. □

Regarding the re-construction of opinion assignments from bad arrows, we furthermore observe the following. According to the definition we clearly have $\{u, v\} \notin E \implies \beta(u, v) = 0$ for any bad arrows assignment $\beta$. However, there do also exist characteristic functions on the (directed) set of edges of $G$ that do not form a valid bad arrows assignment. An example of such an invalid assignment that motivates above definition is shown in Figure 15.3.

Figure 15.3 shows two different assignments of bad arrows for the $K_3$, a clique of size 3. The left assignment is valid, whereas the right assignment cannot be valid. This is since in cliques of odd size all nodes share the same opinion after exactly one step. Therefore all nodes at step $t$ will have the same opinion. Since, however, $u_1$ had in step $t - 1$ a different opinion than this majority opinion in step $t$, a bad arrow must exist between $u_1$ and $u_3$ (and also a loop from $u_1$ to itself, if we consider self-loops).

As already stated at the end of Section 14.1, one might intuitively assume



**(a)** valid assignment **(b)** invalid assignment

**Figure 15.3:** Not every characteristic function on the directed edges is a valid bad arrows assignment. In above graph, the $K_3$, all nodes share the same opinion after at most one time step. Therefore, $u_1$ and $u_3$ have the same opinion at time $t$. Since $u_1$ has a bad arrow to $u_2$, there must be also a bad arrow from $u_1$ to $u_3$.

that the voting time is bounded by the diameter of the network. However, the following lemma shows that this, at least straightforwardly, is not true.

**Lemma 59.** *For any given graph $G$ with diameter $\Delta$, there exists a graph $G'$ with the following properties.*

- *For any opinion assignment $f$ for $G$, there exists an assignment $f'$ for $G'$ such that the convergence time of $G$ is the same as in $G'$*
- *The diameter of $G'$ is constant*
- *$G$ is a subgraph of $G'$.*

*Proof.* We augment $G$ by adding a clique $C_0$ of size $n$ where all nodes have Opinion 0 to $G$. We then add node $u_0$ initialized with 0 and connect it to all nodes of $G$ and $C_0$. Symmetrically, we add a clique $C_1$ of size $n$ where all nodes have Opinion 1 to $G$. We then add a node $u_1$ initialized with 1 and connect it to all nodes of $G$ and $C_1$. Note that every node $u \in G$ is also in $G'$ and the opinion of $u$ is the same in both graphs for any point in time. Hence the convergence time remains the same in $G'$ and the claim follows by observing that $G'$ has a constant diameter. □

Note that above lemma shows that for any connected graph $G = (V, E)$ and any initial opinion assignment $f_0$ one can construct another graph $G'$ which has $G$ as an induced subgraph, asymptotically the same number of nodes and edges, the same convergence time for a related initial opinion assignment, but a constant diameter. However, there are even examples of graphs where the convergence time of the deterministic binary majority process w.r.t. a given initial opinion assignment $f_0$ is asymptotically larger than the diameter of the network without modifying the graph, that is, $\mathfrak{T}(G, f_0) = \omega(\text{diam}(G))$.

An example for such a graph is shown in Figure 15.4. In this example, we are given a two-dimensional grid $G$ of size $|V| = \sqrt{n} \times \sqrt{n}$. Clearly, the diameter of this graph is $2 \cdot \sqrt{n}$. However, by laying a winding *serpentine* path of white nodes in an entirely black grid as initial opinion assignment $f_0$ we can force the process to require a convergence time of $\mathfrak{T}(G, f_0) = \Omega(n) \gg \text{diam}(G) = O(\sqrt{n})$.

In Theorem 39 in Section 14.2, we show that for the voting time we have $\max_f \mathfrak{T}(G^\Delta, f) \geq \max_f \mathfrak{T}(G, f)$. However, in general it is not the case that

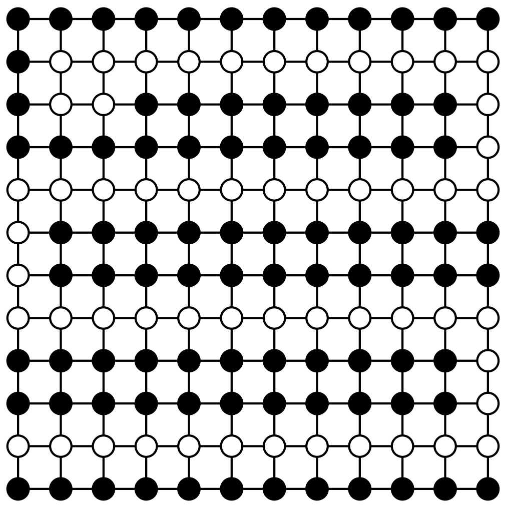
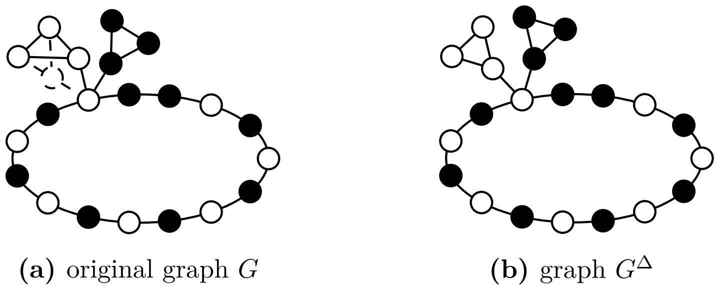


**Figure 15.4:** a two-dimensional grid $G$ with $\mathrm{diam}(G) = \sqrt{n}$ and an initial opinion assignment that converges only after $\Omega(n)$ steps

$\mathfrak{T}(G^{\Delta}, f) \geq \mathfrak{T}(G, f)$ for every opinion assignment $f$, as we formally state and show in the following lemma.

**Lemma 60.** *Let $G = (V, E)$ be a graph with initial opinion assignment $f$ and $G^{\Delta}$ be the asymmetric graph constructed from $G$. In general, it does not hold that $\mathfrak{T}(G^{\Delta}, f) \geq \mathfrak{T}(G, f)$.*

*Proof.* An example for such a graph for which $\mathfrak{T}(G^{\Delta}, f) \leq \mathfrak{T}(G, f)$ is shown in Figure 15.5. □



**(a)** original graph $G$ **(b)** graph $G^{\Delta}$

**Figure 15.5:** The left graph $G$ is a circle graph with additional gadgets connected to one node. The dashed node is removed to obtain the graph $G^{\Delta}$ shown on the right.

Part IV

# Rapid Plurality Consensus

# 16 Introduction

In this final part, we again consider the problem of distributed voting in a network of players modeled as a graph $G = (V, E)$ with $|V| = n$. Each node in the network starts with one initial opinion, which we will also call color, from a set of possible opinions. We distinguish between the synchronous and the asynchronous setting. In the *synchronous model*, all nodes simultaneously communicate with their neighbors and update their opinions according to some function of their neighborhood. In the *asynchronous model*, we assume that each node has a random clock which ticks according to a Poisson distribution once per time unit in expectation. Again, upon activation the nodes update their opinion according to their neighborhood.

Regardless of the underlying model of synchronicity, if eventually all nodes agree on one opinion, we say this opinion *wins* and the process *converges*. Typically, one would demand from such a voting procedure to run accurately, that is, the opinion with the highest number of initial supporters should win with decent probability $1 - o(1)$, and to be efficient, that is, the voting process should converge within as few communication steps as possible. Additionally, voting algorithms are usually required to be simple, fault-tolerant, and easy to implement [HP01, Joh89].

Distributed voting algorithms have applications in a multitude of fields. In distributed computing, applications contain, among others, consensus [HP01] and leader election [BMPS04]. Early results in other areas can be attributed to distributed databases [Gif79] where voting algorithms have been used to serialize read and write operations. In game theory, distributed voting is used to analyze social behavior [DP94]. Also, as already discussed in Part III, the existence and characterization of so-called *monopolies*, which are sets of nodes that dominate the outcome of the voting process, have been investigated [Ber01, Pel02, Pel14, ACF+15]. Various processes based on the *majority rule* were analyzed in the study of *influence networks* [FKW13, LM15], and they have been used to measure the competition of opinions in social networks [MT14].

Variants of these processes are applied for distributed community detection [CG10, KPS13, RAK07]. In computational sciences, voting processes can be utilized to model chemical reaction networks [Dot14], neural and automata networks [GM90], and cells' behavior in biology [CC12].

## Pull Voting

One major line of research on plurality consensus has its roots in gossiping and rumor spreading. Communication in these models is often restricted to pull requests, where nodes can query other nodes' opinions and use a simple rule to update their own opinion. See [Pel02] for a slightly dated but thorough survey.

One straightforward variant is the so-called *pull voting* running in discrete rounds, during which each player contacts a node chosen uniformly at random from the set of its neighbors and adopts the opinion of that neighbor. The two works by Hassin and Peleg [HP01] and Nakata et al. [NIY99] have considered the discrete time two-opinion voter model on connected graphs. In these papers, each node is initially assigned one of two possible opinions. Their main result is that the probability for one opinion $\mathcal{A}$ to win is $P_{\mathcal{A}} = d(\mathcal{A})/(2m)$, where $d(\mathcal{A})$ denotes the sum of the degrees of all vertices supporting opinion $\mathcal{A}$. It has furthermore been shown by Hassin and Peleg [HP01] that the expected time for the two-opinion voting process to converge on general graphs can only be bounded by $\mathrm{O}(n^3 \log n)$. Tighter bounds for the expected completion time on random $d$-regular graphs have been shown in [CFR09]. In [CEOR13], Cooper et al. showed that the convergence time for pull voting on any connected graph $G = (V, E)$ is asymptotically almost always $\mathrm{O}(n/(\nu(1 - \lambda_2)))$. In this bound, $\lambda_2$ is the second largest eigenvalue of the transition matrix of a random walk on the graph $G$. The parameter $\nu$ measures the *regularity* of $G$ with $1 \leq \nu \leq n^2/(2m)$, where the equality $\nu = 1$ holds for regular graphs. Recently, in [BGKM16] it was shown that the voting time is bounded by $\mathrm{O}(m/(\delta \cdot \phi))$, where $m$ is the number of edges in the graph, $\phi$ is the conductance of the underlying graph, and $\delta$ is the minimum degree.

The expected convergence time for pull voting is at least $\Omega(n)$ on many graphs, such as regular expanders and complete graphs. Taking into account that solutions to many other fundamental problems in distributed computing, such as information dissemination [KSSV00] or aggregate computation [KDG03], are known to run much more efficiently, Cooper et al. noted that there is room for improvement. To address this issue, Cooper et al. [CER14] introduced the two-choices voting process. In this modified process, one is given a graph $G = (V, E)$ where each node has one of two possible opinions. The process runs in discrete rounds during which, other than in classical pull voting, every node is allowed to contact two neighbors chosen uniformly at random. If both neighbors have the same opinion, then this opinion is adopted, otherwise the calling vertex retains its current opinion in this round.

They show that in random $d$-regular graphs, with high probability all nodes agree after $\mathrm{O}(\log n)$ steps on the largest initial opinion, provided that $c_1 - c_2 = K \cdot (n\sqrt{1/d + d/n})$ for $K$ large enough, where $c_1$ and $c_2$ denote the sizes of the initially largest and second-largest colors. For an arbitrary $d$-regular graph $G$, they need $c_1 - c_2 = K \cdot \lambda_2 \cdot n$. In the more recent work by Cooper et al. [CER+15], the results from [CER14] have been extended to general expander graphs, cutting out the restrictions on the node degrees but nevertheless proving that the convergence time for the voting procedure remains in $\mathrm{O}(\log n)$. Recently, the authors of [CRRS16] showed the following bound on the consensus time in regular expanders. If the initial bias between the largest and second-largest opinion is at least $c_1 - c_2 \geq Cn \max\{\sqrt{\log n / c_1}, \lambda_2\}$, where $\lambda$ is the absolute second eigenvalue of the matrix $P = Adj(G)/d$ and $C$ is a suitable constant, then the largest opinion wins in $\mathrm{O}((n \log n)/c_1)$ steps, with high probability.

One extension is five-sample voting in $d$-regular graphs with $d \geq 5$, where in each round at least five distinct neighbors are consulted. Abdullah and Draief showed an $\mathrm{O}(\log_d \log_d n)$ bound [AD15], which is tight for a wider class of voting protocols. A more general analysis of multi-sample voting has been conducted by Cruise and Ganesh [CG14] on the complete graph.

Becchetti et al. [BCN+14] consider a similar update rule on the clique for $k$ opinions. Here, each node pulls the opinion of three random neighbors and adopts the majority opinion among those three (breaking ties uniformly at random). They need $\mathrm{O}(\log k)$ memory bits and prove a tight run time of $\Theta(k \cdot \log n)$ for this protocol, given a sufficiently large bias $c_1 - c_2$. Moreover, they show that if the bias is only of order $\sqrt{kn}$, then with constant probability the difference $c_1 - c_2$ decreases. As we show in this part, the two-choices process behaves differently since the difference required by the two choices process is only $\Omega(\sqrt{n \log n})$. The reason for this phenomenon is that the variance of the number of nodes switching per round differs largely in these two processes. In the regime where all opinions are roughly of the same size, the probability of switching in the two-choices process is $\mathrm{o}(1)$, whereas it is $1 - \mathrm{o}(1)$ in the 3-majority process. More details can be found in Section 17.2.

In another recent paper, Becchetti et al. [BCN+15b] build upon the idea of the 3-state population protocol by Angluin et al. [AAE08]. Using a slightly different time and communication model, they generalize the protocol to $k$ opinions. In their model, nodes act in parallel and in each round pull the opinion of a random neighbor. If it holds for the largest color that $c_1 \geq (1+\varepsilon) \cdot c_2$ for a constant $\varepsilon > 0$, the number of colors is bounded by $k = \mathrm{O}\Big((n/\log n)^{1/3}\Big)$, and under presence of $\log k + \mathrm{O}(1)$ bits of memory, their protocol agrees with high probability on the plurality opinion in time $\mathrm{O}(\mathrm{md}(\mathbf{c}) \cdot \log n)$ in the clique. Here, $\mathrm{md}(\mathbf{c})$ is the so-called *monochromatic distance* that depends on the initial opinion distribution $\mathbf{c}$. In contrast to all the results above for $k > 2$ opinions, we only require a bias of size $\mathrm{O}(\sqrt{n \log n})$.

Also interested in balancing the requirement for additional memory with convergence time, in [BFGK16] the authors propose two plurality consensus protocols. Both assume a complete graph and realize communication via the random phone call model. The first protocol is very simple and, with high probability, achieves plurality consensus within $\mathrm{O}\big(\log(k) \cdot \log\log_\gamma n + \log\log n\big)$ rounds using $\Theta(\log\log k)$ bits of additional memory. The second, more sophisticated protocol achieves plurality consensus within $\mathrm{O}\big(\log(n) \cdot \log\log_\gamma n\big)$ rounds using only 4 overhead bits. In both cases, $k$ denotes the number of colors, and $\gamma$ denotes the initial relative plurality gap, the ratio between the plurality opinion and the second-largest opinion. They require an initial absolute gap of $\omega\big(\sqrt{n}\log^2 n\big)$. At the heart of their protocols lies the use of the *undecided state*, originally introduced by Angluin et al. [AAE08]. A very recent result by Ghaffari and Parter [GP16] introduces a protocol for plurality consensus with time and memory bounds similar to our bounds for Algorithm 18.1. They employ a similar basic idea of consolidation and bit-propagation rounds, which they refer to as selection and recovery. While aspects of [GP16] and the first protocol in [BFGK16] are similar to our own protocol described in Chapter 18 (in terms of expectation but not distribution), they were all developed independently and initially approached the problem with different specific objectives.

Another interesting model allows for adversarial corruption of opinions. Doerr et al. [DGM+11] investigate the so-called 3-median rule which allows an adversary to arbitrarily change the opinion of $F = \sqrt{n}$ arbitrary nodes. The required time to reach near-consensus is $\mathrm{O}(\log k \log\log n + \log n)$, where $k$ is the size of the set of opinions. Their algorithm assumes a total ordering on the opinions and requires nodes to be able to perform basic algebraic operations. In a recent paper, Becchetti et al. [BCN+16] overcome these assumptions and show that the 3-majority rule is stable against an $F = \mathrm{o}(\sqrt{n})$ dynamic-adversary. It is worth noting that both [BCN+16, DGM+11] are only interested in consensus and not necessarily plurality, which would mean that the initially dominant color wins with high probability if the initial bias is large enough.

## Population Protocols

The second major line of work on majority voting considers *population protocols*, in which the nodes usually act asynchronously. In its basic variant, nodes are modeled as finite state machines with a small state space. Communication partners are chosen either adversarially or randomly, see [AAER07, AR07] for a more detailed description. Angluin et al. [AAE08] propose a 3-state (that is, constant memory) population protocol for majority voting with $k = 2$ in the clique to model the mixing behavior of molecules. We refer to their communication model as the *sequential model.* In each time step, an edge is chosen uniformly at random, such that only one pair of nodes communicates. To allow for an easier comparison with the synchronous model, we will normalize the run time of all sequential algorithms and continuous processes throughout this part by dividing their runtime by $n$ [AGV15]. To make this explicit, we sometimes refer to this as *parallel time.* This is a typical measure for population protocols and based on the intuition that, in expectation, each node communicates with one neighbor within $n$ time steps. If the initial bias $c_1 - c_2$ is $\omega(\sqrt{n}\log n)$, their protocol lets all nodes agree with high probability on the majority opinion in $O(n \cdot \log n)$ steps. Mertzios et al. [MNRS14] showed that this 3-state protocol fails on general graphs, that is, there are infinitely many graphs on which it returns the minority opinion or has exponential run time. They also provide a 4-state protocol for *exact* majority voting, which in time $O(n^5)$ *always* returns the majority opinion, independently of the initial bias, in arbitrary graphs, and in time $O(n^2 \cdot \log n/(c_1 - c_2))$ in the clique. This result is optimal in that no population protocol for exact majority can have fewer than four states. In a recent paper, Alistarh et al. [AGV15] gave a sophisticated sequential protocol for $k = 2$ in the clique. It solves exact majority and has, with high probability, parallel run time $O\left(\log^2 n/(s \cdot (c_1 - c_2)) + \log^2 n \cdot \log s\right)$, where $s$ is the number of states with $s = O(n)$ and $s = \Omega(\log n \cdot \log\log n)$. Recall that the parallel run time in the sequential model is the number of sequential time steps divided by $n$ [AGV15]. Most recently, Cooper et al. [CDFR16] considered the *Discordant* voting processes on finite graphs for $k = 2$. In their protocol, an edge whose endpoint colors differ is said to be discordant and a vertex is discordant if it incident to a discordant edge. The authors consider different protocols assuming that at every step a randomly chosen discordant edge is chosen. In contrast to the protocols described above, we assume that $k$ is a function that grows with $n$.

### Further Models

Apart from the two research lines mentioned above, there is a multitude of related but quite different models. They differ, for example, in the consensus requirement, the time model, or the graph models. For completeness, this paragraph gives a small overview over such variants.

In one very common variant of the voter model [AF02, CEOR13, DW83, HP01, HL75, LN07, Lig12], the authors are interested in the time it takes for the nodes to agree on any arbitrary opinion. Another variant [PVV09] of distributed voting considers the 3-state protocol from [AAE08] for two opinions in the complete graph, but in a continuous time model. In [AD15], Abdullah and Draief consider majority voting on special graphs given by a degree sequence. Other protocols such as the one presented by Draief and Vojnović [DV12] guarantee convergence to the majority opinion. Moreover, Berenbrink et al. [BFK+16] use load balancing algorithms to solve the plurality consensus in general graphs and for general bias. However, in the setting where the bias is of order $\sqrt{n \log n}$ and the number of colors is polynomial in $n$, their run time becomes substantial.

## 16.1 Model

In the following section, we will formally introduce the model which we consider in the remainder of this part. We give a formal definition of the consensus process in the synchronous and the asynchronous model followed by an overview of our results in Section 16.2.

### Two-Choices Model

We consider the following classical plurality consensus process. We are given a graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges. In this network, we run the following process in discrete time steps $t \in \mathbb{N}^0$, starting with time $t = 0$. Initially, the nodes are partitioned into $k$ groups representing $k$ colors $\mathcal{C}_1, \dots, \mathcal{C}_k$. We denote the set of all colors as $C = \{\mathcal{C}_1, \dots, \mathcal{C}_k\}$. Also, we will occasionally abuse notation and use $\mathcal{C}_i$ to denote the *set* of all vertices having color $\mathcal{C}_i$. At time $t$, every node chooses two neighbors uniformly at random, with replacement. If the chosen nodes' colors coincide and this color is not the color of the node itself at time $t-1$, then the node switches to this new color at time $t$. We denote this process as the *plurality consensus process with two choices*. We will say that the process *converges* when all nodes have the same color.

Let $t$ denote an arbitrary but fixed time step and let $c_1, \dots, c_k$ be the numbers of nodes of colors $\mathcal{C}_1, \dots, \mathcal{C}_k$ at time step $t$. W.l.o.g., we assume that at every time step $t$ the colors are ordered in descending order such that $c_1 \geq c_2 \geq \cdots \geq c_k$. Note that in our analysis we will justify this assumption by showing that the initial plurality color $\mathcal{C}_1$ always remains the largest color with high probability. Our first two results from Chapter 17 and Chapter 18 will be shown w.r.t. this synchronous model.

In the following, we will denote the dominating color $\mathcal{C}_1$ as $\mathcal{A}$ with size $a = c_1$ and we will use $\mathcal{B}$ to denote the second largest color $\mathcal{C}_2$ of size $b = c_2$.

### Parallel Asynchronous Model

In the asynchronous model, we are again given a graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges. In this network, we run the following process, starting at time $t_0 = 0$. As before, the nodes are initially partitioned into $k$ groups representing the $k$ colors $\mathcal{C}_1, \dots, \mathcal{C}_k$. Each node $v$ is equipped with a random clock which ticks in the unit time interval according to a Poisson distribution with parameter $\lambda = 1$. That is, we assume a memory-less random clock, such that for every node the time between two ticks is exponentially distributed with parameter $\lambda = 1$. Consequently, from the memory-less property it follows that at any time $t$ each node has the same probability $1/n$ to be the next one to tick.

Whenever a node's clock ticks, the node samples two neighbors $u_1$ and $u_2$

uniformly at random, with replacement, from $V$. If the chosen nodes' colors coincide, $v$ adopts this color. Otherwise, $v$ keeps its current color. We denote this process as the *asynchronous plurality consensus process with two choices*.

### Sequential Asynchronous Model

While the parallel model described above represents real-world processes for which event frequencies are commonly modeled by Poisson clocks, we give in the following a more theoretical yet equivalent model.

We observe that the Poisson distribution used for the clocks in the parallel model has the so-called *memory-less property*. That is, at any given time $t$, regardless of the previous events, every node has exactly the same probability to be the next node to tick, namely $1/n$. We furthermore assume that, upon a node's activation, the execution of one step occurs atomically, that is, no two nodes are ever active concurrently. Therefore, instead of considering the asynchronous parallel process in real time, we rather analyze the process in the so-called sequential model. In this sequential model, we assume that a discrete time is given by the sequence of ticks, and at any of the discrete time steps, a node is selected to perform its task uniformly at random from the set of all nodes.

Observe that we can relate the number of ticks in the sequential model to the real time in the asynchronous model as follows, see also [AGV15]. We have for any tick $t$ in the asynchronous sequential model that $\mathrm{E}[T_t] = t/n$, where $T_t$ is the random variable for the real time of tick $t$. Moreover, for the expected number of ticks allotted by the asynchronous voting algorithm described in Chapter 19, we obtain that the real time is concentrated around the expected value such that with high probability the asynchronous voting process converges after at most $\mathrm{O}(\log n)$ real time units. See, e.g., [BGPS06, Lemma 1] for details on the concentration.

### Stability

In our analysis, we will show that the two-choices process can tolerate the presence of an adversary which is allowed to arbitrarily change the opinion of up to $F = c_1(c_1 - c_2)/(8n)$ arbitrarily selected nodes after every round. We will show that under these assumptions our two-choice process still guarantees that with high probability a vast majority of nodes accept the plurality opinion, that is, the initially most dominant opinion. Observe that, similarly, all our theorems also hold if the adversary is allowed to change opinions at the *beginning* of a round. We use a definition similar to the definition by Becchetti et al. [BCN+16], which in turn has its roots in [AAE08, AFJ06].

**Definition 22.** *A* stabilizing near-plurality protocol *ensures the following properties:*

1. Almost agreement. *Starting from any initial configuration, in a finite number of rounds, the system must reach a regime of configurations where all but a negligible* bad *subset of nodes of size at most* $\mathrm{O}(n^{\varepsilon})$ *for some constant* $\varepsilon < 1$ *support the same opinion.*
2. Almost validity. *Given a large enough initial bias, the system is required to converge to the plurality opinion* $\mathcal{A}$*, with high probability, where all but a negligible* bad *set of nodes have opinion* $\mathcal{A}$.
3. Non-termination. *In dynamic distributed systems, nodes represent simple and anonymous computing units which are not necessarily able to detect any global property.*
4. Stability. *The convergence to such a weaker form of agreement is only guaranteed to hold with high probability.*

## 16.2 Our Contribution

Our first main contribution is an extension of the results by Cooper et al. [CER14] on the complete graph to more than two colors. That is, in our model we assume that every node of the $K_n$ initially has one of $k$ possible opinions where $k = \mathrm{O}(n^{\epsilon})$ for some small positive constant $\epsilon$. In the following, we state this as our first main theorem.

**Theorem 61.** *Let* $G = K_n$ *be the complete graph with* $n$ *nodes. Let* $k = \mathrm{O}(n^{\varepsilon})$ *be the number of opinions for some small constant* $\varepsilon > 0$. *Let* $c_1$ *be the size of the largest opinion at the beginning of the process. The plurality consensus process with two choices defined in Algorithm 17.1 on* $G$ *converges to* $\mathcal{A}$ *with high probability within* $\mathrm{O}(n/c_1 \cdot \log n)$ *time steps, if the initial bias is at least* $c_1 - c_2 \geq z \cdot \sqrt{n \log n}$ *for some constant* $z$. *Moreover, assuming this bias, the process fulfills the stabilizing near-plurality conditions in presence of any* $F = c_1(c_1 - c_2)/(8n)$*-dynamic adversary.*

The difficulty in the analysis lies in the possibly diminishingly small initial *mass* of $\mathcal{A}$ in comparison to the mass of all other colors. Interestingly, the required initial gap does not depend on the number of opinions present. As we show later, the required number of time steps is $\Omega(n/c_1 + \log n)$. Moreover, we show that if $c_1 - c_2 = \mathrm{O}(\sqrt{n})$, then $\mathcal{B}$ wins with constant probability.

The resilience of two choices in comparison to other protocols comes at the price of a large run time. To overcome this issue, we combine the two choices process with a rumor spreading algorithm. This allows us to obtain a significantly faster algorithm, which we denote `OneBit`.

For the algorithm `OneBit`, we investigate a slightly modified model which we call the *memory model*. This improved model is described in full detail in Chapter 18. In this model, we allow each node to store and transmit one additional bit. As stated in Theorem 62, this allows us to reduce the run time from $\mathrm{O}(n/c_1 \cdot \log n)$ to $\mathrm{O}((\log(c_1/(c_1 - c_2)) + \log\log n) \cdot (\log k + \log\log n)) =$

$\mathrm{O}\left(\log^2 n\right)$, while still the dominating color wins with high probability, assuming only a slightly larger initial bias towards the dominating color than in the two-choices approach. The bound becomes $\mathrm{O}(\log\log n \cdot (\log k + \log\log n))$ for $c_1 \geq c_2\left(1 + 1/\log^{\mathrm{O}(1)} n\right)$. If we assume that a tight upper bound on $n/c_1$ is known to the nodes, the run time of `OneBit` can further be improved to $\mathrm{O}((\log\log n) \cdot (\log(n/c_1) + \log\log n))$. The theorem is formally stated as follows.

**Theorem 62.** *Let $G = K_n$ be the complete graph with $n$ nodes. Let $k = \mathrm{O}(n^{\varepsilon})$ be the number of opinions for some small constant $\varepsilon > 0$. The plurality consensus process* `OneBit` *defined in Algorithm 18.1 on $G$ converges within*

$$\mathrm{O}((\log(c_1/(c_1 - c_2)) + \log\log n) \cdot (\log k + \log\log n))$$

*time steps to $\mathcal{A}$, with high probability, if $c_1 - c_2 \geq z \cdot \sqrt{n \log^3 n}$ for some constant $z$.*

This can be further improved to $\mathrm{O}((\log(c_1/(c_1 - c_2)) + \log\log n) \cdot \log k)$ if we change the algorithm slightly as described in Chapter 18. As mentioned earlier, essentially the same result was obtained, independently, by Berenbrink et al. [BFGK16] with the first protocol in their paper.

Note that also in the classical two-choices protocol each node implicitly is assumed to have local memory, which is used, e.g., to store its current opinion. The main difference between the classical model and the memory model is that in the memory model each node also transmits one additional bit along with its opinion when contacted by a neighbor. In contrast to existing work considering $k > 2$, our algorithm ensures that the dominant color $\mathcal{A}$ wins within a small (at most $\mathrm{O}\left(\log^2 n\right)$) number of rounds, even if the bias is only $\mathrm{O}\left(\sqrt{n \log^3 n}\right)$. The thorough analysis of this synchronous algorithm is the basis for understanding and analyzing the corresponding asynchronous protocol.

Our final main contribution is an adaption of the algorithm `OneBit` to the asynchronous setting. The main question is whether the same (or similar) results as in the synchronous case can also be obtained in the asynchronous setting. As discussed below in more detail, a straight-forward observation is that in the sequential asynchronous model many nodes may remain *unselected* for up to $\mathrm{O}(\log n)$ time steps. Therefore no algorithm can converge in $\mathrm{o}(\log n)$ steps. Thus, our aim is to construct a protocol that solves plurality consensus in $\mathrm{O}(\log n)$ time. We show that if the difference between the numbers of the largest two opinions is at least $\Omega(c_2)$, where $c_2$ is the size of the second largest opinion, and $k = n^{\mathrm{O}\left(1/\log^2 \log n\right)}$, then our algorithm solves plurality consensus and achieves the best possible run time of $\mathrm{O}(\log n)$, provided a node is allowed to communicate with at most constantly many other nodes in a step.

The key to the rapidity of `OneBit` is that we pair a phase in which all nodes execute the two-choice process with a phase in which successful opinions are propagated quickly – much like in broadcasting. For this to work it is crucial to separate the two phases. While this is trivial in the synchronous setting, it is impossible in the asynchronous setting. The number of activations of different nodes can easily differ by $\Theta(\log n)$, rendering any attempt of full synchronization futile if one aims for a run time of $O(\log n)$. Thus, we restrict ourselves to the concept of *weak synchronicity* as follows. At any time we only require that a fraction of $1 - o(1)$ nodes are *almost* synchronous. To cope with the influence of the remaining nodes, we rely on a toolkit of gadgets, which we believe are interesting in their own right. The obtained weak synchronicity allows us to use the high-level structure of the proof and the analysis of `OneBit`. Our result is formally stated in the following theorem.

**Theorem 63.** *Let $G = K_n$ be the complete graph with $n$ nodes. Let $k = \exp\left(O\left(\log n / \log^2 \log n\right)\right)$ be the number of opinions. Let $\varepsilon_{\text{bias}} > 0$ be a constant. The asynchronous plurality consensus process* `AsyncPlurality` *defined in Algorithm 19.1 on $G$ converges within time $\Theta(\log n)$ to the majority opinion $\mathcal{A}$, with high probability, if $c_1 \geq (1 + \varepsilon_{\text{bias}}) \cdot c_i$ for all $i \geq 2$.*

# 17 Plurality Consensus with Two Choices

In this chapter we prove our first main theorem stated in Theorem 61. The algorithm discussed in this chapter is formally defined in Algorithm 17.1.

The structure of the proofs is as expected. We show using Chernoff bounds that the number of nodes which change their opinion to $\mathcal{A}$ is larger than the number of nodes which switch to $\mathcal{B}$. Given that the initial bias is large enough, the difference between $\mathcal{A}$ and $\mathcal{B}$ increases rapidly in every round with high probability and using union bound yields the theorem. The difficult part lies in bounding the the number of switches to $\mathcal{A}$ and to $\mathcal{B}$. Indeed, just applying Chernoff bound on every single color appears to lead to much weaker results. instead, we carefully aggregate colors when considering the nodes switching to $\mathcal{A}$ and $\mathcal{B}$. Intuitively, the difficulty lies in the sheer number of initial opinions we allow. In fact, their total mass tremendously exceeds, in contrast to most previous work, the initial mass of $\mathcal{A}$.

Let $f_{ij}$ denote the random variable for the *flow* from color $\mathcal{C}_i$ to color $\mathcal{C}_j$, that is, $f_{ij}$ at a given time step $t$ represents the number of nodes which had color $\mathcal{C}_i$ at the previous time step $t-1$ and switched to color $\mathcal{C}_j$ at time $t$. We will use $c'_1, \ldots, c'_k$ to denote the number of nodes of corresponding colors after the switching has been performed before the adversary changes $F$ arbitrary nodes.

For simplicity of notation, we will assume that in the following the dominating color $\mathcal{C}_1$ is denoted as $\mathcal{A}$ with $a = c_1$. Furthermore, we will use $\mathcal{B}$ to denote the second largest color $\mathcal{C}_2$ of size $b = c_2$. Also, we will use $f_{\mathcal{AB}}$ and $f_{\mathcal{BA}}$ to denote $f_{1,2}$ and $f_{2,1}$, respectively.

Observe that in the complete graph the number $f_{ij}$ of nodes switching from $\mathcal{C}_i$ to $\mathcal{C}_j$ has a binomial distribution with parameters $f_{ij} \sim B(c_i, c_j^2/n^2)$. Clearly,

```
Algorithm two-choices($G = (V, E)$, color : $V \to C$)
    for round $t = 1$ to $|C| \cdot \log |V|$ do
        at each node $v$ do in parallel
            let $u_1, u_2 \in N(v)$ uniformly at random;
            if color($u_1$) = color($u_2$) then
                color($v$) ← color($u_1$);
```

**Algorithm 17.1:** distributed voting protocol with two choices

the expectation and variance of $f_{ij}$ are

$$\mathrm{E}[f_{ij}] = \frac{c_i \cdot c_j^2}{n^2} \qquad \text{and} \qquad \mathrm{Var}[f_{ij}] = \frac{c_i \cdot c_j^2 (n - c_j)(n + c_j)}{n^4} \ .$$

Observe that if $a \geq (1/2 + \varepsilon_1)n$ for some constant $\varepsilon_1 > 0$, the process converges within $\mathrm{O}(\log n)$ steps with high probability. This follows from [CER14] since in the case of $a \geq (1/2 + \varepsilon_1)n$ the process is stochastically dominated by the two color voting process. In order to increase readability we assume in the following that $a \leq n/2$. Furthermore, observe that $a > n/k$, since $\mathcal{A}$ is the largest of $k$ color classes. We start with the following definitions.

Let $S \subseteq C$ be a set of colors. We will use the random variable $f_{iS}$ to denote the sum of all flows from color $\mathcal{C}_i$ to any color in $S$ and $f_{Si}$ to denote the sum of all flows from any color in $S$ to $\mathcal{C}_i$. We have in expectation

$$\mathrm{E}[f_{Si}] = \sum_{\mathcal{C}_j \in S} \frac{c_j \cdot c_i^2}{n^2} \qquad \text{and} \qquad \mathrm{E}[f_{iS}] = \sum_{\mathcal{C}_j \in S} \frac{c_i \cdot c_j^2}{n^2} \ .$$

Let $\mathcal{C}_i$ be a color and $\overline{\mathcal{C}_i}$ be the set of all other colors, defined as $\overline{\mathcal{C}_i} = C \setminus \mathcal{C}_i$. We observe that after one round the new number of nodes supporting $\mathcal{C}_i$ is a random variable

$$c_i' = c_i + \sum_{j \neq i} f_{ji} - \sum_{j \neq i} f_{ij} = c_i + f_{\overline{\mathcal{C}_i} i} - f_{i \overline{\mathcal{C}_i}} \ .$$

Since all nodes perform their choices independently, the first sum $f_{\overline{\mathcal{C}_i} i}$ has a binomial distribution with parameters $f_{\overline{\mathcal{C}_i} i} \sim B(n - c_i,\, c_i^2/n^2)$. Furthermore, every node of color $\mathcal{C}_i$ changes its color away from $\mathcal{C}_i$ to any other opinion with probability $p_i^{\text{away}} = \sum_{j \neq i} c_j^2/n^2$. Therefore, the second sum $f_{i\overline{\mathcal{C}_i}}$ also has a binomial distribution with parameters $f_{i\overline{\mathcal{C}_i}} \sim B(c_i,\, p_i^{\text{away}})$. That is, we have in expectation

$$\mathrm{E}[c_i'] = c_i + \frac{(n - c_i)c_i^2}{n^2} - \frac{c_i}{n^2} \sum_{j \neq i} c_j^2 \ . \tag{17.1}$$

Note that these expected values are monotone w.r.t. the current size. This is described more formally in the following observation.

**Observation 64.** *Let $\mathcal{C}_r$ and $\mathcal{C}_s$ be two colors. It holds that if $c_r \leq c_s$ then $\mathrm{E}[c'_r] \leq \mathrm{E}[c'_s]$.*

*Proof.* We first rewrite (17.1) as

$$\mathrm{E}[c'_i] = c_i + \frac{c_i^2}{n} - \frac{c_i}{n^2}\sum_{\mathcal{C}_j} c_j^2 = c_i\left(1 + \frac{c_i}{n} - \sum_{\mathcal{C}_j}\frac{c_j^2}{n^2}\right) \ .$$

Using this representation of $\mathrm{E}[c'_i]$ gives us

$$\mathrm{E}[c'_r] = c_r\left(1 + \frac{c_r}{n} - \sum_{\mathcal{C}_j}\frac{c_j^2}{n^2}\right) \overset{c_r \leq c_s}{\leq} c_s\left(1 + \frac{c_s}{n} - \sum_{\mathcal{C}_j}\frac{c_j^2}{n^2}\right) = \mathrm{E}[c'_s] \ . \qquad \square$$

For the following lemma, recall that $\mathcal{A} = \mathcal{C}_1$ denotes the dominant color of size $a = c_1$ and $\mathcal{B} = \mathcal{C}_2$ denotes the second largest color of size $b = c_2$.

**Lemma 65.** *Let $\mathcal{A}$ be the dominating color and $\mathcal{B}$ be the second largest color. Assume that $a - b > z \cdot \sqrt{n \log n}$. There exists a constant $z$ such that $a' - b' > (a - b)(1 + a/4n)$ with high probability.*

In the following proof we utilize certain methods which have also been used in [CER14] for the two-opinion plurality consensus process with two choices in more general graphs.

*Proof.* First we observe that

$$\begin{aligned}
\mathrm{E}[a' - b'] &= a + \mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}}] - \mathrm{E}[f_{\mathcal{A}\overline{\mathcal{A}}}] - b - \mathrm{E}[f_{\overline{\mathcal{B}}\mathcal{B}}] + \mathrm{E}[f_{\mathcal{B}\overline{\mathcal{B}}}] \\
&= a + (n - a) \cdot \frac{a^2}{n^2} - \frac{a}{n^2}\sum_{\mathcal{C}_i \neq \mathcal{A}} c_i^2 - b - (n - b) \cdot \frac{b^2}{n^2} + \frac{b}{n^2}\sum_{\mathcal{C}_i \neq \mathcal{B}} c_i^2 \\
&= a - b + \frac{1}{n^2}\left(a^2 n - a^3 - b^2 n + b^3 - a\sum_{\mathcal{C}_i \neq \mathcal{A}} c_i^2 + b\sum_{\mathcal{C}_i \neq \mathcal{B}} c_i^2\right) \\
&= a - b + \frac{1}{n^2}\left(n\left(a^2 - b^2\right) - a\left(a^2 + \sum_{\mathcal{C}_i \neq \mathcal{A}} c_i^2\right) + b\left(b^2 + \sum_{\mathcal{C}_i \neq \mathcal{B}} c_i^2\right)\right) \\
&= a - b + \frac{1}{n}\left(a^2 - b^2\right) - \frac{1}{n^2}\left(a\sum_{\mathcal{C}_i} c_i^2 - b\sum_{\mathcal{C}_i} c_i^2\right) \\
&= a - b + \frac{(a - b)(a + b)}{n} - \frac{1}{n^2}\sum_{\mathcal{C}_i} c_i^2 (a - b) \\
&= (a - b) \cdot \left(1 + \frac{(a + b)}{n} - \frac{1}{n^2}\sum_{\mathcal{C}_i} c_i^2\right) \ .
\end{aligned}$$

We now use that $\mathcal{A}$ and $\mathcal{B}$ are the largest and second largest colors, respectively, to bound the sum $\sum_{\mathcal{C}_i} c_i^2$ as follows.

$$\sum_{\mathcal{C}_i} c_i^2 = a^2 + \sum_{\mathcal{C}_i \neq \mathcal{A}} c_i^2 \leq a^2 + \sum_{\mathcal{C}_i \neq \mathcal{A}} c_i \cdot b = a^2 + (n-a) \cdot b \leq a^2 + n \cdot b$$

Therefore, we obtain

$$\begin{aligned} \mathrm{E}[a' - b'] &\geq (a-b)\left(1 + \frac{(a+b)}{n} - \frac{a^2 + n \cdot b}{n^2}\right) \\ &\geq (a-b)\left(1 + \frac{a}{n} \cdot \left(1 - \frac{a}{n}\right)\right) \end{aligned}$$

and since $a \leq n/2$ we finally get

$$\mathrm{E}[a' - b'] \geq (a-b)\left(1 + \frac{a}{2n}\right) .$$

We now apply Chernoff bounds to $a' - b'$. Let $\delta_1$, $\delta_2$, $\delta_3$, $\delta_4$ be defined as

$$\delta_1 = \frac{2\sqrt{n \log n}}{a} , \qquad \delta_2 = \frac{2n\sqrt{\log n}}{\sqrt{a \sum_{\mathcal{C}_i \neq \mathcal{A}} c_i^2}} ,$$

$$\delta_3 = \frac{2\sqrt{n \log n}}{b} , \qquad \text{and} \qquad \delta_4 = \frac{2n\sqrt{\log n}}{\sqrt{b \sum_{\mathcal{C}_i \neq \mathcal{B}} c_i^2}}$$

for the corresponding random variables $f_{\overline{\mathcal{A}}\mathcal{A}}$, $f_{\mathcal{A}\overline{\mathcal{A}}}$, $f_{\overline{\mathcal{B}}\mathcal{B}}$, $f_{\mathcal{B}\overline{\mathcal{B}}}$ with expected values $\mu_1, \mu_2, \mu_3, \mu_4$ given by

$$\mu_1 = (n-a)\frac{a^2}{n^2} , \qquad \mu_2 = \frac{a}{n^2} \sum_{\mathcal{C}_i \neq \mathcal{A}} c_i^2 ,$$

$$\mu_3 = (n-b)\frac{b^2}{n^2} , \qquad \text{and} \qquad \mu_4 = \frac{b}{n^2} \sum_{\mathcal{C}_i \neq \mathcal{B}} c_i^2 .$$

Since $a \leq n/2$ we know for the second largest color $\mathcal{B}$ that $b \geq n/2k$. Together with $a \geq n/k \geq n^{1-\varepsilon}$ we get $0 < \delta_i < 1$ and $\delta_i^2 \cdot \mu_i = \Omega(\log n)$ for $i = 1, 2, 3, 4$. We now apply Chernoff bounds to $a' - b'$ and obtain with high probability

$$a' - b' \geq (a-b) \cdot \left(1 + \frac{a}{2n}\right) - E$$

where the error term $E$ is bounded as follows.

$$\begin{aligned}
E &= \delta_1 \cdot \mu_1 + \delta_2 \cdot \mu_2 + \delta_3 \cdot \mu_3 + \delta_4 \cdot \mu_4 \\
&= \frac{2\sqrt{n \log n}}{n^2} \left( an - a^2 + \sqrt{an \sum_{\mathcal{C}_i \neq \mathcal{A}} c_i^2} + bn - b^2 + \sqrt{bn \sum_{\mathcal{C}_i \neq \mathcal{B}} c_i^2} \right) \\
&\leq \frac{2\sqrt{n \log n}}{n^2} \left( \sqrt{n \sum_{\mathcal{C}_i} c_i^2} \left( \sqrt{a} + \sqrt{b} \right) + an + bn \right) \\
&\leq \frac{2\sqrt{n \log n}}{n^2} (2an + an + bn) \\
&\leq \frac{8a\sqrt{n \log n}}{n} ,
\end{aligned}$$

where we used that $\sum_{\mathcal{C}_i} c_i^2 \leq \sum_{\mathcal{C}_i} a \cdot c_i \leq an$. From the definition of the lemma we know that $(a - b) \geq z \cdot \sqrt{n \log n}$ for some constant $z$. If we assume that $z$ is large enough, e.g., $z \geq 32$, then we get with high probability

$$a' - b' \geq (a - b) \cdot \left( 1 + \frac{a}{4n} \right) . \qquad \square$$

While Lemma 65 shows that in the absence of an adversary, the difference between colors $\mathcal{A}$ and $\mathcal{B}$ does indeed increase in every round with high probability, it does not cover the remaining colors $\mathcal{C}_j$ for $j \geq 3$ nor does it cover an adversary. To show that also the smaller colors $\mathcal{C}_j$ do not interfere with $\mathcal{A}$ and thus the minimum of the difference between $\mathcal{A}$ and any $\mathcal{C}_j$ increases, we use the following coupling.

At any time step $t$, there exists a bijective function which maps any instance of the two-choices protocol at time $t$ to another instance of the same protocol such that the outcome $c'$ of the first instance is at most the outcome $b'$ of the mapped instance.

**Lemma 66.** *Let $\mathcal{A}$ be the dominating color of size $a$ and let $\mathcal{B}$ be the second largest color of size $b$. Let $\mathcal{C} \neq \mathcal{A}, \mathcal{B}$ be one of the remaining colors of size $c$. Furthermore, let $\pi : V \to V$ be a bijection and let $P$ be the original process. We can couple a process $P' = P(\pi)$ to the original process $P$ such that $c'^{\,(P)} \leq b'^{\,(P')}$, where $c'^{\,(P)}$ is the random variable $c'$ in the original process and $b'^{\,(P')}$ is the random variable $b'$ in the coupled process.*

*Proof.* Let $t$ be an arbitrary but fixed round. In the following, we use the notation that $\mathcal{B}_t$ and $\mathcal{C}_t$ are sets containing all vertices of colors $\mathcal{B}$ and $\mathcal{C}$, respectively, in round $t$. As before, we have color sizes $b = |\mathcal{B}_t|$ and $c = |\mathcal{C}_t|$. The proof proceeds by a simple coupling argument. We start by defining $\hat{\mathcal{B}}_t, \mathcal{B}_t^*, \mathcal{C}_t^* \subseteq V$ as follows. Let $\hat{\mathcal{B}}_t$ be an arbitrary subset of $\mathcal{B}_t$ such that $|\hat{\mathcal{B}}_t| = |\mathcal{C}_t|$. Let furthermore $\mathcal{B}_t^*$ be defined as $\mathcal{B}_t^* = \mathcal{B}_t \setminus \hat{\mathcal{B}}_t$, and finally let $\mathcal{C}_t^*$ be an arbitrary subset of $V \setminus (\mathcal{B}_t \cup \mathcal{C}_t)$ such that $|\mathcal{C}_t^*| = |\mathcal{B}_t^*|$.

Additionally, we construct the bijective function $\pi : V \to V$ as follows. Let $\hat{\pi}$ be an arbitrary bijection between $\mathcal{C}_t$ and $\hat{\mathcal{B}}_t$. Let furthermore $\pi^*$ be an arbitrary bijection between $\mathcal{C}_t^*$ and $\mathcal{B}_t^*$. We now define $\pi$ as

$$\pi(v) = \begin{cases} \hat{\pi}(v) & \text{if } v \in \mathcal{C} \ , \\ \hat{\pi}^{-1}(v) & \text{if } v \in \hat{\mathcal{B}} \ , \\ \pi^*(v) & \text{if } v \in \mathcal{C}^* \ , \\ \pi^{*-1}(v) & \text{if } v \in \mathcal{B}^* \ , \\ v & \text{if } v \in V \setminus (\mathcal{B}_t \cup \mathcal{C}_t \cup \mathcal{C}_t^*) \ . \end{cases} \tag{17.2}$$

A graphical representation of $\pi$ can be seen in Figure 17.1.

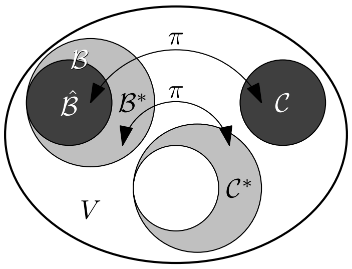


**Figure 17.1:** schematic representation of the bijective function $\pi$ defined in (17.2)

It can easily be observed that $\pi$ indeed forms a bijection on $V$. We now use $\pi$ to couple a process $P' = P(\pi)$ to the original process $P$, to show that $b'^{(P')} \geq c'^{(P)}$, where the notation $b'^{(P)}$ means the variable $b'$ in the original process $P$ and $c'^{(P')}$ means the variable in the coupled process $P'$. Let $u \in V$ be an arbitrary but fixed node. The coupling is now constructed such that whenever $u$ samples a node $v \in V$ in the original process $P$, then $u$ samples $\pi(v)$ in the coupled process $P'$.

Let $X$ be the set of nodes which change their opinion to $\mathcal{C}$ from any other color in $P$, that is,

$$X = \{v \in V : v \notin \mathcal{C}_t \wedge v \in \mathcal{C}_{t+1}\} \ .$$

Clearly, $X$ consists of two disjoint subsets $X = \hat{X} \cup X^*$, defined as

$$\hat{X} = \{v \in V : v \notin (\mathcal{C}_t \cup \mathcal{C}_t^*) \wedge v \in \mathcal{C}_{t+1}\}$$

and

$$X^* = \{v \in V : v \in \mathcal{C}_t^* \wedge v \in \mathcal{C}_{t+1}\} \ .$$

The set $\hat{X}$ consists of all nodes which change their opinion to $\mathcal{C}$ from any other color except for nodes in $\mathcal{C}^*$. The set $X^*$ contains the remaining nodes in $\mathcal{C}^*$

| Set | Process $P$ | Process $P'$ |
|---|---|---|
| $X$ | nodes which change their color to $\mathcal{C}$ | nodes which now belong to $\hat{\mathcal{B}}$ |
| $\hat{X}$ | nodes which change their color to $\mathcal{C}$ except nodes from $\mathcal{C}^*$ | nodes which change their color to $\mathcal{B}$ |
| $X^*$ | nodes from $\mathcal{C}^*$ which change their color to $\mathcal{C}$ | nodes which change their color to $\mathcal{B}$ |
| $Y$ | nodes which change their color away from $\mathcal{C}$ | nodes which no longer belong to $\hat{\mathcal{B}}$ |
| $\hat{Y}$ | nodes which change their color away from $\mathcal{C}$ but not to $\mathcal{C}^*$ | nodes which change their color away from $\hat{\mathcal{B}}$ but not to $\mathcal{B}^*$ |
| $Y^*$ | nodes which change their color from $\mathcal{C}$ to $\mathcal{C}^*$ | nodes which change from $\hat{\mathcal{B}}$ to $\mathcal{B}^*$ |

**Table 17.1:** corresponding sets between processes $P$ and $P'$

which change their opinion to $\mathcal{C}$. Analogously to $X$, let $Y$ be the set of nodes which change their opinion from $\mathcal{C}$ to any other color in $P$, that is,

$$Y = \{v \in V : v \in \mathcal{C}_t \wedge v \notin \mathcal{C}_{t+1}\} \ .$$

Again, we have $Y = \hat{Y} \cup Y^*$ which are defined as

$$\hat{Y} = \{v \in V : v \in \mathcal{C}_t \wedge v \notin (\mathcal{C}_{t+1} \cup \mathcal{C}^*_{t+1})\}$$

and

$$Y^* = \{v \in V : v \in \mathcal{C}_t \wedge v \in \mathcal{C}^*_{t+1}\} \ .$$

We now analyze the behavior of these sets in the coupled process $P'$. The coupling ensures the correspondences described in Table 17.1. We therefore have in $P$

$$c'^{\ (P)} = c^{(P)} + |X| - |Y| \ . \tag{17.3}$$

In $P'$, we first observe that $|\mathcal{B}| = |\hat{\mathcal{B}}| + |\mathcal{B}^*|$ and therefore

$$\begin{aligned} b'^{\ (P')} &\geq b^{(P')} + |\hat{X}| - |\hat{Y}| - (|\mathcal{B}^*| - |X^*|) &(17.4)\\ &\geq |\hat{\mathcal{B}}| + |\mathcal{B}^*| + |\hat{X}| - |\hat{Y}| - |\mathcal{B}^*| + |X^*| \\ &= |\hat{\mathcal{B}}| + |X| - |\hat{Y}| \\ &\geq |\hat{\mathcal{B}}| + |X| - |Y| \\ &= c^{(P)} + |X| - |Y| &(17.5) \end{aligned}$$

where the expression $|\mathcal{B}^*| - |X^*|$ in (17.4) is an upper bound on the number of nodes in $\mathcal{B}^*$ which change their color away from $\mathcal{B}$ to any other color except for $\hat{\mathcal{B}}$. Combining equations (17.3) and (17.5) gives us

$$c'^{\ (P)} \leq b'^{\ (P')}$$

which concludes the proof. □

We now use Lemma 65 and Lemma 66 to show our first main result, Theorem 61.

*Proof.* Let $\mathcal{A} = \mathcal{C}_1$ be the dominant color and $\mathcal{B} = \mathcal{C}_2$ the second largest color. Assume $a - b \geq z \cdot \sqrt{n \log n}$ for a sufficiently large constant $z$. From Lemma 65 we know that $a' - b' \geq (a - b) \cdot (1 + a/4n)$ with high probability. Since $\mathcal{B}$ is the second largest color, we obtain from Lemma 66 for any remaining color $\mathcal{C}_j$ with $j \geq 3$ that with high probability $a' - c'_j \geq a' - b' \geq (a - b) \cdot (1 + a/4n)$. Note that it may very well happen, especially if all colors have the same size except for $\mathcal{A}$, that another color $\mathcal{C}_j$ *overtakes* $\mathcal{B}$. However, the resulting distance between $\mathcal{A}$ and this new second largest color $\mathcal{C}_j$ will be larger than $(a - b) \cdot (1 + a/4n)$ with high probability. Let $a''$ and $b''$ denote the sizes of the color after the round, that is, after the adversary changed the opinion of up to $F$ arbitrary nodes. We have $a'' - b'' \geq a' - b' - 2F \geq (a - b) \cdot (1 + a/4n - 2F/a - b) \geq (a - b) \cdot (1 + a/8n)$, since $F = a(a - b)/8n$.

Taking union bound over all colors, we conclude that the distance between the first color $\mathcal{A}$ and every other color grows in every round by a factor of at least $(1 + a/4n)$ with high probability. Therefore, after $\tau = 4n/a$ rounds, the distance between $\mathcal{A}$ and $\mathcal{B}$ doubles with high probability. Hence, the required time for $\mathcal{A}$ to reach size of at least $(1/2 + \varepsilon_1) \cdot n$ for a constant $\varepsilon_1 > 0$ is bounded by $\mathrm{O}(n/a \cdot \log n)$. This bias is large enough that we assume in the following that all nodes which are not of color $\mathcal{A}$ are of color $\mathcal{B}$ In absence of an adversary, we can see that after additional $\mathrm{O}(\log n)$ rounds every node has with high probability the same color $\mathcal{A}$, see [CER14]. In each individual round, the growth described in Lemma 65 takes place with high probability. Union bound over all $\mathrm{O}(n/a \cdot \log n)$ rounds yields that the protocol indeed converges to $\mathcal{A}$ within $\mathrm{O}(n/a \cdot \log n)$ rounds with high probability. The same analysis of [CER14] can be used even in the presence of an adversary. However, in this case only *almost validity* according to Definition 22 can be reached, since the adversary is allowed to change $F = \mathrm{o}(n)$ nodes per round.

Finally, we argue that the two-choices process trivially fulfills the property *almost agreement* according to Definition 22. Starting from an arbitrary initial distribution of colors, there is in every round a probability which is positive, albeit super-exponentially small in $n$, that all nodes adopt the same round. □

## 17.1 Lower Bounds

In the previous section, we showed that the plurality consensus process converges to $\mathcal{A}$ with high probability, if the initial imbalance $a - b$ is not too small. Precisely, Theorem 61 states that if $a - b \geq z \cdot \sqrt{n \log n}$ for some constant $z$, $\mathcal{A}$ wins with high probability. Conversely, in the following section we examine a lower bound on the initial bias. We will show, as stated in Theorem 68, that for an initial bias $a - b \leq z \cdot \sqrt{n}$ for some constant $z$ we have a constant probability that $\mathcal{B}$ *overtakes* $\mathcal{A}$ in the first round, that is, $\Pr[a' < b'] = \Omega(1)$.

Our proof of Theorem 68 is based on the normal approximation of the binomial distribution. In this context, we adapt Theorem 2 and equation (6.7) from [Fel68] as stated in the following theorem.

**Theorem 67** (DeMoivre-Laplace limit theorem [Fel68])**.** *Let $X$ be a random variable with binomial distribution $X \sim B(N,p)$. It holds for any $x > 0$ with $x = \mathrm{o}\left(N^{1/6}\right)$ that*

$$\Pr\left[X \geq \mathrm{E}[X] + x \cdot \sqrt{\mathrm{Var}[X]}\right] = \frac{1}{\sqrt{2\pi} \cdot x} \cdot \exp\left(-x^2 2\right) \pm \mathrm{o}(1) \ .$$

We now use Theorem 67 and prove Theorem 68 which states that There exists an initial color assignment for which $a = b + z' \cdot \sqrt{n}$ but color $\mathcal{B}$ wins with constant probability even in absence of an adversary.

**Theorem 68** (Lower Bound on the Initial Bias)**.** *For any $k \leq \sqrt{n}$ and constant $z'$ there exists an initial assignment of colors to nodes for which $a = b + z' \cdot \sqrt{n}$ but $\Pr[a' < b'] = \Omega(1)$ even in absence of an adversary.*

*Proof.* Let $z = z'/2$ and $n' = \frac{n-k+2}{2}$. Assume that we have the following initial color distribution among the nodes.

$$(c_1, c_2, c_3, \ldots, c_k) = (\lfloor n' \rfloor + \lfloor z \cdot \sqrt{n} \rfloor, \lceil n' \rceil - \lfloor z \cdot \sqrt{n} \rfloor, 1, \ldots, 1).$$

Clearly, $\sum_{\mathcal{C}_j} c_j = n$. In the following we will omit the floor and ceiling functions for simplicity and readability reasons. First, we start by giving an upper bound on the number of nodes which change their color away from $\mathcal{B}$. Now recall that $f_{\mathcal{B}\overline{\mathcal{B}}}$ follows a binomial distribution $f_{\mathcal{B}\overline{\mathcal{B}}} \sim B(b, \sum_{Cj \neq B} c_j^2/n^2)$ with expected value

$$\begin{aligned}
\mathrm{E}[f_{\mathcal{B}\overline{\mathcal{B}}}] &= b \cdot \frac{a^2 + k - 2}{n^2} \\
&= (n' - z \cdot \sqrt{n}) \cdot \frac{(n' + z \cdot \sqrt{n})^2 + k - 2}{n^2} \\
&\leq \frac{(n' + z \cdot \sqrt{n})^3 + k - 2}{n^2} \\
&\leq \frac{n}{8} + 4z\sqrt{n} \ .
\end{aligned}$$

Applying Chernoff bounds to $f_{\mathcal{B}\overline{\mathcal{B}}}$ gives us

$$\Pr\left[f_{\mathcal{B}\overline{\mathcal{B}}} \geq \left(1 + \sqrt{3/\,\mathrm{E}[f_{\mathcal{B}\overline{\mathcal{B}}}]}\right) \cdot \mathrm{E}[f_{\mathcal{B}\overline{\mathcal{B}}}]\right] \leq 1/e \ . \tag{17.6}$$

That is, with constant probability at least $1 - 1/e$ we have

$$\begin{aligned}
f_{\mathcal{B}\overline{\mathcal{B}}} &\leq \left(1 + \sqrt{3/\,\mathrm{E}[f_{\mathcal{B}\overline{\mathcal{B}}}]}\right) \cdot \mathrm{E}[f_{\mathcal{B}\overline{\mathcal{B}}}] \\
&\leq \frac{n}{8} + 4z\sqrt{n} + \sqrt{3 \cdot \mathrm{E}[f_{\mathcal{B}\overline{\mathcal{B}}}]} \\
&\leq \frac{n}{8} + (4z + 1) \cdot \sqrt{n} \ .
\end{aligned}$$

Secondly, we give the following lower bound on the number of nodes which change their color from $\mathcal{A}$ to $\mathcal{B}$. Again, the random variable $f_{\mathcal{AB}}$ for the flow from $\mathcal{A}$ to $\mathcal{B}$ has a binomial distribution $f_{\mathcal{AB}} \sim B(a, b^2/n^2)$ with expected value

$$\begin{aligned}
\mathrm{E}[f_{\mathcal{AB}}] &= (n' + z \cdot \sqrt{n}) \cdot \frac{(n' - z \cdot \sqrt{n})^2}{n^2} \\
&\geq \frac{(n' - z \cdot \sqrt{n})^3}{n^2} \\
&\geq \frac{(n/2 - (z + 1/2)\sqrt{n})^3}{n^2} \\
&\geq \frac{n}{8} - 4z\sqrt{n}
\end{aligned}$$

and variance

$$\begin{aligned}
\mathrm{Var}[f_{\mathcal{AB}}] &= \mathrm{E}[f_{\mathcal{AB}}] \cdot \left(1 - \frac{(n' - z \cdot \sqrt{n})^2}{n^2}\right) \\
&\geq \frac{n}{9} \cdot \frac{1}{2} = \frac{n}{18} \ .
\end{aligned}$$

We now apply Theorem 67 to $f_{\mathcal{AB}}$. Let $x$ be a constant which we define as $x = \frac{\sqrt{18}}{2}(18z + 4)$. We derive

$$\Pr\left[f_{\mathcal{AB}} \geq \mathrm{E}[f_{\mathcal{AB}}] + x \cdot \sqrt{\mathrm{Var}[f_{\mathcal{AB}}]}\right] = \frac{1}{\sqrt{2\pi} \cdot x} \exp\left(-x^2/2\right) \pm \mathrm{o}(1) = \Omega(1) \ .$$

That is, we have with constant probability

$$f_{\mathcal{AB}} \geq \mathrm{E}[f_{\mathcal{AB}}] + x \cdot \sqrt{\mathrm{Var}[f_{\mathcal{AB}}]} \geq \frac{n}{8} - 4z\sqrt{n} + x \cdot \sqrt{\frac{n}{18}} \ . \tag{17.7}$$

Finally, assume that in the worst case every node of colors $\mathcal{C}_3, \ldots, \mathcal{C}_k$ changes to $\mathcal{A}$ but not a single node changes away from $\mathcal{A}$ to these colors $\mathcal{C}_3$ to $\mathcal{C}_k$. Observe that $f_{\mathcal{B}\overline{\mathcal{B}}}$ is an upper bound on $f_{\mathcal{BA}}$. Therefore,

$$\begin{aligned}
a' - b' &\leq (a + (k-2) + f_{\mathcal{BA}} - f_{\mathcal{BA}}) - (b + f_{\mathcal{AB}} - f_{\mathcal{B}\overline{\mathcal{B}}}) \\
&\leq a - b + (k-2) + 2f_{\mathcal{B}\overline{\mathcal{B}}} - 2f_{\mathcal{AB}} \\
&\leq 2z \cdot \sqrt{n} + (k-2) + 2f_{\mathcal{B}\overline{\mathcal{B}}} - 2f_{\mathcal{AB}} \\
&\leq (2z+1) \cdot \sqrt{n} + 2f_{\mathcal{B}\overline{\mathcal{B}}} - 2f_{\mathcal{AB}} \ .
\end{aligned}$$

We plug in (17.6) and (17.7) to bound the random variables $f_{\mathcal{AB}}$ and $f_{\mathcal{B}\overline{\mathcal{B}}}$ and obtain with constant probability

$$\begin{aligned}
a' - b' &\leq (2z+1) \cdot \sqrt{n} + 2\left(\frac{n}{8} + (4z+1)\sqrt{n}\right) - 2\left(\frac{n}{8} - 4z\sqrt{n} + x \cdot \sqrt{\frac{n}{18}}\right) \\
&= (2z + 1 + 8z + 2 + 8z - 2x/\sqrt{18}) \cdot \sqrt{n} \\
&= (18z + 3 - 2x/\sqrt{18}) \cdot \sqrt{n}
\end{aligned}$$

which gives us for $x = \frac{\sqrt{18}}{2}(18z + 4)$

$$a' - b' < 0 \ .$$

Therefore, we have $\Pr[a' < b'] = \Omega(1)$ and thus we conclude that color $\mathcal{B}$ prevails with constant probability. □

## Lower Bound on the Run Time

**Theorem 69** (Lower Bound on the Run Time)**.** *Assume the initial bias is exactly $z\sqrt{n \log n}$ for some constant $z$. The number of rounds required for plurality consensus process defined in Algorithm 17.1 to converge is at least $\Omega(n/a + \log n)$ with constant probability even in absence of an adversary.*

*Proof.* Let $a(t)$ denote the size of color $\mathcal{A}$ in round $t$. Assume $\mathcal{A}$ is the largest color of initial size $a(0) = n/k + z \cdot \sqrt{n \log n}$. Furthermore, assume that $k \geq 3 \cdot z$. We show by induction on the rounds that $a(t) \leq a(0) \cdot (1 + 3 \cdot a(0)/n)^t$ for $1 \leq t \leq n/(10 \cdot a(0))$ with probability $1 - t/n$. First we note that

$$\begin{aligned} a(t) &\leq a(0) \cdot \left(1 + 3 \cdot \frac{a(0)}{n}\right)^t \\ &\leq a(0) \cdot \left(1 + 3 \cdot \frac{a(0)}{n}\right)^{n/(10 \cdot a(0))} \\ &\leq a(0) \cdot \exp(1/2) \\ &\leq 2 \cdot a(0) \end{aligned} \tag{17.8}$$

and

$$a(t) \geq a(0) \ . \tag{17.9}$$

We now prove the induction claim. The base case holds trivially. Consider step $t + 1$. By induction hypothesis we have with probability at least $1 - t/n$ that $a(t) \leq a(0) \cdot (1 + 3 \cdot a(0)/n)^t$. Note that we have with high probability

$$\begin{aligned} a(t+1) &\leq a(t) + f_{\overline{\mathcal{A}}\mathcal{A}} \\ &\leq a(t) + \left(1 + \frac{\sqrt{3 \log n}}{\sqrt{\mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}}]}}\right) \cdot \mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}}] \ , \end{aligned}$$

where the latter inequality follows by Chernoff bounds. Using (17.8) and (17.9), we derive

$$\begin{aligned} a(t+1) &\le a(t) + \left(1 + \frac{\sqrt{3\log n}}{\sqrt{a(t)^2/(2\cdot n)}}\right)\frac{a(t)^2}{n} \\ &\le a(t) + \left(1 + \frac{\sqrt{3\log n}}{\sqrt{a(0)^2/(2\cdot n)}}\right)\frac{a(t)^2}{n} \\ &\le a(t) + \frac{3}{2}\cdot\frac{a(t)^2}{n} \\ &= a(t)\cdot\left(1+\frac{3}{2}\cdot\frac{a(t)}{n}\right) \\ &\le a(t)\cdot\left(1+\frac{3\cdot a(0)}{n}\right) \ . \end{aligned}$$

From the induction hypothesis we therefore obtain

$$\begin{aligned} a(t+1) &\le a(0)\cdot\left(1+\frac{3\cdot a(0)}{n}\right)^t\cdot\left(1+\frac{3\cdot a(0)}{n}\right) \\ &= a(0)\cdot\left(1+\frac{3\cdot a(0)}{n}\right)^{t+1} \ . \end{aligned}$$

Using a union bound to account for all errors, we derive that with probability at least $1-(t+1)/n$ we have $a(t+1)\le a(0)\cdot(1+3\cdot a(0)/n)^{t+1}$, which completes the proof of the induction and proves the lower bound of $\Omega(n/a)$.

In the remainder we establish the bound $\Omega(\log n)$. Assume only two colors $\mathcal{A}$ and $\mathcal{B}$, where $\mathcal{A}$ is the largest color of initial size $a(0)=n/2+\sqrt{n}\log n$. We show by induction on the rounds that $a(t)\le a(0)+6^t\sqrt{n}\log n$ for $1\le t\le \log n/20$ with probability $1-2t/n$. First we note that

$$a(t)\le a(0)+6^t\sqrt{n}\log n\le n/2+n^{5/6}<n$$

and

$$a(t)\ge a(0) \ .$$

We now prove the induction claim. The base case holds trivially. Consider step $t+1$. By induction hypothesis we have with probability at least $1-2t/n$ that $a(t)\le a(0)+6^t\sqrt{n}\log n$. We have, writing $a=a(t)$ and $\beta=6^t\sqrt{n}\log n$.

$$\begin{aligned} n^2\cdot \mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}}-f_{\mathcal{A}\overline{\mathcal{A}}}] &= (n-1)a^2 - a\cdot(n-a)^2 = (n-a)a(2a-n) \\ &\le n/2\cdot a\cdot 2\beta = n\cdot\beta(n+\beta) = n^2\cdot\beta + n\cdot\beta^2 \ . \end{aligned}$$

Similarly as before, we obtain by Chernoff bounds, that with high probability

$$\begin{aligned}
a(t+1) - a(t) &= f_{\overline{\mathcal{A}}\mathcal{A}} - f_{\mathcal{A}\overline{\mathcal{A}}} \\
&\leq \left(1 + \frac{\sqrt{3\log n}}{\sqrt{\mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}}]}}\right)\mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}}] - \left(1 - \frac{\sqrt{3\log n}}{\sqrt{\mathrm{E}[f_{\mathcal{A}\overline{\mathcal{A}}}]}}\right)\mathrm{E}[f_{\mathcal{A}\overline{\mathcal{A}}}] \\
&\leq \mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}} - f_{\mathcal{A}\overline{\mathcal{A}}}] + 2\sqrt{3\log n} \cdot \sqrt{\mathrm{E}[f_{\overline{\mathcal{A}}\mathcal{A}}]} \\
&\leq \beta + \beta^2/n + 2\sqrt{3\log n} \cdot \sqrt{n} \leq 3\beta \enspace .
\end{aligned}$$

From the induction hypothesis we therefore obtain

$$\begin{aligned}
a(t+1) &\leq a(0) + 6^t\sqrt{n}\log n + 3\beta \\
&\leq a(0) + 6^{t+1}\sqrt{n}\log n \enspace ,
\end{aligned}$$

which completes the induction and yields the lower bound of $\Omega(\log n)$. □

## 17.2 Comparison with the 3-Majority Process

In this section we elaborate on the difference between the two-choices process and the 3-majority rule [BCN+14], where in the latter each node pulls the opinion of three random neighbors and adopts the majority opinion among those three, breaking ties uniformly at random. As mentioned before, the 3-majority process of [BCN+14] uses $\mathrm{O}(\log k)$ memory bits and the authors prove a tight run time of $\Theta(k \cdot \log n)$ for this protocol, given a sufficiently high bias $c_1 - c_2$. Moreover, they show that if the bias is only of order $\sqrt{kn}$, then with constant probability the difference $c_1 - c_2$ decreases. This is fundamentally different to the two-choices process, where we only require a bias of $\Omega(\sqrt{n\log n})$.

The reasons are the following. First, the variance in the 3-majority process can be of orders of magnitude larger and second, the expected increase in the difference between the largest and second largest color in the 3-majority process is only of order of the variance. As for the variance, consider an initial setting where all colors are of sublinear size and $\mathcal{A}$ and $\mathcal{B}$ are larger than all other colors, such that

$$\mathrm{o}(n) = a = b + c\sqrt{n\log n} > c_j + c\sqrt{n\log n}$$

and

$$c_j = (n - b - a)/(k-2)$$

for all $2 \leq j \leq k$ with $k = n^{\varepsilon}$ for constants $\varepsilon$ and $c$. Observe that the expected number of switches differs largely. In the two-choices process it is very unlikely for a node to pick a node of the same color twice and the probability of switching is $\mathrm{o}(1)$. In contrast to this, the probability of switching in the 3-majority process is $1 - \mathrm{o}(1)$.

More illustrative, consider the number of switches to color $\mathcal{B}$. By Lemma 2.1 of [BCN+14], the probability that a node switches in the 3-majority process to color $\mathcal{B}$ is $p \in [b/n, 2b/n]$ and the variance becomes $n \cdot p \cdot (1-p) \geq b/2$. However, in the two-choices process, the probability of switching to $\mathcal{B}$ is $q = b^2/n^2$ and the variance is thus a at most $n \cdot q \cdot (1-q) \leq n \cdot q = b^2/n$, which is considerably smaller than $b/2$. This high variance paired with the small expected increase in the difference between $\mathcal{A}$ and $\mathcal{B}$ becomes easily fatal. Again, by Lemma 2.1 of [BCN+14], one can verify that $\mathrm{E}[a' - b'] \leq a - b + (a^2 - b^2)/n$. Now we have $\Pr[a' \leq \mathrm{E}[a']] = \Omega(1)$ and, using the large variance, we obtain from Theorem 67 that

$$\begin{aligned}\Pr\Big[b' \geq b + (a^2 - b^2)/n \,|\, a' \leq \mathrm{E}[a']\Big] &\geq \Pr\Big[b' \geq b + (a^2 - b^2)/n\Big] \\ &\geq \Pr[b' \geq \mathrm{E}[b'] + \mathrm{Var}[b']] = \Omega(1) \ .\end{aligned}$$

Thus the distance between $\mathcal{A}$ and $\mathcal{B}$ decreases with constant probability, that is, $\Pr[a' - b' < a - b] = \Omega(1)$. In comparison to this, we have seen in Chapter 17 that in the given setting the distance between $\mathcal{A}$ and $\mathcal{B}$ in the two-choices process increases with high probability.

# 18

# One Bit of Memory

In this chapter we investigate the `OneBit` protocol which combines the guarantees of the two-choices process to reach plurality consensus with the speed of broadcasting. The protocol consists of $\Theta(\log(n/a) + \log\log n)$ phases which in turn consist of two sub-phases, one round of the Two-Choices process and multiple rounds of the so-called Bit-Propagation sub-phase. In the latter Bit-Propagation sub-phase, each node that changed its opinion during the preceding two-choice step broadcasts its new opinion.

More precisely, we consider the modified model where each node is allowed to store and transmit one additional bit. This bit is set to True if and only if a node changed its opinion in the Two-Choices sub-phase. In the Bit-Propagation sub-phase, each node $u$ samples nodes randomly until a node $v$ with a bit set to True is found. Then $u$ adopts $v$'s opinion and sets its own bit to True, which means that subsequently any node sampling $u$ will set their bit directly.

The first sub-phase ensures that in a round $t$ the number of nodes holding opinion $\mathcal{A}$ and having their bit set to True is concentrated around $a_{t-1}^2/n$. After the Bit-Propagation sub-phase, all nodes will have their bit set, and the distribution and the size of $\mathcal{A}$'s support is concentrated around $a^2/x(1)$, where $x(1)$ is the total number of bits set after the Two-Choices sub-phase. Moreover, we show that no other color grows faster. In fact, we show that the distance between $\mathcal{A}$ and any opinion $\mathcal{C}_j \neq A$ increases quadratically, that is, $a'/c'_j \geq (1 - \mathrm{o}(1)) \cdot a^2/c_j^2$. Due to the quadratic growth in the distance between $\mathcal{A}$ and every other opinion, the number of phases required is only of order $\Theta(\log(n/a) + \log\log n)$. The process runs in multiple phases of length $\Theta(\log k + \log\log n)$ each, therefore we assume that every node is aware of (upper bounds on) $n$ and $k$. The process is formally defined in Algorithm 18.1.

If we assume that each node has knowledge of $n/a$, the run time can be further reduced to $\mathrm{O}((\log(c_1/(c_1 - c_2)) + \log\log n) \cdot (\log(n/a) + \log\log n))$, given $n/a$ is smaller than $k^{\mathrm{o}(1)}$. We start our analysis with Lemma 70 where we derive a lower bound on the number of bits set during the two-sample step. We

```
Algorithm memory($G = (V, E)$, color : $V \to C$, bit : $V \to \{\text{True}, \text{False}\}$)
  for phase $s = 1$ to $\ell \log(U) + \log\log n$ do
    at each node $v$ do in parallel                    /* two-choices (Round 1) */
      let $u_1, u_2 \in N(v)$ uniformly at random;
      if color($u_1$) = color($u_2$) then
        color($v$) ← color($u_1$);
        bit($v$) ← True ;
      else
        bit($v$) ← False ;
    for round $t = 2$ to $2\log k + 2\log\log n$ do       /* bit-propagation */
      at each node $v$ do in parallel
        let $u \in N(v)$ uniformly at random;
        if bit($u$) then
          color($v$) ← color($u$);
          bit($v$) ← True ;
```

**Algorithm 18.1:** distributed voting protocol OneBit with one bit of memory. The variable $\ell$ is a large constant and $U$ is an upper bound on $c_1/(c_1 - c_2)$. Since the process runs in multiple phases of length $\Theta(\log k + \log\log n)$ each, we assume that every node has knowledge of $\ell \cdot U$, $n$ and $k$.

will then use the results by Karp et al. [KSSV00] to argue that after the bit-propagation rounds the number of bits set is $n$ with high probability, that is, the total number of bits set grows until eventually every node has its bit set. Finally, we will prove in Lemma 74 that the relative number of bits set for *large* colors remains close to the initial (relative) value during the propagation steps. Together with the growth of the total number of set bits, this leads to a growth of the imbalance towards $\mathcal{A}$ by at least a constant factor during each phase.

We will use $x^{(i)}(t)$ to denote the random variable for the total number of nodes which have their bit set in a round $t$ of phase $i$. When it is clear from the context, we simply use the notation $x(t)$. Accordingly, $x^{(i)}(1)$ is the number of bits set after the two-choices round of phase $i$. Additionally, we will use $x_j^{(i)}(t)$ to denote the number of nodes of color $\mathcal{C}_j$ which have their bit set in a round $t$. Similarly as before, we simply write $x_j(t)$ when the phase is clear from the context. Furthermore, when analyzing the growth in $x^{(i)}(t)$ and $x_j^{(i)}(t)$ with respect to $x^{(i)}(t-1)$ and $x_j^{(i)}(t-1)$, we will assume that $x^{(i)}(t-1)$ and $x_j^{(i)}(t-1)$ are fixed.

In the following lemmas we analyze an arbitrary but fixed phase.

**Lemma 70.** *After the two-choices round, at least $\Omega(n/k)$ bits are set with high probability.*

*Proof.* The probability for one node to open connections to two nodes of the same color is $p_{\text{two-choices}} = \sum_{\mathcal{C}_j} \frac{c_j^2}{n^2}$. This probability is minimized if all colors are of the same size $n/k$ and therefore $p_{\min} = \frac{1}{n^2} \cdot \sum_{\mathcal{C}_j} \frac{n^2}{k^2} = \frac{1}{k}$. Since all nodes open connections independently, the random variable for the number of bits set after the two-choices round, $x(1)$, has a binomial distribution with expected value at least $\mathrm{E}[x(1)] \geq n/k$. Applying Chernoff bounds to $x(1)$ gives us

$$\Pr\left[x(1) \leq \left(1 - 2\sqrt{\frac{k \log n}{n}}\right)\frac{n}{k}\right] \leq \exp\left(-\frac{4kn \log n}{2kn}\right) = n^{-2} \quad . \qquad \square$$

From the lemma above we obtain that we have at least $x(1) = n/k \cdot (1 - \mathrm{o}(1)) = \Omega(n/k)$ bits set after the first round with high probability. We now investigate the growth of $x(t)$ in the following rounds.

**Lemma 71** (Pull Rumor Spreading [KSSV00])**.** *With high probability, after at most $T = \mathrm{O}(\log k + \log\log n)$ bit propagation rounds, we have $x(T) = n$. Furthermore, it holds that $1 \leq x(t+1)/x(t) \leq 2 + \mathrm{o}(1)$ and there exists a monotonically increasing function $f : \mathbb{N} \to \mathbb{N}$ such that $x(t) = x(1) \cdot f(t) \cdot \left(1 \pm 1/n^{\Omega(1)}\right)$, with high probability.*

In the following, we focus on the colors that are present among those nodes which have their bit set. We start by showing that the initial number of bits is well-concentrated around the expectation for colors which are *large enough*.

**Lemma 72.** *For any color $\mathcal{C}_j$ with $c_j = \Omega(\sqrt{n \log n})$ the number of nodes of color $\mathcal{C}_j$ which have their bit set after the two-choices round is concentrated around the expected value, that is,*

$$x_j(1) = \mathrm{E}[x_j(1)]\left(1 \pm \mathrm{O}\left(\sqrt{\log n}/\sqrt{\mathrm{E}[x_j(1)]}\right)\right)$$

*with high probability. If $c_j = \mathrm{O}(\sqrt{n \log n})$, then $x_j(1) = \mathrm{O}(\log n)$ with high probability.*

*Proof.* Let $\mathcal{C}_j$ be an arbitrary but fixed color with $c_j > 3\sqrt{n \log n}$. The number of nodes of color $\mathcal{C}_j$ which have their bit set after the two-choices round has a binomial distribution $x_j(1) \sim B(n,\ c_j^2/n^2)$ with expected value $\mathrm{E}[x_j(1)] = c_j^2/n > 9 \log n$. We apply Chernoff bounds to $x_j(1)$ and obtain

$$\Pr\left[|x_j(1) - \mathrm{E}[x_j(1)]| > 3\sqrt{\frac{\log n}{\mathrm{E}[x_j(1)]}} \cdot \mathrm{E}[x_j(1)]\right] \leq n^{-2} \quad .$$

That is, we have $|x_j(1) - \mathrm{E}[x_j(1)]| \leq 3\sqrt{\log n \cdot \mathrm{E}[x_j(1)]}$ with high probability. The second statement can be shown in an analogous way. $\square$

**Lemma 73.** *Let $\mathcal{C}_j$ be a color with at least $x_j(t) = \Omega(\log n)$ bits set in a round $t$. Assume $x(t)$ and $x_j(t)$ are given and they are concentrated around their mean. Then we have*

$$\mathrm{E}[x_j(t+1)|x(t), x_j(t)] = x_j(t) + \frac{n - x(t)}{n} \cdot x_j(t) \enspace .$$

*Furthermore, the number of nodes of color $\mathcal{C}_j$ which have their bit set in round $t+1$ is with high probability concentrated around the expected value such that*

$$x_j(t+1) = \mathrm{E}[x_j(t+1)|x_j(t), x(t)] \cdot \left(1 \pm \mathrm{O}\left(\frac{\sqrt{\log n}}{\sqrt{\mathrm{E}[x_j(t+1)|x(t), x_j(t)]}}\right)\right) \enspace .$$

*Proof.* In the following, we will use $\mathrm{bit}_v(t)$ to denote the value of the bit of a node $v$ in a round $t$, where the value can be either True or False. We consider the probability that $v$ has color $\mathcal{C}_j$ in round $t+1$, given that $v$ has its bit set in round $t+1$. We have

$$\Pr[v \in \mathcal{C}_j(t+1) | \, \mathrm{bit}_v(t+1) = \textsc{True}, x_j(t), x(t)] = x_j(t)/x(t) \enspace ,$$

since

$$\begin{aligned}
&\Pr[v \in \mathcal{C}_j(t+1) | \, \mathrm{bit}_v(t+1) = \textsc{True}, x_j(t), x(t)] \\
&\quad = \frac{\Pr[v \in \mathcal{C}_j(t+1) \wedge \mathrm{bit}_v(t+1) = \textsc{True} | x_j(t), x(t)]}{\Pr[\mathrm{bit}_v(t+1) = \textsc{True} | x_j(t), x(t)]} \\
&\quad = \frac{\overbrace{\frac{x_j(t)}{n}\left(\frac{n-x(t)}{n}\right)}^{(i)} + \overbrace{\frac{x_j(t)}{n}}^{(ii)}}{\underbrace{\frac{x(t)}{n}}_{(iii)} + \underbrace{\left(1 - \frac{x(t)}{n}\right)\frac{x(t)}{n}}_{(iv)}} = \frac{x_j(t)}{x(t)} \cdot \frac{1 - \frac{x(t)}{n} + 1}{1 + 1 - \frac{x(t)}{n}} \enspace .
\end{aligned}$$

In above equation, the probability for a node to have color $\mathcal{C}_j$ and the bit set in round $t+1$ is computed as follows.

(i) is the probability that a node has color $\mathcal{C}_j$ and the bit set at time $t$ and selects a node without a bit set
(ii) is the probability that a node chooses another node which has color $\mathcal{C}_j$ and the bit set
(iii) is the probability for choosing a node with a set bit
(iv) is the probability for choosing a node without the bit set which selects another node with the bit set

Consequently, the number of nodes which have color $\mathcal{C}_j$ in the next round has expected value $\mu = \mathrm{E}[x_j(t+1)|x(t+1), x_j(t), x(t)] = x_j(t) \cdot x(t+1)/x(t)$. We apply Chernoff bounds to $x_j(t+1)$ and obtain

$$\Pr\left[|x_j(t+1) - \mu| > 3\sqrt{\frac{\log n}{\mu}} \cdot \mu \,\middle|\, x_j(t), x(t), x(t+1)\right] \le n^{-2} \enspace .$$

Assuming $x(t)$ fulfills Lemma 70, we have [KSSV00]

$$x(t+1) = \mathrm{E}[x(t+1)|x(t)] \cdot \left(1 \pm \mathrm{O}\left(\sqrt{k \log n}/\sqrt{n}\right)\right) \ ,$$

and therefore we obtain the lemma. □

**Lemma 74.** *Let $\mathcal{A}$ be the dominant color of size $a$ and $\mathcal{B}$ the second largest color of size $b$. Let $a'$ and $b'$ be the number of nodes of colors $\mathcal{A}$ and $\mathcal{B}$, respectively, after the bit-propagation phase. Let $T = 2(\log k + \log\log n)$. Given $x(1)$ and assuming it is concentrated around the expected value, we have with high probability after $T$ bit-propagation rounds*

$$a' \geq \frac{a^2}{x(1)} \cdot \left(1 - \mathrm{O}\left(\frac{T \cdot \sqrt{n \log n}}{a}\right)\right)$$

*and*

$$b' \leq \frac{b^2}{x(1)} \cdot \left(1 + \mathrm{O}\left(\frac{T \cdot \sqrt{n \log n}}{b}\right)\right) + \log^2 n \ .$$

*Furthermore, for any other color $\mathcal{C}_j$ of size $c_j$ it holds with high probability that*

$$c_j' \leq \frac{c_j^2}{x(1)} \cdot \left(1 + \mathrm{O}\left(\frac{T \cdot \sqrt{n \log n}}{c_j}\right)\right) + k^2 \cdot \log^4 n \ .$$

*Proof.* Let $a_i = x_1(i)$ be a sequence of random variables for the number of nodes of color $\mathcal{A}$ which have their bit set in round $i$. In the following proof, whenever we condition on $a_j$ or $x(j)$ for any $j$, we assume that they are concentrated around their mean according to Lemma 71, Lemma 72, and Lemma 73.

According to Lemma 73 we know that

$$\mathrm{E}[a_{i+1}|a_i, x(i+1), x(i)] = \frac{x(i+1)}{x(i)} \cdot a_i \ .$$

Note that $\mathrm{E}[a_{i+1}|a_i] \geq a_i$. Therefore we have

$$\Pr\left[a_{i+1} < \frac{x(i+1)}{x(i)} \cdot a_i \cdot \left(1 - \frac{3\sqrt{\log n}}{\sqrt{a_i}}\right) \middle| a_i, x(i-1), x(i)\right] \leq n^{-2} \ .$$

The total number of bits set in the round $i+1$, given the total number of bits in round $i$, is independent of the color distribution among these nodes in round $i$, that is, for any $\beta \leq \gamma$ it holds for any $\alpha$ that

$$\Pr[x(i+1) = \alpha | x_j(i) = \beta, x(i) = \gamma] = \Pr[x(i+1) = \alpha | x(i) = \gamma] \ .$$

We therefore have for any $\tau > i$

$$\Pr\left[a_{i+1} < \frac{x(i+1)}{x(i)} \cdot a_i \cdot \left(1 - \frac{3\sqrt{\log n}}{\sqrt{a_i}}\right) \middle| a_i, x(1), \ldots, x(\tau)\right] \leq n^{-2} \ .$$

The equation above means that the distribution of the colors among the nodes with the bit set at time $i+1$, given $x(1)\dots x(i+1)$, is independent of the number of nodes with the bit set at times $i+2,\dots,\tau$.

Recall that, given $a_1$, $a_i = \Omega(a_1)$ with high probability and therefore we have for given $a_1$, $a_i$, $x(i-1)$, $x(i)$, and a constant $\zeta$ with high probability

$$a_{i+1} \geq \frac{x(i+1)}{x(i)} \cdot a_i \cdot \left(1 - \zeta \cdot \frac{\sqrt{\log n}}{\sqrt{a_1}}\right) . \tag{18.1}$$

Define $T = \mathrm{O}(\log(n/a) + \log\log n)$ such that $x(T) = n$ with high probability according to [KSSV00]. We now show by induction that, given $a_1$, $x(1),\dots,x(T)$, and a constant $\zeta$,

$$a_T \geq \frac{x(T)}{x(1)} \cdot a_1 \cdot \left(1 - \zeta \cdot \frac{\sqrt{\log n}}{\sqrt{a_1}}\right)^T \tag{18.2}$$

with high probability. The base case for round $t = 1$ obviously holds. For the step from $t$ to $t+1$ we use (18.1) as follows.

$$\begin{aligned} a_{t+1} &\overset{(18.1)}{\geq} \frac{x(t+1)}{x(t)} \cdot a_t \cdot \left(1 - \zeta \cdot \frac{\sqrt{\log n}}{\sqrt{a_1}}\right) \\ &\overset{\mathrm{IH}}{\geq} \frac{x(t+1)}{x(t)} \cdot \frac{x(t)}{x(1)} \cdot a_1 \cdot \left(1 - \zeta \cdot \frac{\sqrt{\log n}}{\sqrt{a_1}}\right)^t \cdot \left(1 - \zeta \cdot \frac{\sqrt{\log n}}{\sqrt{a_1}}\right) \\ &\geq \frac{x(t+1)}{x(1)} \cdot a_1 \cdot \left(1 - \zeta \cdot \frac{\sqrt{\log n}}{\sqrt{a_1}}\right)^{t+1} \end{aligned}$$

This concludes the induction. We apply the Bernoulli inequality to (18.2) and obtain

$$a_T \geq \frac{x(T)}{x(1)} \cdot a_1 \cdot \left(1 - \zeta \cdot \frac{T \cdot \sqrt{\log n}}{\sqrt{a_1}}\right) . \tag{18.3}$$

A similar upper bound can be computed for any *large* color. Let $\mathcal{B} = \mathcal{C}_2$ be the second largest color. For $b_i = x_2(i)$ we obtain with high probability

$$b_T \leq \frac{x(T)}{x(1)} \cdot b_1 \cdot \left(1 + \zeta \cdot \frac{T \cdot \sqrt{\log n}}{\sqrt{b_1}}\right) .$$

In the following we analyze how the gap between $\mathcal{A}$ and $\mathcal{B}$ changes during one phase. We use the result from Lemma 72 for $a_1$ in (18.3) and obtain

$$\begin{aligned} a' &\geq \frac{n}{x(1)} \cdot a_1 \cdot \left(1 - \zeta \cdot \frac{T \cdot \sqrt{\log n}}{\sqrt{a_1}}\right) \\ &\geq \frac{n}{x(1)} \cdot \frac{a^2}{n} \cdot \underbrace{\left(1 - \zeta \cdot \frac{T \cdot \sqrt{\log n}}{\sqrt{a_1}}\right)}_{\text{(I)}} \cdot \underbrace{\left(1 - \frac{3\sqrt{\log n} \cdot \sqrt{n}}{a}\right)}_{\text{(II)}} , \end{aligned}$$

where the second expression in parentheses, (II), is asymptotically dominated by the first one, (I). Therefore, there is a $\zeta'$ such that

$$a' \geq \frac{a^2}{x(1)} \cdot \left(1 - \zeta' \cdot \frac{T \cdot \sqrt{\log n}}{\sqrt{a_1}}\right) . \tag{18.4}$$

As before, we can apply a similar calculation for the upper bound of any color $\mathcal{B}$ as long as $b$ is large enough, that is, $b \geq \sqrt{n \log n}$. We therefore obtain with high probability

$$b' \leq \frac{b^2}{x(1)} \cdot \left(1 + \zeta' \cdot \frac{T \cdot \sqrt{\log n}}{\sqrt{b_1}}\right) . \tag{18.5}$$

Finally, the same calculation can be also applied to any other color $\mathcal{C}_j$ of size $c_j$. However, if $c_j$ is between $\sqrt{n}/\log n$ and $\sqrt{n} \cdot \log n$, then we observe that after the two-choices round we have at most $\mathrm{O}\left(\log^2 n\right)$ bits set for $\mathcal{C}_j$. Since in any step the color can increase by at most a factor of $2 \cdot \left(1 + \mathrm{o}\left(1/\log^2 n\right)\right)$ with high probability, we have in the end at most $\mathrm{O}\left(k^2 \cdot \log^4 n\right)$ nodes of color $\mathcal{C}_j$. Since $k \leq n^{\epsilon}$, in the next two-choices phase this color will disappear with high probability.

Taking all contributions into consideration, we observe that there always exists a constant $\zeta'$ such that (18.4) and (18.5) are satisfied. $\square$

We are now ready to put all pieces together and prove our main theorem, Theorem 62, which is restated as follows.

**Theorem 62.** *Let $G = K_n$ be the complete graph with $n$ nodes. Let $k = \mathrm{O}(n^{\varepsilon})$ be the number of opinions for some small constant $\varepsilon > 0$. The plurality consensus process* `OneBit` *defined in Algorithm 18.1 on $G$ converges within*

$$\mathrm{O}((\log(c_1/(c_1 - c_2)) + \log\log n) \cdot (\log k + \log\log n))$$

*time steps to $\mathcal{A}$, with high probability, if $c_1 - c_2 \geq z \cdot \sqrt{n \log^3 n}$ for some constant $z$.*

*Proof.* Assume $x(1)$ is given and concentrated around its expected value. In the following proof, we assume that $b \geq \sqrt{n \log n}$. Recall that in the statement of Theorem 62 we assume $a - b \geq z \cdot \sqrt{n \log^3 n}$. Let $T = 2(\log k + \log\log n)$. From the bounds on $a'$ and $b'$ from Lemma 74 we obtain the following inequality, which holds with high probability.

$$\begin{aligned} a' - b' &\geq \frac{a^2 - b^2}{x(1)} - \frac{\zeta \cdot T \cdot \sqrt{\log n}}{x(1)} \cdot \left(\frac{a^2}{\sqrt{a_1}} + \frac{b^2}{\sqrt{b_1}}\right) \\ &\geq \frac{a^2 - b^2}{x(1)} - \frac{2 \cdot \zeta \cdot T \cdot \sqrt{\log n}}{x(1)} \cdot \frac{a^2}{\sqrt{a_1}} \\ &\geq \frac{a - b}{x(1)} \cdot \left((a + b) - \frac{2 \cdot \zeta \cdot T \cdot \sqrt{\log n}}{a - b} \cdot \frac{a^2}{\sqrt{a_1}}\right) \end{aligned}$$

(using $a_1 = a^2/n \cdot (1 \pm \mathrm{o}(1))$ with high probability according to Lemma 72)

$$\geq \frac{a-b}{x(1)} \cdot \left((a+b) - \frac{2 \cdot \zeta \cdot T \cdot \sqrt{\log n} \cdot a^2 \cdot \sqrt{n}}{(a-b) \cdot a \cdot (1-\mathrm{o}(1))}\right)$$

(using $a - b \geq z \cdot \sqrt{n \log^3 n}$)

$$\geq \frac{a-b}{x(1)} \cdot \left((a+b) - \frac{2 \cdot \zeta \cdot a}{z \cdot (1-\mathrm{o}(1))}\right)$$

Now if $z$ is large enough, we obtain for a small positive constant $\varepsilon = \varepsilon(z)$ that

$$a' - b' \geq (a-b) \cdot \left(\frac{a \cdot (1-\varepsilon) + b}{x(1)}\right) . \tag{18.6}$$

Let $\delta > 0$ be a constant. We distinguish the following two cases.

**Case 1: $a < (1+\delta)b$.** We combine (18.5) and (18.6) and obtain with high probability

$$\begin{aligned}\frac{a'-b'}{b'} &\geq (a-b) \cdot \left(\frac{a \cdot (1-\varepsilon) + b}{x(1)}\right) \cdot \frac{x(1)}{b^2 \cdot (1+\mathrm{o}(1))} \\ &= \frac{a-b}{b} \cdot \left(\frac{a \cdot (1-\varepsilon) + b}{b \cdot (1+\mathrm{o}(1))}\right) \\ &\geq \frac{a-b}{b} \cdot \left(\frac{b \cdot (2-\varepsilon)}{b \cdot (1+\mathrm{o}(1))}\right) \\ &= \frac{a-b}{b} \cdot (1+\varepsilon')\end{aligned}$$

where $\varepsilon' > 0$ is a positive constant. Let $a^{(i)}$ and $b^{(i)}$ denote the number of nodes of color $\mathcal{A}$ and $\mathcal{B}$, respectively, after $i$ phases. After $i = \log_{1+\varepsilon'}(a/(c_1 - c_2))$ phases we have with high probability

$$\begin{aligned}\frac{a^{(i)} - b^{(i)}}{b^{(i)}} &\geq \frac{a-b}{b} \cdot (1+\varepsilon')^{\log_{1+\varepsilon'} \frac{a}{a-b}} \\ &= \frac{a-b}{b} \cdot \frac{a}{a-b} \geq 1 .\end{aligned}$$

We therefore get after $i$ phases that $a^{(i)} - b^{(i)} \geq b^{(i)}$ and thus $a^{(i)}/b^{(i)} \geq 2$.

**Case 2: $a \geq (1+\delta)b$.** Considering the ratio between $a'$ and $b'$ based on (18.4) and (18.5), we obtain a quadratic growth as follows.

$$\frac{a'}{b'} \geq \frac{\frac{a^2}{x(1)} \cdot \left(1 - \zeta \cdot \frac{\log^{\frac{3}{2}} n}{\sqrt{a_1}}\right)}{\frac{b^2}{x(1)} \cdot \left(1 + \zeta \cdot \frac{\log^{\frac{3}{2}} n}{\sqrt{b_1}}\right)} = \frac{a^2}{b^2} \cdot \frac{1-\mathrm{o}(1)}{1+\mathrm{o}(1)} \geq \frac{a^2}{b^2} \cdot (1-\mathrm{o}(1))$$

Note that if $a < (1+\delta)b$ then after $i = \log_{1+\epsilon'}(a/(a-b))$ phases we have $a^{(i)}/b^{(i)} \geq 2$ and the second case applies. After $\mathrm{O}(\log\log n)$ phases, every color except for $\mathcal{A}$ drops below $\sqrt{n \log n}$. As described in the proof of Lemma 74, all other colors disappear in the next two-choices phase with high probability. $\square$

**Room for Improvement.** The bound on the plurality consensus time can be further improved to $\mathrm{O}((\log(c_1/(c_1-c_2)) + \log\log n) \cdot \log k)$, which is of interest for cases where $k = \mathrm{o}(\log n)$. This can be achieved by having shorter Bit-Propagation sub-phases in which not all nodes but a large fraction of nodes set their bit.

# 19 Asynchronous Protocol

In this chapter, we present our asynchronous process which is formally defined in Algorithm 19.1. The algorithm is an adaption of `OneBit` to the asynchronous setting. Recall that the key idea to the speed of `OneBit` is to pair a phase in which all nodes execute the two-choice process with a phase in which opinions are propagated quickly. This interweaving of the processes requires the nodes to execute the phases simultaneously. While this is trivial in the synchronous setting, it is impossible in the asynchronous setting. As already stated, the numbers of activations of different nodes can differ by $\Theta(\log n)$. Therefore, any attempt to reach full synchronization is futile if one aims for a run time of $O(\log n)$. To overcome this restriction, we relax the algorithm to the following weaker notion of synchronicity. At any time we only require a $1 - o(1)$ fraction of the nodes to be *almost synchronous*. To cope with the influence of the remaining nodes, we use a toolkit of *gadgets* which we will introduce later. The resulting weak synchronicity allows us to reuse the high-level structure of the analysis of `OneBit`, however still adding several technical challenges. Recall that `OneBit` has one *Two-Choices* sub-phase and one *Bit-Propagation* sub-phase which propagates the choices of the Two-Choices phase to almost all nodes in the network. After executing both sub-phases the distance between $\mathcal{A}$ and any opinion $\mathcal{C}_j \neq A$ increases quadratically. Our asynchronous algorithm also achieves the same growth of $\mathcal{A}$ after each of $O(\log \log n)$ phases. The Bit-Propagation sub-phase is of length $O(\log n / \log \log n)$, amounting to a total run-time of $O(\log n)$, which incidentally is best possible. The reason for this is that after $o(\log n)$ rounds, with high probability some nodes will have ticked not even once. Although from a high level the asynchronous version looks very similar to `OneBit`, we will see later that both implementation and analysis differ greatly and are much more complicated, arguably making this one of the main contributions of this work.

We proceed by analyzing Algorithm 19.1. Recall that in the sequential asynchronous model we assume that a sequence of discrete time steps is given, where at each time step one node is chosen uniformly at random to perform its tick. The main goal of Algorithm 19.1 is to increase the number of nodes of color $\mathcal{A}$ to a level of $a \geq (1/2 + \varepsilon_{\text{Part 1}}) \cdot n$. Once the execution of Algorithm 19.1 has finished, the nodes execute the two-choices algorithm defined in Algorithm 17.1 in an asynchronous manner, after which $\mathcal{A}$ wins with high probability; we analyze this asynchronous two-choices process in Section 19.5. For the analysis of Algorithm 19.1, we will use the following notation and definitions.

**Definition 23.** *Let $c_1$ denote a sufficiently large positive constant. We refer to a series of $n$ consecutive time steps as a* period, *and we combine $c_1 \cdot \log n / \log\log n$ periods to a* phase. *Let furthermore $T_v(t)$ denote the random variable for the number of ticks of node $v$ during the first $t \cdot n$ time steps in the sequential asynchronous model (note that $t$ is not necessarily an integer). We refer to $T_v(t)$ as the* real time *of $v$. Additionally, let $T'_v(t)$ be the time of $v$ at time step $tn$, which is at the beginning the same as $T_v(t)$, but it may deviate from this value during the execution of the algorithm (see shuffle gadget).*

Intuitively, a period is the number of time steps during which each node ticks once in expectation. It will prove convenient to regard a reference point as the one instruction in the algorithm which would be executed in the corresponding period if every node ticked exactly once in every period. Observe that at each time step one node is chosen to tick independently and uniformly at random. Thus $T_v(\tau)$ has a binomial distribution $T_v(\tau) \sim B(\tau \cdot n,\ 1/n)$ and $\mathrm{E}[T_v(\tau)] = \tau$.

For the sake of increased readability, the algorithm specified in Algorithm 19.1 is split into multiple blocks which are defined separately in Algorithm 19.2 (Check-Synchronicity), Algorithm 19.3 (Shuffle Gadget), Algorithm 19.4 (Two-Choices), and Algorithm 19.5 (Bit-Propagation). Whenever invoking one of these blocks we pass on the current number of ticks, the node, and the number of the first and last instruction belonging to that block. In the formal definition, we specify what operations a node performs when selected at a time step to perform its tick $t$. However, all blocks of instructions used in the algorithm contain periods of ticks during which no instructions are given. We will refer to sequences of sequential ticks without instructions as a *do-nothing-block*. Note that in order to increase readability we do not specify these do-nothing-blocks in the formal definitions of the algorithm.

In contrast to the formal definitions, it is more convenient and instructive to represent the algorithm which is executed by each node in each phase in a graphical way. The graphical representations are shown in Figure 19.1 for the overview and in Figure 19.2, Figure 19.3, Figure 19.4, and Figure 19.5 for the individual blocks. In these figures, the instructions are drawn similar to music sheets on a line from left to right, starting with the first instruction at the left endpoint. In the overview given in Figure 19.1, the blocks correspond to multiple instructions specified in the corresponding algorithms. Based on this graphical representation, we will say that a node $V$ is *left* of a reference point $\tau$, if $T'_v(\tau) < \tau$ and *right* of $\tau$, if $T'_v(\tau) > \tau$.

```
Algorithm asynchronous(G, color : $V \to C$)
    let $T = c_1 \cdot \log n / \log\log n$;
    at each node $v$ do asynchronously
     for phase $s = 0$ to $c_2 \cdot \log\log n - 1$
     at each tick $t$
        $\tau_{\mathrm{TC1}} \leftarrow s \cdot T$;
        $\tau_{\mathrm{TC2}} \leftarrow s \cdot T + T/4$;
        $\tau_{\mathrm{BP1}} \leftarrow s \cdot T + 2 \cdot T/4$;
        $\tau_{\mathrm{BP2}} \leftarrow s \cdot T + 3 \cdot T/4$;
        $\tau_{\mathrm{T}} \leftarrow (s+1) \cdot T$;
        if tick $t \in [\tau_{\mathrm{TC1}}, \tau_{\mathrm{TC2}}]$ then
            execute Two-Choices($t$, $v$, $\tau_{\mathrm{TC1}}$, $\tau_{\mathrm{TC2}}$);
        if tick $t \in (\tau_{\mathrm{TC2}}, \tau_{\mathrm{BP1}})$ then
            execute Shuffle($t$, $v$, $\tau_{\mathrm{TC2}} + 1$, $\tau_{\mathrm{BP1}} - 1$);
        if tick $t \in [\tau_{\mathrm{BP1}}, \tau_{\mathrm{BP2}}]$ then
            execute Bit-Propagation($t$, $v$, $\tau_{\mathrm{BP1}}$, $\tau_{\mathrm{BP2}}$);
        if tick $t \in (\tau_{\mathrm{BP2}}, \tau_{\mathrm{T}})$ then
            execute Shuffle($t$, $v$, $\tau_{\mathrm{BP2}}$+1, $\tau_{\mathrm{T}} - 1$);
```

**Algorithm 19.1:** asynchronous voting protocol `AsyncPlurality`. To reach plurality consensus, we first execute this algorithm before executing Algorithm 17.1. The constants $c_1, c_2$ are chosen large enough. For any time step not specified, the algorithm *does nothing*.

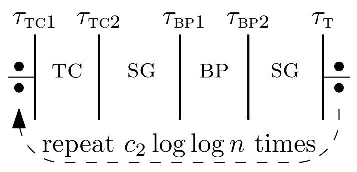


**Figure 19.1:** graphical representation of Algorithm 19.1

We commence the analysis with the observation that throughout the entire process there do not exist nodes which perform more than $\mathrm{O}(\log n)$ ticks, with high probability.

**Observation 75.** *Let $\mathfrak{T}$ denote the total number of time steps until all nodes have completed the execution of Algorithm 19.1. With high probability, we have $\mathfrak{T} \le 3/2 \cdot c_1 \cdot c_2 \cdot n \cdot \log n$. Furthermore, with high probability we have for some constant $c$*

$$\max_{v \in V}\{T_v(3/2 \cdot c_1 \cdot c_2 \cdot \log n)\} \le 2c \cdot \log n$$

*and*

$$\max_{v \in V}\{T'_v(3/2 \cdot c_1 \cdot c_2 \cdot \log n)\} \le 3c \cdot \log n \quad .$$

Observe that according to Algorithm 19.1 a node completes the execution of the algorithm when $T'_v$ reaches $c_1 \cdot c_2 \cdot \log n$. The proof follows, if $c_1 \cdot c_2$ is large enough, from an application of Chernoff bounds to $T_v(\mathfrak{T})$ and from Observation 79 for an arbitrary but fixed node $v$ and union bound over all nodes.

In the following, we implicitly consider a process $P'$ coupled to the original process $P$. While in the original process $P$ it is possible, albeit extremely unlikely, for a single node to tick more than $2c_1 \cdot c_2 \cdot \log n$ times during $\mathfrak{T}$ time steps, we restrict this behavior in the coupled process $P'$. That is, in $P'$ we assume that when a node reaches this limit and is still selected to tick in a subsequent time step, nothing at all happens in that time step. Note that from Observation 75 it follows that with high probability the processes $P$ and $P'$ do not deviate at all.

We proceed by showing that *most* nodes are *almost synchronous* at carefully chosen reference points. Intuitively, a huge fraction of nodes has a number of ticks which is concentrated around the expected value and therefore most nodes will execute instructions which are *close* together. We formalize this concept in Lemma 76 which is based on the following definition.

**Definition 24.** *Let $\Delta = \Theta(\log n/\log\log n)$. We say a node is $\Delta$-close to a reference point $\tau$ for $\tau \le c\cdot\log n$ w.r.t. $T_v$ ($T'_v$) if $|T_v(\tau) - \tau| \le \Delta$ ($|T'_v(\tau) - \tau| \le \Delta$). If we say a node is $\Delta$-close without specifying a reference point, we mean that it is $\Delta$-close to the expected number of ticks. If it is clear from the context whether $T_v$ or $T'_v$ is meant, then this will not be specified.*

**Lemma 76.** *Let $\Delta = \Theta(\log n/\log\log n)$, let $\tau$ be a reference point with $\tau \le c_1 \cdot c_2 \cdot \log n$, and let $Y(\tau)$ be the random variable for the number of nodes which are $\Delta$-close to $\tau$ w.r.t. $T_v$. We have*

$$Y(\tau) \ge n \cdot \Big(1 - \exp\Big(-\Theta\Big(\log n/\log^2\log n\Big)\Big)\Big) .$$

*Proof.* Let $\mathcal{E}_v(\tau)$ be the event that a node $v$ is $\Delta$-close to $\tau$, that is,

$$\mathcal{E}_v(\tau) = \big[\tau - \Theta(\log n/\log\log n) \;\le\; T_v(\tau) \;\le\; \tau + \Theta(\log n/\log\log n)\big] .$$

We apply Chernoff bounds to $T_v(t)$ and obtain

$$\Pr[\mathcal{E}_v(\tau)] \ge 1 - \exp\Big(-\Theta\Big(\frac{\log n}{\log\log n \cdot s}\Big)\Big) , \tag{19.1}$$

where $s$ is the current phase of node $v$ with $s \le \log\log n$. Let in the following $Y_v(\tau)$ be an indicator random variable for a node $v$ and a reference point $\tau$ defined as

$$Y_v(\tau) = \begin{cases} 1, & \text{if } \mathcal{E}_v(\tau) \;, \\ 0, & \text{otherwise.} \end{cases}$$

Summing up over all nodes gives us $Y(\tau) = \sum_{v\in V} Y_v(\tau)$. By linearity of expectation, we have $\mathrm{E}[Y(\tau)] \ge n \cdot (1 - \exp(-\Theta(\log n/(\log\log n \cdot s))))$. Note that the random variables $T_v(\tau)$, and therefore also the random variables $Y_v(\tau)$, are not independent. We thus consider the process of uncovering $Y_v(\tau)$ one node after the other in order to obtain the Doob martingale of $Y(\tau)$ as follows. We define the sequence $Z_j(\tau)$ as $Z_j(\tau) = \mathrm{E}[Y(\tau)|T_j(\tau), \ldots, T_1(\tau)]$ with $Z_0(\tau) = \mathrm{E}[Y(\tau)]$. We have

$$\mathrm{E}[Z_j(\tau)|T_{j-1}(\tau), \ldots, T_1(\tau)] = \mathrm{E}[\mathrm{E}[Y(\tau)|T_j(\tau), \ldots, T_1(\tau)]|T_{j-1}(\tau), \ldots, T_1(\tau)]$$

which, applying the tower property, gives us that

$$\mathrm{E}[Z_j(\tau)|T_{j-1}(\tau), \ldots, T_1(\tau)] = \mathrm{E}[Y(\tau)|T_{j-1}(\tau), \ldots, T_1(\tau)] = Z_{j-1}(\tau) \ .$$

Therefore $Z_j(\tau)$ is indeed the Doob martingale of $Y(\tau)$.

According to Observation 75 each node ticks at most $2c \cdot \log n$ times, that is, $|T_{j+1}(\tau) - T_j(\tau)| \leq 2c \cdot \log n$. This holds with high probability in the original process $P$ and with probability 1 in the coupled process $P'$. Since at most $2c \cdot \log n$ of the random variables $Y_{j+1}(\tau), \ldots, Y_n(\tau)$ differ, we have

$$\begin{aligned}|Z_{j+1}(\tau) - Z_j(\tau)| = |\,&\mathrm{E}[Y_n(\tau) + \cdots + Y_1(\tau)|T_{j+1}(\tau), \ldots, T_1(\tau)] \\ &- \mathrm{E}[Y_n(\tau) + \cdots + Y_1(\tau)|T_j(\tau), \ldots, T_1(\tau)]| \leq 2c \cdot \log n \ .\end{aligned}$$

Applying the Azuma-Hoeffding bound to $Y(\tau) = \sum_{v \in V} Y_v(\tau)$ gives us

$$\Pr\left[|Y(\tau) - \mathrm{E}[Y(\tau)]| \geq \sqrt{c^3 \cdot n \cdot \log^3 n}\right] \leq \exp\left(-\frac{c^3 \cdot n \cdot \log^3 n}{2 \cdot \sum_{j=1}^{n} (2c \cdot \log n)^2}\right) \ ,$$

which for sufficiently large $c$ yields $|Y(\tau) - \mathrm{E}[Y(\tau)]| \leq \sqrt{c^3 \cdot n \cdot \log^3 n}$ with high probability. We finally conclude that, with high probability, at least $n \cdot \Big(1 - \exp\Big(-\Theta\Big(\log n / \log^2 \log n\Big)\Big)\Big)$ nodes are synchronous up to a deviation of at most $\Delta = \Theta(\log n / \log\log n)$ ticks from the expected number of ticks at the given reference point $\tau$. □

**Corollary 77.** *Let $\Delta = \Theta(\log n / \log\log n)$. Throughout the entire execution of Algorithm 19.1, at least $n \cdot \Big(1 - \exp\Big(-\Theta\Big(\log n / \log^2 \log n\Big)\Big)\Big)$ nodes are $\Delta$-close w.r.t. $T_v$ with high probability.*

*Proof.* Let $\Delta' = \Delta/2$, and let $\tau_i$ be a sequence of reference points with $\tau_i = i \cdot \Delta'$. Observe that according to Observation 75 at most $\mathrm{O}(\log\log n)$ reference points suffice to *cover* the entire execution of Algorithm 19.1. We individually apply Lemma 76 with parameter $\Delta'$ to each reference point $\tau_i$. From a union bound over all reference points we obtain that with high probability at most $n \cdot \Theta\Big(\exp\Big(-\log n / \log^2 \log n\Big)\Big) \cdot \log\log n$ nodes are not $\Delta'$-close to all reference points. Since the distance between two reference points is at most $\Delta/2$, at least $n \cdot \Big(1 - \exp\Big(-\Theta\Big(\log n / \log^2 \log n\Big)\Big)\Big)$ nodes are $\Delta$-close at any time step. □

**Notation.** *We will use* $\zeta = \zeta(n) = \exp\Big(-\Theta\Big(\log n/\log^2\log n\Big)\Big)$ *in the remainder of the chapter.*

From Corollary 77 we obtain that with high probability asymptotically almost all nodes are always close to the expected number of ticks. We will refer to those nodes which are $\Delta$-close throughout the execution of the algorithm as the *bulk*. In the following, we seek to reduce the impact of those nodes whose current instruction is far off the reference point. This is ensured by the following two means. First, we require for the Two-Choices sub-phase and the Bit-Propagation that each communication between two nodes happens only in a so-called *communication window* of the calling node's current instruction. This communication window is defined as follows. We assume that whenever a node contacts another node, the current number of ticks is requested and transmitted as well. If the caller receives a number of ticks from the callee which lies outside of the communication window belonging to the caller's current instruction defined in the algorithm, the communication is regarded as failed. The communication windows for the Two-Choices sub-phase and the Bit-Propagation sub-phase are formally specified in Algorithm 19.1. They are furthermore sketched in the graphical representation of the algorithm in Figure 19.1 as dotted lines below the instructions. In addition to the communication window, we use a so-called Check-Synchronicity procedure. This procedure of length $\Theta(\log n/\log\log n)$ works as follows.

```
Algorithm CheckSynchronicity(tick $t$, node $v$, $\tau_{CS1}$, $\tau_{CS2}$)
    if $t \in [\tau_{CS1} + 2\Delta, \tau_{CS2} - 2\Delta - 1]$ then
        let $u \in N(v)$ uniformly at random;
        if $T_v \in [\tau_{CS1} + \Delta, \tau_{CS2} - \Delta - 1]$ then
            $sync_v \leftarrow sync_v + 1$ ;
    if $t = \tau_{CS2}$ then
        if $sync_v \leq (\tau_{CS2} - \tau_{CS1} - 4\Delta)/2$ then
            $dead_v \leftarrow$ TRUE;
        $sync_v \leftarrow 0$;
```

**Algorithm 19.2:** the Check-Synchronicity procedure. Recall that $t$ is the *working-time* ($T'_v$) and $T_v$ is the *real-time.*

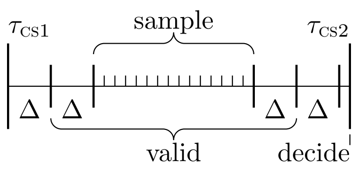


**Figure 19.2:** graphical representation of Algorithm 19.2, the Check-Synchronicity procedure

Assume the Check-Synchronicity procedure consists of instructions $\tau_{CS1}$

to $\tau_{\text{CS2}}$. At each tick $\in [\tau_{\text{CS1}} + 2\Delta, \tau_{\text{CS2}} - 2\Delta - 1]$ of some node $v$, where $t$ corresponds to $T'_v$ each node $v$ opens a connection to a randomly chosen neighbor and queries that neighbor's current number of ticks. If this number is in $[\tau_{\text{CS1}} + \Delta, \tau_{\text{CS2}} - \Delta - 1]$, a counter variable $\texttt{sync}_v$ is incremented. At the very last tick, $\tau_{\text{CS2}}$, each node performs the following check. If more than half of the sampled neighbors are in the proper interval and therefore $\texttt{sync}_v > (\tau_{\text{CS2}} - \tau_{\text{CS1}} - 4\Delta)/2$, the node remains *alive* and resets its counter $\texttt{sync}_v$ to zero. Otherwise the node failed the synchronicity check and marks itself as *dead* according to the following definition.

**Definition 25** (Dead Node)**.** *A* dead node *is a node which did not survive a Check-Synchronicity procedure. Formally, each node $v$ has a Boolean variable $\texttt{dead}_v$ indicating whether $v$ is dead or alive. Dead nodes do not participate in the algorithm any more prior to the final two-choices phase. When they are scheduled to tick, they do nothing in that time step. When contacted by other nodes, they do supply the information that they are dead. Unless explicitly stated otherwise, an alive node which contacts a dead node does not perform any action in that tick. Any node, dead or alive, however, always counts its number of ticks, and communicates this number when asked for it.*

**Definition 26** (Alive Nodes)**.** *We define $L(\tau) = \{v : \texttt{dead}_v(\tau) = \textsc{False}\}$ as the set of* alive nodes *at a given reference point $\tau$.*

Note that a dead node does not become alive ever again throughout the execution of Algorithm 19.1. However, the dead nodes do participate in the final phase, which consists of only two-choices steps. For readability reasons we do not explicitly consider the aliveness of nodes in the formal description of the asynchronous algorithm in Algorithm 19.1.

## 19.1 Analysis of the Check-Synchronicity procedure

Based on above description of the Check-Synchronicity procedure and the definition of dead nodes, we state and prove the following lemma for an arbitrary but fixed Check-Synchronicity block.

**Lemma 78.** *Consider an arbitrary but fixed Check-Synchronicity block and assume that node $v$ goes through the Check-Synchronicity procedure between time steps $\tau_1 n$ and $\tau_2 n$. Let $v$ be an arbitrary but fixed node which is alive at time step $\tau_1 n$. Let furthermore $\texttt{dead}_v(t)$ denote the value of the variable* $\texttt{dead}$ *of node $v$ at time step $tn$. The following statements hold with high probability.*

1. *If $v$ is $\Delta$-close during $[\tau_1 n, \tau_2 n]$ w.r.t. $T'_v$, then $\texttt{dead}_v(\tau_2) = \textsc{False}$.*
2. *If $T'_v(\tau_2) < \tau_1 - \tau_{\text{CS2}} + \tau_{\text{CS1}}$, then $\texttt{dead}_v(\tau_2) = \textsc{True}$.*
3. *If $T'_v(\tau_1) > \tau_2 + \tau_{\text{CS2}} - \tau_{\text{CS1}}$, then $\texttt{dead}_v(\tau_2) = \textsc{True}$.*

*Proof of Lemma 78.1.* From Corollary 77 we obtain that with high probability the set $S$ of nodes $u$ which are $\Delta$-close w.r.t. $T_u$ at all time steps $t \in [\tau_1 n, \tau_2 n]$

has size $|S| \geq (1-\zeta) \cdot n$. Therefore the probability that the node $v$ samples another node which is in the same Check-Synchronicity procedure is at least $1-\zeta$. Since the sampling is done independently, the claim follows from Chernoff bounds on the number of successful sampling ticks. □

*Proof of Lemma 78.2.* Again, we obtain from Corollary 77 that at least $(1-\zeta)\cdot n$ nodes $u$ are $\Delta$-close w.r.t. $T_u$ to $\tau_2$, with high probability, when $v$ reaches $T'_v(\tau_2) = \tau_{\text{CS2}}$. We know that at that time $T'_v(\tau_2) < \tau_1 - \tau_{\text{CS2}} + \tau_{\text{CS1}}$, which implies that $\tau_1 > \tau_{\text{CS2}} + (\tau_{\text{CS2}} - \tau_{\text{CS1}})$. Since $(1-\zeta)\cdot n$ nodes $u$ were $\Delta$-close w.r.t. $T_u$ to $\tau_1$ when $v$ entered the Check-Synchronicity procedure, we conclude that at most $\zeta \cdot n$ nodes could have some of their ticks in $[T'_v(\tau_1), T'_v(\tau_2)]$ while $v$ went through the Check-Synchronicity procedure. As before, the claim follows from Chernoff bounds on the number of successful sampling ticks. □

*Proof of Lemma 78.3.* As before, at least $(1-\zeta)\cdot n$ nodes $u$ are $\Delta$-close w.r.t $T_u$ to a reference point which is at most $\tau_1$, with high probability. Using similar arguments as before, we conclude that the node $v$ went through the entire Check-Synchronicity procedure while at most $\zeta \cdot n$ nodes had some of their ticks in $[T'_v(\tau_1), T'_v(\tau_2)]$. Once again, the claim follows from Chernoff bounds on the number of successful sampling ticks. □

In the remainder we assume that the claims of Lemma 78 hold with probability 1. Similar to the maximum number of ticks, see Observation 75, we can define a coupled process $P'$ in which the claims hold with probability 1. With high probability, the coupled process $P'$ does not deviate from the original process $P$.

Observe that the Check-Synchronicity procedures along with the communication windows ensure that only $\Delta$-close nodes participate in the algorithm. However, we do not yet have any information about the nodes which die in the Check-Synchronicity procedure. In order to establish tight bounds on the number of nodes of a given color which do not survive a Check-Synchronicity block, we need to argue a weak independence between the number of ticks and the color of a node. We therefore introduce the so-called *Shuffle Gadget*. The Shuffle Gadget is a block consisting of $\Theta(\log n/\log\log n)$ instructions. Assume these instructions are defined between $\tau_{\text{SG1}}$ to $\tau_{\text{SG2}}$. The Shuffle Gadget ensures that the order in which nodes perform tick $\tau_{\text{SG2}}$ among the nodes which are alive at this tick is *well distributed*. The Shuffle Gadget is sketched in Figure 19.3. It is formally defined as follows.

## 19.2 Analysis of the Shuffle Gadget

A node's *real time*, denoted as `ticks` in Algorithm 19.1, is the time without any adjustments, that is, the number of ticks it has observed so far. We now describe a gadget which assigns to each node a so-called *working time*, that is,

```
Algorithm Shuffle(tick $t$, node $v$, $\tau_{\mathrm{SG}1}$, $\tau_{\mathrm{SG}2}$)
    jumped$_v$ $\leftarrow$ FALSE;
    medians$_v$ $\leftarrow \emptyset$;
    $\tau_i \leftarrow \tau_{\mathrm{BP}1} + i \cdot (\tau_{\mathrm{BP}2} - \tau_{\mathrm{BP}1})/10$;
    $\tau_{\mathrm{SH}'} \leftarrow \tau_{\mathrm{SH}} + c_3 \cdot \log^2 \log n$;
    if tick $t \in [\tau_0, \tau_1]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_0$, $\tau_1$);
    if tick $t \in [\tau_2, \tau_3]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_2$, $\tau_3$);
    if tick $t \in (\tau_{SH}, \tau_{SH'})$ and jumped$_v$ = FALSE then
        let $u \in N(v)$ uniformly at random;
        increase all values in medians$_v$ by 1;
        medians$_v$ $\leftarrow$ medians$_v$ $\cup$ {T$_u$};
    if tick $t = \tau_{SH'}$ and jumped$_v$ = FALSE then
        $t \leftarrow$ median(medians$_v$) ;
        if tick $t >$ median(medians$_v$) then
            dead$_v$ = TRUE ;
        jumped$_v$ = TRUE;
```

**Algorithm 19.3:** the Shuffle Gadget. Recall that $t$ is the *working-time* ($T'_v$) and $T_v$ is the *real-time.*

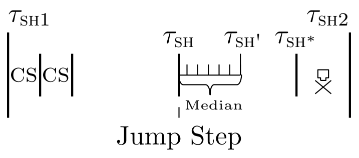


**Figure 19.3:** graphical representation of Algorithm 19.3, the Shuffle Gadget. As usual, the instructions which do not belong either to a Check-Synchronicity procedure or the jump step form do-nothing blocks. Observe, however, that in the $\Theta(\log n/\log\log n)$ ticks which immediately follow the jump step at tick $\tau_{\mathrm{SG}}$ nodes perform so-called *forward-checks.*

$T'_v$ for a node $v$ and denoted as $t$ in Algorithm 19.1, in order to weakly decouple times and colors. The gadget consists of two Check-Synchronicity procedure calls, each followed by a do-nothing-block. Then, during the next $c \log^2 \log n$ ticks, where $c$ is a sufficiently large constant, the node samples a new node, and it collects the *real times* (that is, $T_u$ for a node $u$) it has sampled during these time steps. Note that at each of these $c \log^2 \log n$ ticks, the node increments all real times sampled so far by 1. The median of these values will then be $v$'s working time for the next (two-choices or bit-propagation) sub-phase. That is, the node acts and checks synchronicity according to *this* time, plus the number of ticks the node observed since this working time was set.

If the working time assigned to the node is larger than $\tau_{\mathrm{SH}^*}$, then the node dies. Notice that each node gets the *working time* assigned exactly once while it executes the shuffle gadget. Once the working time is set, that is, time step $\tau_{\mathrm{SH}}$ has been reached once, see Figure 19.3, it will not be reset within this gadget anymore, even if the node jumps back to some time preceding $\tau_{\mathrm{SH}}$. As we show in Lemma 80, most nodes are assigned a working time which lies within the two do-nothing-blocks directly preceding and following $\tau_{\mathrm{SH}}$.

We mention here that a node may have entered the shuffle gadget with a previous working time which is different from the working time sampled in this gadget. If the node goes through a Check-Synchronicity procedure call in this gadget, then this is done according to this old working time. If a node dies in one Check-Synchronicity procedure call, then it will remain dead during the whole execution of the algorithm.

Note that in Observation 75 we considered the number of ticks and the time a node can have. Since the shuffle gadget can reset the time of the nodes, we need the following observation.

**Observation 79.** *With high probability, the time of a node $v$ is reset in a shuffle gadget by at most $\mathrm{O}(\log n/\log\log n)$. The deviation between $T_v(t)$ and $T'_v(t)$ is for every node $v$ at any time step $t < 3/2c_1c_2\log n$ at most $c\log n/4$, with high probability, if $c_1$ is large enough.*

This follows from the fact that in each gadget, a node can reset its working time to the median of the sampled real times. However, if the node is alive, then it must have survived the second Check-Synchronicity procedure in this gadget, meaning that most of the nodes have passed the first Check-Synchronicity procedure of this gadget at the time the working time is assigned. Furthermore, $n\cdot\Big(1-\exp\Big(-\Omega\Big(\log n/\log^2\log n\Big)\Big)\Big)$ nodes are $\Delta$-close w.r.t. their real times during the whole execution of the algorithm, with high probability. Thus, the node will not jump back more than $\mathrm{O}(\log n/\log\log n)$ steps in a shuffle gadget. Furthermore, the node dies if the working time is set to some value larger than $\tau_{\mathrm{SH}^*}$. This implies that the node dies if it jumps forward more than $\mathrm{O}(\log n/\log\log n)$ ticks. Since there are $\mathrm{O}(\log\log n)$ shuffle gadgets, the claim follows.

Next we show that there will be sufficiently many alive nodes during the execution of the algorithm. This generalizes Corollary 77 to our setting in which the working-time of nodes are modified.

**Lemma 80.** *There are at least $n\cdot\Big(1-\exp\Big(-\Omega\Big(\log n/\log^2\log n\Big)\Big)\Big)$ nodes alive during the whole execution of the algorithm, with high probability.*

*Proof.* In this proof we assume for simplicity that the $c''\log^2\log n$ sampled nodes are taken in one single step. First, we show that the median of the sampled times is close to the average of all (real) times, with high probability. We know that there are $n\cdot\Big(1-\exp\Big(-\Omega\Big(\log n/\log^2\log n\Big)\Big)\Big)$ nodes $u$ which are with high probability $\Delta$-close w.r.t. $T_u$ during the whole execution of the algorithm, see also Lemma 76. More precisely, we know that for $n \cdot \Big(1 - \exp\Big(-\Omega\Big(\log n/\log^2\log n\Big)\Big)\Big)$ nodes, if $X_v^\tau$ is the random variable denoting the number of ticks the node $v$ had until period $\tau$, then $|X_v^\tau - \tau| \leq c' \log n/\log\log n$ for some constant $c'$. Thus, if $c''$ is large enough, by Chernoff bounds we conclude that more than $9c''\log^2\log n/10$ nodes are chosen from the set of nodes with this property, with high probability. Then, with high probability the median will be the real time of such a *good* node, that is, a node from the set of nodes being close to the average of all ticks.

Let us consider a fixed execution of a shuffle gadget, and let $v_1, \dots, v_m$ be the nodes which are alive according to their real times during the whole execution of the algorithm. Furthermore, let $X_i^\tau$ be the random variable counting the number of ticks of node $v_i$ within the first $\tau$ periods. As described above, we know that for all these nodes, $|X_i^\tau - \tau| < c'\log n/\log\log n$. Now let a node $X_j^\tau$ be the median taken by $v_i$ in some period $\tau$. Then $|X_i^\tau - X_j^\tau| < 2c'\log n/\log\log n$. Furthermore, the time of node $v_i$, after it has taken $X_j^\tau$, in some period $\tau'$ is $X_j^\tau + X_i^{\tau'} - X_i^\tau$. However, we know that $|X_i^{\tau'} - \tau'| < c'\log n/\log\log n$ and according to the arguments above,

$$|X_j^\tau + X_i^{\tau'} - X_i^\tau - \tau'| < 3c'\log n/\log\log n.$$

Due to Lemma 78, Setting the length of the Check-Synchronicity procedure accordingly, all nodes $v_i$ choosing some median $X_j^\tau$ with $i \in \{1, \dots, k\}$ will be alive during the sub-phase following the shuffle gadget. Taking into account that $k = n \cdot \Big(1 - \exp\Big(-\Omega\Big(\log n/\log^2\log n\Big)\Big)\Big)$, the lemma follows. □

Next we show that any node which survives the shuffle gadget, will die in the next (two choices or bit propagation) sub-phase with probability only $\exp(-\Omega(\log n/\log\log n))$.

**Lemma 81.** *A node which survives the shuffle gadget also survives the following sub-phase (and the following shuffle gadget) with probability* $1 - \exp(-\Omega(\log n/\log\log n))$. *Such a node $v$ is $3c'\log n/\log\log n$-close w.r.t. $T'_v$ during the sub-phase following the current shuffle gadget (and during the next shuffle gadget) with probability* $1 - \exp(-\Omega(\log n/\log\log n))$.

*Proof.* Let $v$ be a node that had assigned to it some time $X_v$ in the shuffle gadget. Assume that this value was assigned to $v$ in period $\tau$. Then we have $|X_v - \tau| \leq c'\log n/\log\log n$. Let us consider period $c''\log n/\log\log n$ after that point in time. The probability that this node is chosen fewer than (or more than) $(c'' - 3c')\log n/\log\log n$ $((c'' + 3c')\log n/\log\log n)$ times for communication by this time is $\exp(-\Omega(\log n/\log\log n))$. Setting the length of the Check-Synchronicity procedure accordingly, we obtain the lemma. □

In the next lemma we consider each color separately.

**Lemma 82.** *For an opinion $i$, let $\mathcal{C}_i(\tau_1)$ be the set of opinions entering a fixed sub-phase. Then,* $|\mathcal{C}_i(\tau_1)|(1 - \exp(-\Omega(\log n/\log\log n))) - \mathrm{O}(\sqrt{n}\log n)$ *nodes*

*$v$ from this set will always satisfy $|X_v^\tau - \tau| \leq 3c' \log n / \log\log n$ for any period $\tau$ during this sub-phase (as well as in the following shuffle gadget), with high probability.*

*Proof.* We know that with high probability no node will be chosen more than $\mathrm{O}(\log n)$ times within a sub-phase, that is, in $\mathrm{O}(\log n/\log\log n)$ periods. We divide the set of nodes $\mathcal{C}_i(\tau_1)$ into sets of size $\sqrt{n}$. Let $v_1, \ldots, v_{\sqrt{n}}$ be the nodes of such a set and assume for simplicity that $\sqrt{n}$ divides $|\mathcal{C}_i(\tau_1)|$. Consider the following process. At a time step $j$ within some period we choose $v_1$ to tick with probability $1/n$. If $v_1$ is not selected, then we choose $v_2$ with probability $1/(n-1)$, and so on. If $v_1, \ldots, v_q$ all decided not to communicate, then we choose $v_{q+1}$ with probability $1/(n-q)$. Thus, for a node $v_{q+1}$ it holds that it communicates in step $j$ with probability at most $1/(n-q) \leq 1/(n-\sqrt{n})$, regardless of the communications of the other nodes. That is, if some node $v_r$ with $r \leq q$ is chosen to tick at this step, then $v_{q+1}$ is selected with probability 0 (no two nodes can be selected), and if there is no $v_r$ with $r \leq q$ selected, then $v_{q+1}$ is chosen with probability $1/(n-q)$. Hence, the probability that a node is chosen from this set to tick more than $\tau + 3c' \log n/\log\log n$ times within $\mathrm{O}(\log n/\log\log n)$ periods is still $\exp(-\Omega(\log n/\log\log n))$, see also Lemma 81. Since the upper bound $1/(n-\sqrt{n})$ is independent of the behavior of the other nodes, we apply Chernoff bounds and obtain that at most $\sqrt{n}/\exp(\Omega(\log n/\log\log n))$ nodes are chosen for communication more than $\tau + 3c' \log n/\log\log n$ times each, with high probability.

On the other hand, each node communicates at most $\log n$ times within a period, thus there are at least $n - \sqrt{n}\log n$ steps in which $v_q$ communicates with probability at least $1/n$. Similar arguments as above imply that at most $\sqrt{n}/\exp(\Omega(\log n/\log\log n))$ nodes are chosen for communication fewer than $\tau - 3c' \log n/\log\log n$ times, with high probability. Putting these together, and applying a union bound over all sets of size $\sqrt{n}$ of a certain opinion $i$ and over all periods of this sub-phase, we obtain the lemma. $\square$

## 19.3 Analysis of the Two-Choices sub-phase

We consider an arbitrary but fixed Two-Choices sub-phase which consists of instructions $\tau_{\mathrm{TC}1}$ to $\tau_{\mathrm{TC}2}$. As in Algorithm 19.4, we define $\tau_i$ for that Two-Choices sub-phase as $\tau_{\mathrm{TC}1} + i \cdot (\tau_{\mathrm{TC}2} - \tau_{\mathrm{TC}1})/10$. The two-choice sampling step occurs at instruction $\tau_{\mathrm{TC}} = \tau_5$ and the communication window $[\omega_1, \omega_2]$ is defined as $[\omega_1, \omega_2] = [\tau_1, \tau_{10} - 2]$. We commence the analysis of the Two-Choices sub-phase with the following observation, which also applies to the Bit-Propagation sub-phase.

**Observation 83.** *Let $v \in L(\tau_{\mathrm{TC}2})$ be a node which is alive at the end of the Two-Choices sub-phase and let $\theta$ be the time step when $v$ performs the Two-Choices sampling step at tick $\tau_{\mathrm{TC}}$. Furthermore, let $S$ be the set of nodes*

```
Algorithm Two-Choices (tick $t$, node $v$, $\tau_{\text{TC1}}$, $\tau_{\text{TC2}}$)
    $\tau_i \leftarrow \tau_{\text{TC1}} + i \cdot (\tau_{\text{TC2}} - \tau_{\text{TC1}})/10$;
    $[\omega_1, \omega_2] \leftarrow [\tau_1, \tau_{10} - 2]$;
    if tick $t \in [\tau_0, \tau_1]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_0$, $\tau_1$);
    if tick $t \in [\tau_2, \tau_3]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_2$, $\tau_3$);
    if tick $t = \tau_5$ then
        let $u_1, u_2 \in N(v)$ uniformly at random;
        if color($u_1$) = color($u_2$) and $u_1, u_2$ in $[\omega_1, \omega_2]$ then
            intColor($v$) $\leftarrow$ color($u_1$);
        else
            intColor($v$) $\leftarrow$ NULL;
    if tick $t \in [\tau_7, \tau_8]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_7$, $\tau_8$);
    if tick $t \in [\tau_9, \tau_{10} - 1]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_9$, $\tau_{10} - 1$);
    if tick $t = \tau_{\text{TC2}}$ then
        if intColor($v$) $\neq$ NULL then
            color($v$) $\leftarrow$ intColor($v$);
            bit($v$) $\leftarrow$ TRUE;
        else
            bit($v$) $\leftarrow$ FALSE;
```

**Algorithm 19.4:** Two-Choices sub-phase

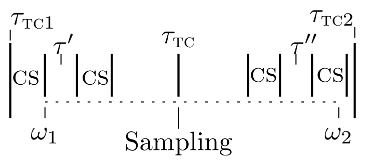


**Figure 19.4:** graphical representation of Algorithm 19.4, the Two-Choices sub-phase. The dotted line depicts the communication window. The reference points $\tau'$ and $\tau''$ are used in the analysis.

*which are alive and within the communication window for that Two-Choices sampling step at time step $\theta$. Finally, let $\mathcal{C}_j(\tau_{\text{TC1}})$ denote the set of nodes which are alive and have color $\mathcal{C}_j$ at their tick $\tau_{\text{TC1}}$. With high probability, we have for a color $\mathcal{C}_j$*

$$|\{u \in S : \texttt{color}(u) = \mathcal{C}_j\}| \geq \\ |\mathcal{C}_j(\tau_{\text{TC1}})| \cdot \left(1 - \exp\left(-\Omega\left(\frac{\log n}{\log\log n}\right)\right)\right) - \sqrt{n} \cdot \log n \ .$$

*Proof.* Lemma 82 implies that from the set of nodes $\mathcal{C}_j(\tau_{\mathrm{TC}1})$ at least $|\mathcal{C}_j(\tau_{\mathrm{TC}1})|(1 - \exp(-\Omega(\log n/\log\log n))) - \sqrt{n}\log n$ have the property that with high probability their tick(s) in some period $\tau$ differ from $\tau$ by at most $3c'\log n/\log\log n$, that is, are concentrated ($c'$ is the constant defined in Lemma 82). We refer to them as $\mathcal{C}^c_j(\tau_{\mathrm{TC}1})$. Thus, at time step $\theta$, $v$ has already passed $\tau_3$ (since $\tau_{\mathrm{TC}>}\tau_3$), and since $v$ is alive most nodes must have been in the range $[\tau_2, \tau_3]$ when $v$ had its tick $\tau_3$, with high probability. This implies that all nodes of $\mathcal{C}^c_j(\tau_{\mathrm{TC}1})$ have already passed time step $\tau'$. Also, using similar arguments we conclude that with high probability $v$ can only survive the Check-Synchronicity procedure starting with $\tau_7$ if most nodes have not reached $\tau_8$ when $v$ executes the two-choices protocol. Hence, with high probability no node of $\mathcal{C}^c_j(\tau_{\mathrm{TC}1})$ has reached $\tau''$ by step $\theta$. The statement of the observation follows then from the result of Lemma 82 w.r.t. the size of $\mathcal{C}^c_j(\tau_{\mathrm{TC}1})$. □

For any color $\mathcal{C}_j$ let $\tilde{X}_j(\tau_{\mathrm{TC}})$ denote the set of alive nodes which succeeded in the Two-Choices sampling step at tick $\tau_{\mathrm{TC}}$ and set their intermediate color $\mathtt{intColor}_v(\tau_{\mathrm{TC}})$ in step $\tau_{\mathrm{TC}}$ to $\mathcal{C}_j$, that is, $\tilde{X}_j(\tau_{\mathrm{TC}}) = \{v \in L(\tau_{\mathrm{TC}1}) : \mathtt{intColor}_v(\tau_{\mathrm{TC}}) = \mathcal{C}_j\}$. Furthermore, let $\tilde{X}(\tau_{\mathrm{TC}})$ denote the set of nodes which have their intermediate color set after tick $\tau_{\mathrm{TC}}$, that is, $\tilde{X}(\tau_{\mathrm{TC}}) = \bigcup_{Cj} \tilde{X}_j(\tau_{\mathrm{TC}})$.

The following lemma is the asynchronous counterpart to Lemma 72. It establishes that the number of nodes which pick up a bit for color $\mathcal{C}_j$ is with high probability concentrated around the expectation. This provides the basis for the quadratic growth of the color ratios during the phase.

**Lemma 84.** *Consider an arbitrary but fixed Two-Choices sub-phase with two-choice sampling step at tick $\tau_{\mathrm{TC}}$. Among the nodes which survive the final Check-Synchronicity procedure call at the end of the Two-Choices sub-phase we have*

$$\left|\tilde{X}_j(\tau_{\mathrm{TC}2})\right| = |L(\tau_{\mathrm{TC}2})| \cdot \frac{|\{v \in L(\tau_{\mathrm{TC}1}) : v \in \mathcal{C}_j(\tau_{\mathrm{TC}1})\}|^2}{n^2} \cdot (1 \pm \mathrm{o}(1)) \pm \mathrm{O}\left(\log^2 n\right)$$

*with high probability. Furthermore, in some period $\tau$ in the Two-Choices sub-phase or in the following shuffle gadget we have*

$$\begin{aligned}\left|\left\{v \in \tilde{X}_j(\tau_{\mathrm{TC}2}) \mid |T'_v(\tau) - \tau| \le \frac{3c'\log n}{\log\log n}\right\}\right| \\ =& |L(\tau_{\mathrm{TC}2})| \cdot \frac{|\{v \in L(\tau_{\mathrm{TC}1}) : v \in \mathcal{C}_j(\tau_{\mathrm{TC}1})\}|^2}{n^2} \cdot (1 \pm \mathrm{o}(1)) + \mathrm{O}\left(\log^2 n\right) \ .\end{aligned}$$

*Proof.* According to Observation 83 each node which survives all Check-Synchronicity procedure calls in the Two-Choices sub-phase sets its intermediate color to $\mathcal{C}_j$ with probability at least $(\mathcal{C}^c_j/n)^2$ and at most $\left(|\mathcal{C}_j(\tau_{\mathrm{TC}1})|/n \cdot \left(1 - \exp\left(-\Omega\left(\log n/\log^2\log n\right)\right)\right)\right)^2$, independently. According to Lemma 82

$$|\mathcal{C}^c_j| \ge |\mathcal{C}_j(\tau_{\mathrm{TC}1})| \cdot \left(1 - \exp\left(-\Omega\left(\frac{\log n}{\log\log n}\right)\right)\right) - \sqrt{n} \cdot \log n \ .$$

By applying Chernoff bounds, we obtain the lemma. □

From Lemma 84 and Lemma 82 we obtain the following corollary.

**Corollary 85.** *Let a Bit-Propagation sub-phase follow the Two-Choices sub-phase. Then,*

$$|L(\tau_{\mathrm{TC}2})| \cdot \frac{|\{v \in L(\tau_{\mathrm{TC}1}) : v \in \mathcal{C}_j(\tau_{\mathrm{TC}1})\}|^2}{n^2} \cdot (1 \pm \mathrm{o}(1)) + \mathrm{O}\left(\log^2 n\right)$$

*nodes have intermediate color $\mathcal{C}_j$ at the end of the Two-Choices sub-phase and are $\Delta$-close during the whole Bit-Propagation sub-phase as well as during the following shuffle gadget,with high probability.*

## 19.4 Analysis of the Bit-Propagation sub-phase

```
Algorithm Bit-Propagation(tick $t$, node $v$, $\tau_{\mathrm{BP}1}$, $\tau_{\mathrm{BP}2}$)
    $\tau_i \leftarrow \tau_{\mathrm{BP}1} + i \cdot (\tau_{\mathrm{BP}2} - \tau_{\mathrm{BP}1})/10$;
    $[\omega_1, \omega_2] \leftarrow [\tau_1, \tau_{10} - 2]$;
    if tick $t \in [\tau_0, \tau_1]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_0$, $\tau_1$);
    if tick $t \in [\tau_2, \tau_3]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_2$, $\tau_3$);
    if tick $t \in [\tau_3 + 1, \tau_7 - 1]$ then
        if bit($v$) = FALSE then
            let $u \in N(v)$ uniformly at random;
            if bit($u$) = TRUE and $u$ in $[\omega_1, \omega_2]$ then
                bit($v$) ← TRUE;
                color($v$) ← color($u$);
    if tick $t \in [\tau_7, \tau_8]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_7$, $\tau_8$);
    if tick $t \in [\tau_9, \tau_{10}]$ then
        execute CheckSynchronicity($t$, $v$, $\tau_9$, $\tau_{10}$);
```

**Algorithm 19.5:** Bit-Propagation sub-phase

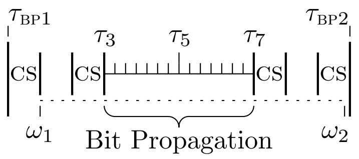


**Figure 19.5:** graphical representation of Algorithm 19.5, the Bit-Propagation sub-phase. The dotted line depicts the communication window.

In the following, we observe that the behavior of the asynchronous system during the bit-propagation sub-phase can be modeled by a generalized version of the so-called Pólya urn process. In the original urn model [JK77], we are given an urn containing marbles of two colors. In every step, one marble is drawn uniformly at random from the urn. Its color is observed, the marble is returned to the urn and one more marble of the same color is added. For any given color, the ratio of marbles with that given color over the total number of marbles is a martingale.

In the generalized version, we allow that in addition to the original model marbles of a given color can be added or removed at fixed time steps. Intuitively, the generalized urn process corresponds to the bit-propagation sub-phase, where at fixed time steps *slow* nodes enter the communication window and *fast* nodes prematurely exit the communication window. Since we perform a worst-case analysis, we assume that no additional black marbles are added nor white marbles removed. The generalized urn process therefore runs in discrete time steps where in each step either a black marble is removed, a white marble is added, or a marble is added with a random color chosen according to the current color distribution within the urn. Formally, the generalized urn process is defined as follows.

**Definition 27** (Generalized Pólya Urn Process)**.** *Let* $\text{Pólya}(x, y, S, T)$ *with* $x, y \geq 0$*,* $S, T \subset \mathbb{N}$*, and* $S \cap T = \emptyset$ *be the following urn process. At the beginning there are* $x$ *black marbles and* $y$ *white marbles in the urn. For every time step* $t$ *let* $x(t)$ *and* $y(t)$ *be the number of black and white marbles in the urn, respectively. In every time step* $t$ *the process does the following.*

- *If* $t \in S$*, remove a black marble.*
- *If* $t \in T$*, add a white marble.*
- *If* $t \notin S \cup T$*, with probability* $x(t)/(x(t) + y(t))$ *add a black marble and with the remaining probability* $y(t)/(x(t) + y(t))$ *add a white marble.*

Since the generalized Pólya urn process corresponds to the bit-propagation sub-phase, we show in the following lemma how to majorize the generalized process by the original Pólya urn process which we can analyze by means of martingale techniques. Observe that the generalized Pólya urn process with parameters $S = T = \emptyset$ describes precisely the original Pólya urn process.

**Lemma 86.** *Let* $x, y \in \mathbb{N}$*,* $S, T \subset \mathbb{N}$ *and* $|S| \leq x$*. Let* $F_x(t)$ *be the fraction of black marbles and* $F_y(t)$ *the fraction of white marbles in* $\text{Pólya}(x, y, S, T)$ *at time step* $t$ *and let furthermore* $F'_x(t)$ *be the fraction of black marbles and* $F'_y(t)$ *the fraction of white marbles in* $\text{Pólya}(x - |S|, y + |T|, \emptyset, \emptyset)$*. We can couple the generalized Pólya urn process with the original Pólya urn process such that*

$$F'_x(t) \preceq F_x(t + |S| + |T|) \text{ and } F'_y(t) \succeq F_y(t + |S| + |T|) \ ,$$

*where the operators* $\preceq$ *and* $\succeq$ *denote stochastic minorization and majorization, respectively.*

*Proof.* The proof follows from an induction on the elements of $S \cup T$. We will show that we can couple the generalized Pólya urn process with another generalized Pólya urn process where each element of $S$ and $T$ has been moved to the first $|S| + |T|$ steps such that $S \cup T = \{1, 2, \ldots, |S| + |T|\}$. As induction step, we will show that we maintain a majorization with each move.

Let $z_i$ be the $i$-th largest element of $S \cup T$ and let $z$ be the smallest element of $S \cup T$ for which $z > z_i$, that is, $z$ is the first element which can be moved to an earlier step. If $z \in S$, we define $S' = S \cup \{z - 1\} \setminus \{z\}$ and $T' = T$. Otherwise, if $z \in T$, we define $S' = S$ and $T' = T \cup \{z - 1\} \setminus \{z\}$.

For the induction, let $\hat{F}_x(t)$ denote the fraction of black marbles and $\hat{F}_y(t)$ the fraction of white marbles in Pólya$(x, y, S, T)$. Let analogously $\hat{F}'_x(t)$ denote the fraction of black marbles and $\hat{F}'_y(t)$ the fraction of white marbles in Pólya$(x, y, S', T')$. We can couple Pólya$(x, y, S', T')$ with Pólya$(x, y, S, T)$ such that the two processes do not deviate up to step $z - 2$. The following two steps, $z - 1$ and $z$, consist of removing a black marble in one step and adding a marble with color chosen according to the urn model in the other step. Let $\mathcal{E}$ denote the event that we sample and add a black marble in step $z - 1$ in Pólya$(x, y, S, T)$, and let $\mathcal{E}'$ be the event that we sample and add a black marble in step $z$ in Pólya$(x, y, S', T')$. We observe that $\Pr[\mathcal{E}'] \leq \Pr[\mathcal{E}]$. From the structure of the urn process it follows that the majorization holds for all following sampling steps and therefore

$$\hat{F}'_x(t) \preceq \hat{F}_x(t) \text{ and } \hat{F}'_y(t) \succeq \hat{F}_y(t) \ .$$

This concludes the induction step.

By repeated application of above observation we can move $z_i$ to position $i$. From the induction over all possible moves of elements of $S \cup T$ we obtain that we can maintain the majorization while moving all elements to the first $|S| + |T|$ positions. As a final observation we note that if we have $S \cup T = \{1, 2, \ldots, |S| + |T|\}$ then the urn process Pólya$(x, y, S, T)$ starting after step $|S| + |T|$ is identical to the urn process Pólya$(x - |S|, y + |T|, \emptyset, \emptyset)$. □

We now focus on the analysis of the Bit-Propagation sub-phase. Similar to the analysis done for the synchronous case, we first analyze the number of bits which are set during the Bit-Propagation sub-phase without taking the color into consideration. The following lemma follows from the observation that the Bit-Propagation can be modeled by a simple asynchronous randomized-gossip-based information dissemination process.

**Lemma 87.** *All nodes which survive the Check-Synchronicity procedure following the Bit-Propagation sub-phase have their bit set with high probability.*

*Proof.* Consider the references points $\tau_3$, $\tau_5$, and $\tau_7$ of the Bit-Propagation block as defined in Algorithm 19.5. We will show that the number of set bits increases quickly in the intervals $[\tau_3, \tau_5]$ and $(\tau_5, \tau_7]$. More specifically, we proceed in three parts and we argue that with high probability

(i) $\left|\tilde{X}(\tau_3)\right| \geq c \cdot n/k^2$ for some small enough constant $c > 0$,

(ii) $\left|\tilde{X}(\tau_5)\right| \geq n/2$, and

(iii) $\left|\tilde{X}(\tau_7)\right| \geq (1 - \mathrm{o}(1)) \cdot n$.

Lemma 80 together with (iii) implies the claim.

In each part we will rely on the fact (Lemma 80) that at each reference point $\tau_3$, $\tau_5$, and $\tau_7$ the number of alive nodes is at least $(1 - \mathrm{o}(1)) \cdot n$.

**Part (i).** By Corollary 85, the number of nodes with a bit set (to TRUE) is at least

$$\left|\tilde{X}(\tau_{\mathrm{SH2}})\right| = |L(\tau_{\mathrm{SH2}})| \cdot \frac{|\{v \in L(\tau_{\mathrm{TC1}}) : v \in \mathcal{C}_j\}|^2}{n^2} \cdot (1 - \mathrm{o}(1)) = \Omega\left(a^2/n\right).$$

Furthermore, we have $a^2/n \geq (n/k)^2/n = n/k^2$ and thus $\left|\tilde{X}(\tau_{\mathrm{SH2}})\right| = \Omega(n/k^2)$. By Lemma 80, the fraction of these nodes which are alive at $\tau_3$ is at least $c \cdot n/k^2$ for some small enough constant $c > 0$.

**Part (ii).** Let $x(t)$ be the number of nodes which have a bit set at time $t$. We show by induction that for $i \in [0, \tau_5 - \tau_3]$,

$$x(\tau + i) \geq \min\left\{\frac{n}{2}, \frac{c \cdot n}{k^2}\left(1 + \frac{1}{8e}\right)^i\right\}.$$

Fix an arbitrary $i$. If $x(\tau + i - 1) > n/2$, then we are done and thus we assume $x(\tau + i - 1) < n/2$. Let $U$ be the set of nodes which did not have their bit set at time $\tau + i - 1$. By assumption, $|U| \geq n/2$. Let $S$ be the set of nodes which ticked at round $\tau + i$. By a standard balls-into-bins arguments, we have that $|S|$ is at least $n/e$ with high probability. Since each node is equally likely to tick we conclude from this and the assumption $x(\tau + i - 1) \leq n/2$ that $|S \cap U| \geq n/(4e)$ with high probability.

For $j \in S \cap U$ define $Z_j$ to the indicator variable that node $j$ set a bit in round $\tau + i$. Note that all $Z_j$ are independent and $\Pr[Z_j = 1] = x(\tau + i - 1)/n$. For $Z = \sum Z_i$, $Z \geq |S \cap U| \cdot x(\tau + i - 1)/(2n) \geq x(\tau + i - 1)/(8e)$ with high probability, by Chernoff bounds. We get that

$$x(\tau+i) \geq x(\tau+i-1) + |\{i \mid Z_i = 1\}| \geq x(\tau+i-1)\left(1 + \frac{1}{8e}\right) \geq \frac{c \cdot n}{k^2} \cdot \left(1 + \frac{1}{8e}\right)^i,$$

where the last inequality is due to the induction hypothesis. This completes the induction. We obtain, using $\tau_5 - \tau_3 - 2\Delta \geq 16 \log(k^2/(2c))$,

$$x(\tau_5) \geq \frac{c \cdot n}{k^2}\left(1 + \frac{1}{8e}\right)^{\tau_5 - \tau_3 - 2\Delta} \geq \frac{c \cdot n}{k^2} \cdot \frac{k^2}{2c} = n/2 \ .$$

This completes the proof of (ii).

**Part (iii).** Let $S$ be the set of nodes which do not have a bit set at time $\tau_5$ and which were alive at $\tau_7$. Consider an arbitrary node $i \in S$. Since $i$ was alive at $\tau_7$ we have that it ticked at least $\iota = \tau_7 - \tau_5 - 2\Delta$ times. The probability that it never sampled a node with a set bit is thus at most $2^{-\iota} = 2^{-\mathrm{O}(\log n/\log\log n)}$. Hence, the expected number of nodes of $S$ not setting a bit is, by using independence and Chernoff bounds, at most $2|S| \cdot 2^{-\mathrm{O}(\log n/\log\log n)}$. Furthermore,

$$\begin{aligned}\left|\tilde{X}(\tau_7)\right| &\geq |L(\tau_{\mathrm{TC}2})| - 2|S| \cdot 2^{-\mathrm{O}(\log n/\log\log n)} \\ &\geq (1-\mathrm{o}(1)) \cdot n - 2|S| \cdot 2^{-\mathrm{O}(\log n/\log\log n)} \qquad \geq (1-\mathrm{o}(1)) \cdot n \ .\end{aligned}$$

This completes Part (iii) and the proof follows by Lemma 80. □

We now state the main lemma for the bit propagation sub-phase. On a high level we argue that after the bit-propagation phase a large fraction of nodes have their bit set and that the distribution leads to a quadratic increase in the difference between the largest and second largest color.

**Lemma 88.** *Consider an arbitrary but fixed Bit-Propagation sub-phase which consists of ticks $\tau_{\mathrm{BP}1}$ to $\tau_{\mathrm{BP}2}$. For every $t \in [\tau_{\mathrm{BP}1}, \tau_{\mathrm{BP}2}]$ we have with high probability that*

$$|X_1(t)| = |X(t)| \cdot \frac{|X_1(\tau_{\mathrm{BP}1})|}{|X(\tau_{\mathrm{BP}1})|} \cdot (1-\mathrm{o}(1)) \ ,$$

*and for any color $\mathcal{C}_j \neq \mathcal{A}$*

$$|X_j(t)| = |X(t)| \cdot \frac{|X_j(\tau_{\mathrm{BP}1})|}{|X(\tau_{\mathrm{BP}1})|} \cdot (1+\mathrm{o}(1)) + \mathrm{O}\left(n^{2/3}\right) \ .$$

*Proof.* We start by observing that during the Bit-Propagation sub-phase nodes only perform actions when they have their bit not yet set. Therefore, a node sets its bit at most once. Note that it may very well happen that a node without bit is selected to tick, and this node samples another node with unset bit. However, the color distribution among the nodes does not change in that case at this time step. We therefore consider the subsequence of time steps at which nodes which do not yet have their bit set successfully sample another node which has its bit set. More precisely, we only consider those time steps, during which a node sets its bit and changes its color.

In the following, we couple the Bit-Propagation sub-phase with respect to color $\mathcal{A}$ with the generalized Pólya urn process defined in Definition 27. For the coupling, we assume that each node of color $\mathcal{A}$ which has its bit set corresponds to a black marble and each node of any other color $\mathcal{C}_j \neq \mathcal{A}$ which has its bit set corresponds to a white marble. While we describe the coupling in detail for color $\mathcal{A}$ to give lower bounds, the same arguments holds symmetrically if we couple such that nodes of any other color $\mathcal{C}_j \neq A$ which have their bit set correspond to white marbles and the remaining nodes of color $\mathcal{C}_{j'} \neq \mathcal{C}_j$ which have their bit set correspond to black marbles.

There are three types of events which alter the content of the urn. First, nodes may tick, survive the Check-Synchronicity procedure, and enter the communication window of the Bit-Propagation sub-phase late, that is at any time after $\tau_2$. Since these nodes now can be sampled by other nodes which are already in the urn, they alter the urn by adding either a black or white marble, depending on their color. Their number will be small, as we will see later. Secondly, nodes may leave the communication window early, that is before $\tau_7$, and therefore decrease the number of nodes which can be sampled. This alters the urn by removing either a black or a white marble, depending on their color. Again, we will show that the error introduced by this is small. Finally, nodes which have do not yet have their bit set may open a connection to a node which has its bit set. In that case, an additional marble is added. Observe that the nodes open connections uniformly at random. Therefore, the color of the marble to be added is selected uniformly at random from the colors of the set of nodes which are already in the urn, that is, they have their bit set.

Since we are interested in a lower bound on the number of black marbles in the urn at any time, we can couple the original process with a process where (i) nodes $T$ which enter the urn later are always white and (ii) the nodes $S$ leaving the urn early are black.

We therefore conclude that we can model the Bit-Propagation sub-phase for color $\mathcal{A}$ by means of the generalized Pólya urn process Pólya$(x, y, S, T)$, where we have

$$x \geq |X_1(\tau_{\mathrm{BP4}})| \cdot (1 - \mathrm{o}(1)) \ ,$$
$$y \leq \sum_{\mathcal{C}_j \neq \mathcal{A}} |X_j(\tau_{\mathrm{BP4}})| \cdot (1 + \mathrm{o}(1)) \ ,$$

the set $S \subset \mathbb{N}$ contains the steps in the urn process at which nodes of color $\mathcal{A}$ prematurely leave the urn, and $T \subset \mathbb{N}$ is the set of steps in the urn process at which late nodes of color $\mathcal{C}_j \neq A$ enter the urn. Observe that in a worst-case analysis, we do not take into consideration nodes of color $\mathcal{A}$ which are added to the urn, nor nodes of any other color $\mathcal{C}_j \neq A$ which are removed from the urn.

From Lemma 86 we obtain that we can couple Pólya$(x, y, S, T)$ with Pólya$(x - |S|, y + |T|, \emptyset, \emptyset)$, which again corresponds to the original Pólya urn model.

Let $M$ be the minimum number of nodes which have their bit set in the Pólya urn process. Due to Lemma 80, we have with high probability

$$M \geq |X(\tau_{\mathrm{BP1}})| - \zeta \cdot n \geq n/(2k) - \zeta \cdot n \geq n/(4k)$$

for sufficiently small $k$. Let $F'_x(t)$ be the fraction of black marbles in step $t$ of the original Pólya urn process corresponding to Pólya$(x - |S|, y + |T|, \emptyset, \emptyset)$ and recall that the fraction of black marbles in the original Pólya urn process is a martingale. Observe furthermore that $|F'_x(t) - F'_x(t-1)| \leq 1/M$ throughout the entire process. Let $T$ be the last step of the original Pólya urn process and observe that $T \leq n$. Applying Azuma's inequality to $F'_x(t)$ for any $t \leq T$ gives us

$$\begin{aligned}\Pr[|F'_x(t) - F'_x(1)| \geq \delta] &\leq 2 \cdot \exp\left(-\frac{\delta^2}{2 \cdot \sum_{i=1}^{t} 1/M^2}\right) \\ &\leq 2 \cdot \exp\left(-\frac{\delta^2 \cdot M^2}{2 \cdot t}\right) \ .\end{aligned}$$

We set $\delta = c' \cdot k \cdot \sqrt{\log n/n}$ and obtain

$$\begin{aligned}\Pr\left[|F'_x(t) - F'_x(1)| \geq c \cdot k \cdot \sqrt{\log n/n}\right] &\leq 2 \cdot \exp\left(-\frac{c \cdot k^2 \cdot M^2 \cdot \log n}{n \cdot T}\right) \\ &\leq 2 \cdot \exp(-c \cdot \log n) \ .\end{aligned}$$

From the calculation above, and taking the union bound over the colors, we see that for any color $\mathcal{C}_j$ the ratio of nodes of that color to all nodes remains almost a constant. To derive a lower bound on the number of black marbles at the end of the process or, equivalently, a lower bound on $|X_1(\tau_{\text{BP2}})|$, we bound $|S|$ and $|T|$ in the following way. By Corollary 85 we have $|S| = \text{o}(|X_1(\tau_{\text{BP2}})|)$. Let $x = |X_1(\tau_{\text{BP2}})|$ Furthermore, by Lemma 80, $|T| \leq \zeta n$. We thus have

$$F'_x(1) \geq \frac{x - |S|}{|X(\tau_{\text{BP1}})|} \geq \frac{x(1-\text{o}(1))}{|X(\tau_{\text{BP1}})|}$$

and for any $t \leq T$,

$$\begin{aligned}F'_x(t) &\geq \frac{x(1-\text{o}(1))}{|X(\tau_{\text{BP1}})|} - c \cdot k \cdot \sqrt{\log n/n} \\ &= \frac{x(1-\text{o}(1)) - |X(\tau_{\text{BP1}})|c \cdot k \cdot \sqrt{\log n/n}}{|X(\tau_{\text{BP1}})|} \\ &\geq \frac{x(1-\text{o}(1)) - (2k \cdot \alpha^2/n)c \cdot k \cdot \sqrt{\log n/n}}{|X(\tau_{\text{BP1}})|} \\ &\geq \frac{x(1-\text{o}(1))}{|X(\tau_{\text{BP1}})|} \ ,\end{aligned}$$

where by Chernoff bounds the second inequality holds with high probability. Furthermore, the probability that a randomly chosen node has opinion $\mathcal{A}$ at time $t$ is is at least $|X_1(\tau_{\text{BP1}})|/|X(\tau_{\text{BP1}})| \cdot (1 - \text{o}(1))$ and thus using a standard coupling and Chernoff bounds we derive that with high probability

$$|X_1(t)| = |X(t)| \cdot \frac{|X_1(\tau_{\text{BP1}})|}{|X(\tau_{\text{BP1}})|} \cdot (1 - \text{o}(1)).$$

It remains to establish an upper bound on $|X_j(\tau_{\text{BP2}})|$ for every other color $C_j \neq \mathcal{A}$. We will use a symmetric argument. Let $\mathcal{C}_j$ be an arbitrary but fixed color. This time we use the white marbles to represent $\mathcal{C}_j$ and the black marbles to represent all colors $\mathcal{C}_i \neq \mathcal{C}_j$. Again, let $S$ and $T$ denote the set of black marbles which are removed early and the set of white marbles which are added during the urn process. By Corollary 85, $|T| \leq (1 - o(1)) \cdot |X_j(\tau_{\text{BP2}})| + O\left(\log^2 n\right)$ and, by Lemma 80, $|S| \leq \zeta \cdot n$.

According to Lemma 86 we can again couple Pólya$(x, y, S, T)$ with the original Pólya urn process Pólya$(x - |S|, y + |T|, \emptyset, \emptyset)$ and derive, for every $t \leq T$,

$$\Pr\left[|F_y'(t) - F_y'(1)|\right] \leq 2 \cdot \exp\left(-\frac{c \cdot k^2 \cdot M^2 \cdot \log n}{n \cdot T}\right) ,$$

which yields, by using the same arguments as before,

$$F_y'(t) \leq \frac{x(1 + o(1))}{|X(\tau_{\text{BP1}})|} + O\left(\frac{1}{n^{1/3}}\right) .$$

Symmetrically, the number of nodes of color $\mathcal{C}_j$ is with high probability

$$|X_j(t)| = |X(t)| \cdot \frac{|X_j(\tau_{\text{BP1}})|}{|X(\tau_{\text{BP1}})|} \cdot (1 + o(1)) + O\left(n^{2/3}\right) ,$$

which finishes the proof. □

Recall that after the Bit-Propagation sub-phase the nodes go through a shuffle gadget. The following lemma deals with the properties of the nodes after they went through this gadget.

**Lemma 89.** *We consider the Two-Choices sub-phase following the Bit-Propagation sub-phase. For any period $\tau$ in the Two-Choices sub-phase or in the shuffle gadget following the Two-Choices sub-phase, we have*

$$\begin{aligned} &\left|\{v \in X_1(\tau_{\text{BP2}}) : |T_v'(\tau) - \tau| \leq 3c' \log n / \log\log n\}\right| \\ &\quad \geq |L(\tau_{\text{BP2}})| \cdot \frac{|X_1(\tau_{\text{BP1}})|}{|X(\tau_{\text{BP1}})|} \cdot (1 - o(1)) . \end{aligned}$$

*Similarly, for any color $\mathcal{C}_j$ we have*

$$\begin{aligned} &\left|\{v \in X_j(\tau_{\text{BP2}}) : |T_v'(\tau) - \tau| \leq 3c' \log n / \log\log n\}\right| \\ &\quad \leq |L(\tau_{\text{BP2}})| \cdot \frac{|X_j(\tau_{\text{BP1}})|}{|X(\tau_{\text{BP1}})|} \cdot (1 + o(1)) + O\left(n^{2/3}\right) . \end{aligned}$$

*Furthermore, it holds that*

$$\begin{aligned} |X_j(\tau_{\text{BP2}})| = &\left|\{v \in X_j(\tau_{\text{BP2}}) : |T_v'(\tau) - \tau| \leq 3c' \log n / \log\log n\}\right| \\ &\cdot (1 + o(1)) + O(\log n) , \end{aligned}$$

*with high probability.*

*Proof.* This lemma follows from Lemma 82 and Lemma 88. From Lemma 82 we know that all but an $\exp(-\Omega(\log n/\log\log n))$ fraction of the nodes that are alive after leaving the shuffle gadget preceding the Bit-Propagation sub-phase will be $3c'\log n/\log\log n$-close during the Bit-Propagation sub-phase as well as in the following shuffle gadget, and this also holds for every color separately. According to Lemma 88, every node samples a color $C_j$ with probability at least $|X_1(\tau_{\text{BP1}})|/|X(\tau_{\text{BP1}})|\cdot(1-\text{o}(1))$ for $j=1$ and with probability at most $|X_j(\tau_{\text{BP1}})|/|X(\tau_{\text{BP1}})|\cdot(1+\text{o}(1))+\text{O}\left(n^{-1/3}\right)$ otherwise, where this probability bounds hold independently, if the high probability statement of Lemma 88 is fulfilled. Applying then again Lemma 82 for the nodes which survive the shuffle gadget following the Bit-Propagation sub-phase, we obtain the lemma. □

## 19.5 The Endgame

In the remainder of this chapter, we analyze the simple asynchronous two-choices process which can be used to ensure that $\mathcal{A}$ wins after additional $\text{O}(n\log n)$ time steps in the sequential asynchronous model, one we have $a\geq({}^1\!/\!{}_2+\varepsilon_{\text{Part 1}})\cdot n$. The algorithm is essentially the same as specified in Algorithm 17.1, however the instruction is executed asynchronously upon each tick. To clearly separate the execution of the two-choices algorithm from the first part, we propose to start the second part with a do-nothing-block of length $\Theta(\log n)$ ticks. After these ticks, we can apply standard concentration bounds to observe that all nodes are synchronous up to constant factors. The rest of this section is structured as follows. In Lemma 90 we give a lower bound on the size of $\mathcal{A}$ throughout the rest of this execution of Algorithm 17.1. This lower bound on $\mathcal{A}$ allows to show that the number of nodes having a color different from $\mathcal{A}$ decreases quickly in expectation. This expected drop lets us apply standard drift theorem (Theorem 92) to obtain a bound on the required time until $\mathcal{A}$ prevails and all other colors vanish.

**Lemma 90.** *Fix an arbitrary tick $t$ and an arbitrary constant $c$. If $a_t\geq(1/2+3\varepsilon)n$ for some arbitrary constant $\varepsilon>0$, then for any $t'\leq t+c\cdot n\log n$ we have $a_{t'}\geq(1/2+\varepsilon)n$ with high probability.*

*Proof.* W.l.o.g. let $b_t=n-a_t$. We show by induction that for every $i\in[0,c\cdot\log n/\varepsilon]$ at time $t_i=i\cdot\varepsilon n$ that we have $a_{t_i}\geq(1/2+3\varepsilon)n-i\cdot(c'\sqrt{n}\cdot\log n)$ with high probability for some constant $c'$. Note that this implies the claim since $t_{i+1}-t_i\leq\varepsilon n$. Fix an $i$. Define the random variable

$$X_\tau=\begin{cases}1 & \text{with probability } b_\tau\cdot a_\tau^2/n^2\\ -1 & \text{with probability } a_\tau\cdot b_\tau^2/n^2\\ 0 & \text{otherwise.}\end{cases}$$

Define $Y_\tau = \sum_{k\le\tau} X_k$. We show that $Y_\tau$ is a sub-martingale.

$$\begin{aligned}
\mathrm{E}[Y_\tau|Y_{\tau-1},\dots,Y_1] &= Y_{\tau-1} + \mathrm{E}[X_\tau|Y_{\tau-1},\dots,Y_1] \\
&= Y_{\tau-1} - a_\tau\cdot b_\tau^2/n^2 + b_\tau a_\tau^2/n^2 \\
&= Y_{\tau-1} + a_\tau\cdot b_\tau/n^2(a_\tau - b_\tau) \\
&\ge Y_{\tau-1} \ ,
\end{aligned}$$

by induction $b_\tau \le a_\tau$. Applying the Azuma-Hoeffding bound to $Y_\tau$ gives us

$$\Pr\Big[Y_{t_{i+1}} - Y_{t_i} \le -c'\sqrt{n\cdot\log n}\Big] \le \exp\left(-\frac{c'^2 n\cdot\log n}{2\varepsilon n}\right) \ ,$$

which yields for sufficiently large $c'$ that the inductive steps holds with high probability. This completes the proof.

□

**Lemma 91.** *Fix an arbitrary tick $t$. Assume Process $P'$ where $a_{t'} \ge (1/2+\varepsilon)n$ for $\Theta(n\log n)$ for any $t' \le t+\Theta(n\log n)$. After $\Theta(n\log n)$ time steps all nodes have opinion $\mathcal{A}$ with high probability. Note that $\Theta(n\log n)$ ticks correspond to $\Theta(\log n)$ time steps.*

*Proof.* W.l.o.g. let $b_t = n - a_t$. We have

$$\begin{aligned}
E[b_{\tau+1}|\mathcal{F}_\tau] &= a_\tau\cdot b_\tau^2/n^2 - b_\tau a_\tau^2/n^2 \\
&= a_\tau\cdot b_\tau/n^2(b_\tau - a_\tau) \le \frac{b_\tau}{2n}(-2\varepsilon) \\
&= -\frac{\varepsilon\cdot b_\tau}{n} = \left(1-\frac{n+\varepsilon}{n}\right)b_\tau \ .
\end{aligned}$$

Define $\Phi(x_t) = b_t$. Note that $\Phi(x_{max}) \le n$ and in expectation we have $\mathrm{E}[\Phi(x_{t+1})|\Phi(x_t)] \le (1-\varepsilon/n)\Phi(x_t)$. Let $\tau$ be the first point in time where all nodes agree on color $\mathcal{A}$. We derive from Theorem 92 with parameters $\delta = 1$ and $\nu(n) = n/(n+\varepsilon)$ that $\Pr[\tau \ge n/\varepsilon(\ln\Phi(n) + c\cdot\ln n)] \le n^{-c}$. □

The following is a slightly simplified definition and slightly weaker version of theorem given in [DG13].

**Definition 28** ([DG13])**.** *Let $\nu : \mathbb{N} \to \mathbb{R}_{\ge 0}$ be monotonically increasing. We call $\Phi : \Omega_n \to \mathbb{R}_{\ge 0}$ a feasible $\nu$-drift function for an algorithm $\mathcal{A}$, if the following conditions are satisfied.*

1. *$\Phi(x) = 0$ for all $x \in \Omega_{opt}$;*
2. *$\Phi(x) \ge 1$ for all $x \in \Omega_n \setminus \Omega_{opt}$;*
3. *there exists a constant $\delta > 0$ (which is independent of $n$) such that for all $x_t \in \Omega_n \setminus \Omega_{opt}$*

$$\mathrm{E}[\Phi(x_{t+1})|\Phi(x_t)] \le (1-\delta/\nu(n))\Phi(x_t) \ .$$

**Theorem 92** ([DG13]). *Let $\Phi : \Omega_n \to \mathbb{R}$. Denote by $\Phi_{max} = \max\{\Phi(x) | x \in \Omega_n\}$ the maximum value of $\Phi$. If $\Phi$ is a feasible $\nu$-drift function (with implicit constant $\delta$) under Algorithm $\mathcal{A}$, then the stopping time $\tau = \min\{t \geq 0 : x_t \in \Omega_{opt}\}$, where $x_t$ is the state at time $t$, is at most*

$$\frac{\nu(n)}{\delta}(1 + \ln \Phi(x_{max})).$$

*Also, for any $c > 0$ (possibly depending on $n$), we have that*

$$\Pr\left[\tau \geq \frac{\nu(n)}{\delta}(\ln \Phi(x_{max}) + c \cdot \ln n)\right] \leq n^{-c}.$$

## 19.6 Proof of Theorem 63

We use Lemma 84, Lemma 89, and Lemma 91 to show Theorem 63, which is restated as follows.

**Theorem 63.** *Let $G = K_n$ be the complete graph with $n$ nodes. Let $k = \exp\left(\mathrm{O}\left(\log n / \log^2 \log n\right)\right)$ be the number of opinions. Let $\varepsilon_{\text{bias}} > 0$ be a constant. The asynchronous plurality consensus process* `AsyncPlurality` *defined in Algorithm 19.1 on $G$ converges within time $\Theta(\log n)$ to the majority opinion $\mathcal{A}$, with high probability, if $c_1 \geq (1 + \varepsilon_{\text{bias}}) \cdot c_i$ for all $i \geq 2$.*

*Proof.* From Lemma 84 we get for any color $\mathcal{C}_j$ that at the end of the first reference point of the Bit-Propagation sub-phases

$$|X_j(\tau_{\text{BP1}})| = |L(\tau_{\text{TC2}})| \cdot \frac{|\{v \in L(\tau_{\text{TC1}}) : v \in \mathcal{C}_j(\tau_{\text{TC1}})\}|^2}{n^2} \cdot (1 \pm \mathrm{o}(1)) + \mathrm{O}\left(\log^2 n\right) \tag{19.2}$$

and from Lemma 89 we observe that at the last reference point $\tau_T$ of the phase we get

$$|\{v \in L(\tau_T) : v \in \mathcal{C}_j(\tau_T)\}| = n \cdot \frac{|X_1(\tau_{\text{BP1}})|}{|X(\tau_{\text{BP1}})|} \cdot (1 \pm \mathrm{o}(1)) + \mathrm{O}\left(n^{2/3}\right) . \tag{19.3}$$

Combining (19.2) with (19.3) and applying Lemma 80 yields

$$|\{v \in L(\tau_T) : v \in \mathcal{C}_j(\tau_T)\}| = \frac{|\{v \in L(\tau_{\text{TC1}}) : v \in \mathcal{C}_j(\tau_{\text{TC1}})\}|^2}{|X(\tau_{\text{BP1}})|} \cdot (1 \pm \mathrm{o}(1)) + \mathrm{O}\left(n^{2/3}\right) . \tag{19.4}$$

We observe that (19.4) is the asynchronous counter part of Lemma 74 and essentially shows the desired quadratic increase in $\mathcal{A}$. Iterating the argument for all $\Theta(\log \log n)$ analogously to Theorem 62 shows that at the end of Algorithm 19.1 all but $\mathrm{o}(1) \cdot n$ nodes have opinion $\mathcal{A}$. In the remaining $\Theta(\log n)$ time steps of the simple two-choices protocol, we have by Lemma 91 that all nodes reach consensus after $\Theta(\log n)$ time steps. This completes the proof. $\square$

# 20 Simulation Results

In this chapter we present simulation results to support our theoretical findings. We implemented our simulation to run on a shared memory machine and simulate the distributed system.

First, we measured the run time until the process converged to $\mathcal{A}$. We set $k \approx \sqrt{n}$, $a - b \approx \sqrt{n \log n}$, and $c_2, \dots, c_k \approx n/k$, to simulate the processes for varying $n$. Our simulation results indicate that, as shown in Theorem 62, the memory based plurality consensus protocol outperforms the classical two-choices approach by orders of magnitude, see Figure 20.1.

Secondly, we investigated the behavior of the bits used in Chapter 18. We therefore plotted in Figure 20.2 the relative number of nodes which have a bit set and color $\mathcal{A}$ among all other nodes which have a bit set. That is, our plots show $x_1(t)/x(t)$ for every round $t$ on a complete graph of $n = 10^6$ nodes. Additionally, the relative number of nodes of color $\mathcal{A}$ is shown in the plot. The simulation indeed confirms an exponential growth of $\mathcal{A}$ during the bit propagation phase.

Finally, in Figure 20.3 we empirically analyzed the success rate for varying initial bias on a complete graph of $n = 10^6$ nodes with $k = 1000$ opinions. The success rate is the relative number of runs where $\mathcal{A}$ won over the total number of runs. In the plot, an initial bias of 0 means that $c_1 = \cdots = c_k$, while an initial bias of $x$ means that $c_1 = c_j + x \pm 1$ for $j \geq 2$. The simulation results indicate that the constant $z$ from Theorem 61 required for the initial bias towards $\mathcal{A}$ such that $\mathcal{A}$ wins with high probability is small. Similarly, for the memory-based approach the empirical success rate reaches 1 for a much smaller initial bias than the one predicted by the theoretical results. This may be due to the relatively small number of independent test runs $R \ll n$ for each data point.

Additionally, an asynchronous variant of the voting process described in Algorithm 20.1 has been simulated. For our simulation, we ran the process in the sequential asynchronous model. However, the run time shown in the plot is the parallel run time. Our simulation results empirically show that this simple asynchronous memory-based voting process performs very well for practical application.

**Acknowledgments.** The author would like to thank Gregor Bankhamer for helpful discussions and important hints.

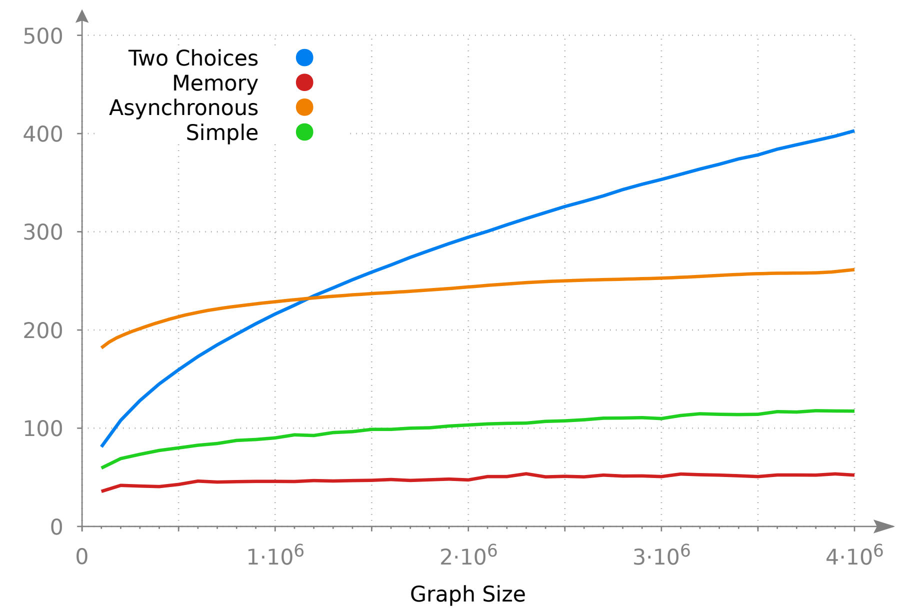


**Figure 20.1:** run times for the algorithms defined in Algorithm 17.1, Algorithm 18.1, Algorithm 19.1, and Algorithm 20.1 over the graph sizes

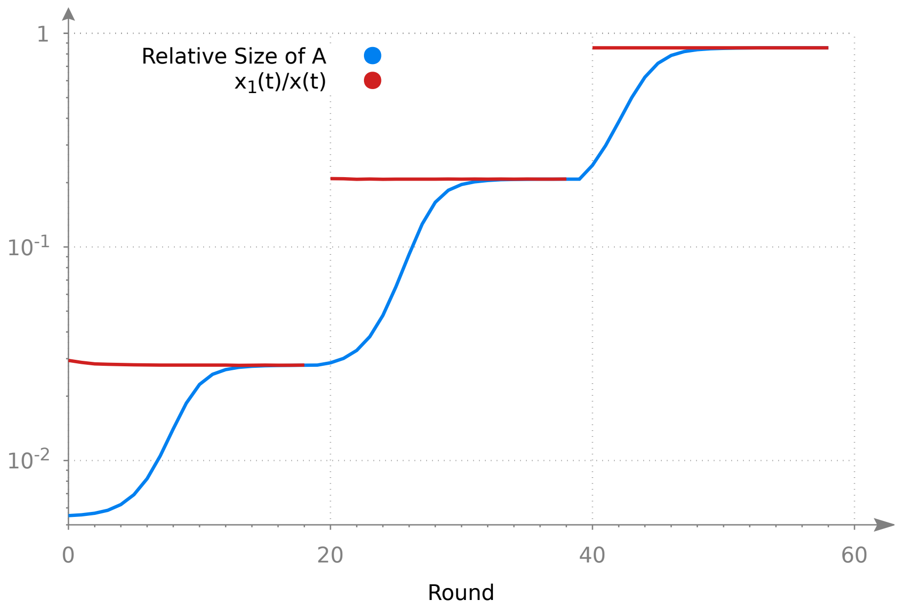


**Figure 20.2:** the relative number of bits and the relative size of the plurality color $\mathcal{A}$ during the execution of Algorithm 18.1

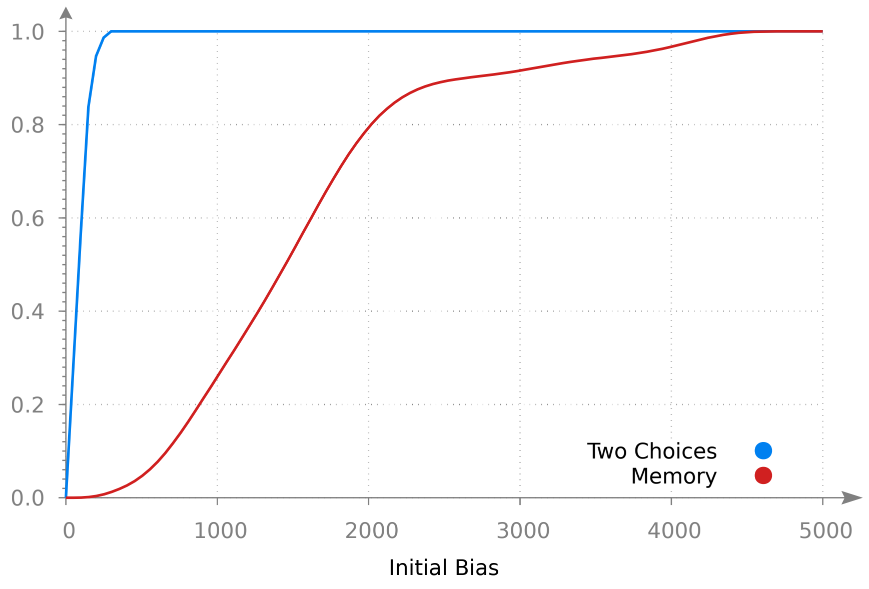


**Figure 20.3:** the success rate of the algorithms defined in Algorithm 17.1 and Algorithm 18.1 for varying initial bias $c_1 - c_2$

```
Algorithm asynchronous($G = (V, E)$; color : $V \to C$;
 bit : $V \to \{\textsc{True}, \textsc{False}\}$)
    at each node $v$ do asynchronously
        for phase $s = 1$ to $10 \cdot \log_2 |V|$ do
            let $u_1, u_2 \in N(v)$ uniformly at random;
            if color($u_1$) = color($u_2$) then
                color($v$) ← color($u_1$);
                bit($v$) ← True ;
            else
                bit($v$) ← False ;
            for 2 ticks do                          /* bit-propagation subphase  */
                let $u \in N(v)$ uniformly at random;
                if bit($u$) = True then
                    color($v$) ← color($u$);
                    bit($v$) ← True ;
```

**Algorithm 20.1:** simple asynchronous distributed voting protocol

# Conclusions and Outlook

# 21 Summary and Conclusions

In Part I, we have shown that for all-to-all communication using randomized gossiping protocols the results from the complete graph by Berenbrink et al. carry over to random graphs. We showed that it is possible to reduce the communication complexity at the cost increasing the run time. Interestingly, in the slightly weaker model by Chen and Pandurangan a contradicting lower bound has been shown. Also, for one-to-all communication in the same model, it had been shown that the performance of randomized broadcasting cannot be achieved in sparse graphs even if they have best expansion and connectivity properties. While the general proofs are in the same spirit as the results by Berenbrink et al., at various places an elaborate analysis was required to maintain the claims.

For our results on randomized gossiping algorithms, we also performed an empirical analysis, where we simulated a distributed system and measured the number of rounds and the total communication complexity, until all messages had been disseminated to all other nodes. Our empirical data directly reflect the benefits of the random walk approach described in Chapter 6. We would like to emphasize the results presented in Figure 8.1. The impact of our random walk approach is directly reflected in the run time of the corresponding protocol. This can be observed in the detailed view in Figure 21.1. Additionally, out empirical analysis does not only support our theoretical findings, but shows that the actual constants required to perform randomized gossiping are relatively small. We conclude that by carefully tuning these constants, one can obtain protocols that work very well in practice.

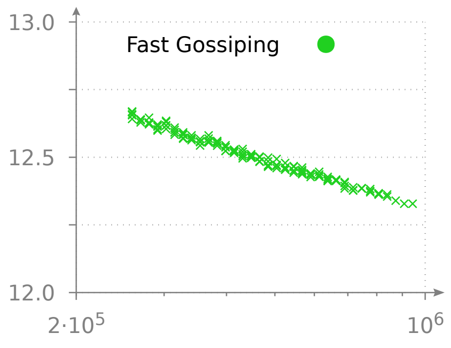


**Figure 21.1:** a detail from Figure 8.1

In Part II, we have presented an empirical analysis with focus on second order diffusion schemes for load balancing applications. Especially for torus graphs, our results for discrete diffusion based load balancing algorithms show a clear advantage of second order schemes over first order schemes.

We furthermore empirically analyzed the remaining load imbalance once the system has converged such that no node has more than a constant number of additional load tokens. We proposed to switch from SOS to FOS once this state is reached, and our simulations show that approach leads to a further drop of the remaining load imbalance.

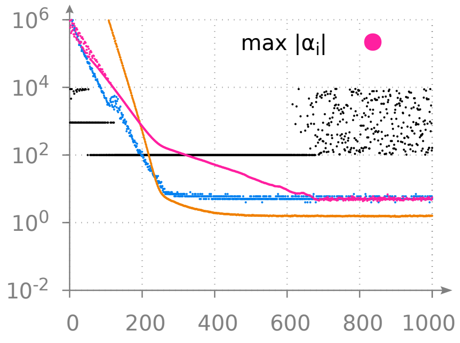


**Figure 21.2:** the impact of eigenvectors on the load, a detail from Figure 11.7

As an empirical contribution in this part, we would like to recall the following detail of a plot from Chapter 11 shown in Figure 21.2. It shows that the leading eigenvector, the eigenvector with the largest coefficient $\alpha_i$, governs the convergence rate of the load balancing process.

In Part III, we analyzed a synchronous distributed voting process, where each node adopts the majority opinion, black or white, among its neighbors in every round. The quantity of interest was the so-called *voting time*, the time until a two-periodic state of the process is reached. We gave a new upper bound on the voting time that could be computed in linear time and asymptotically improved the previously best known bound of $O(|E|)$ on many graph classes. Furthermore, we showed by a reduction of the corresponding decision problem from 3SAT that computing the voting time in general is NP hard. Finally, we analyzed various computational properties of the majority voting process w.r.t. the potential function used in proving the upper bounds on the voting time.

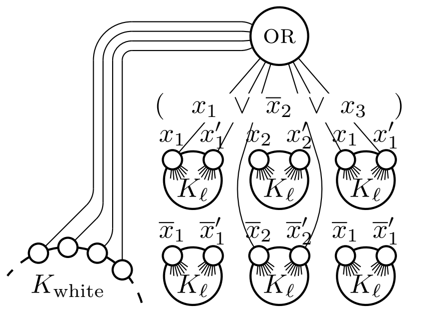

OR
( $x_1$ ∨ $\overline{x}_2$ ∨ $x_3$ )
$x_1$ $x_1'$ $x_2$ $x_2'$ $x_1$ $x_1'$
$K_\ell$
$\overline{x}_1$ $\overline{x}_1'$ $\overline{x}_2$ $\overline{x}_2'$ $\overline{x}_1$ $\overline{x}_1'$
$K_{\text{white}}$

**Figure 21.3:** a gadget from the reduction of 3SAT to VTPD from Figure 13.5

Finally, in Part IV we considered a plurality consensus process, where each node initially had one of $k$ possible opinions where $k = O(n^\varepsilon)$ for a small constant $\epsilon > 0$. We showed that for the two-choices process originally defined by Cooper et al. all nodes adopt the initial plurality opinion after at most $O(n/c_1 \cdot \log n)$ rounds, where $c_1$ is the size of the largest color, if initially the difference between the largest and the second largest color is at least $z \cdot \sqrt{n \log n}$ for some constant $z$.

Then, we slightly changed the model and devised an algorithm which is allowed to transmit one additional bit. We combined the benefits of the two-choices protocol with the power of information dissemination and obtained an algorithm which requires only a slightly larger initial additive bias towards the plurality opinion of $z \cdot \sqrt{n \log^3 n}$, but converges after only $\mathrm{O}\left(\log^2 n\right)$ rounds.

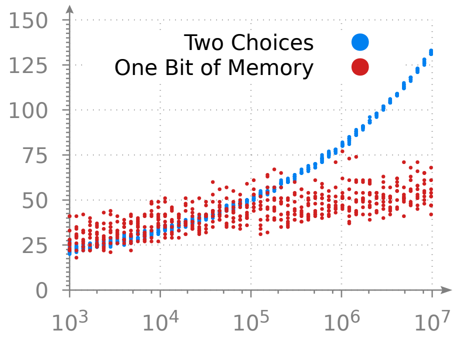


**Figure 21.4:** run time of the two-choices and the memory protocols from Figure 20.1

Finally, we adapted this algorithm to the asynchronous setting. In the asynchronous model, we assumed that nodes are equipped with a random clock that ticks according to a Poisson distribution once per time unit. For our analysis, we assumed the equivalent model in which nodes are selected uniformly at random in discrete time steps to perform their actions. In our model, we allowed nodes to communicate upon activation with at most a constant number of neighbors. We showed that the results from the memory model carry over to the asynchronous setting. Precisely, we showed that for $k = \exp\left(\mathrm{O}\left(\log n / \log^2 \log n\right)\right)$ colors and an initial bias towards the plurality color $\mathcal{C}_1$ such that $c_1 \geq (1 + \varepsilon_{\text{bias}})c_i$ for any other color $\mathcal{C}_i$ with $i \geq 2$ our algorithm solves plurality consensus and the process converges to color $\mathcal{C}_1$ in the best possible run time of $\Theta(\log n)$ time.

# 22 Open Problems

Finally, in this last chapter we identify various open problems related to the results presented in this thesis. In Part I, we presented our first result regarding the classical random phone call model for random graphs in the configuration model, while our second result regarding the memory model was presented for random graphs in the Erdős-Rényi model. For completeness, it would be interesting to show the first result in the Erdős-Rényi model and the second result for the configuration model as well. Also, it would be interesting to know whether these results could be further generalized, in the following sense. In our proofs, we extensively used the structural properties of random graphs. However, one interesting contribution would be to identify global graph properties, such as expansion or connectivity, and analyze whether these properties imply the same or similar results.

In the analysis of the gossiping algorithm, we furthermore discussed the behavior of random walks under congestion. That is, in our model we assumed that all nodes collect incoming random walks and send these random walks out one after the other, according to the random phone call model. Our result showed that for a rather small number of $O(n/\log n)$ random walks, each of these random walks performs at least $\Omega(\log n)$ steps in a period of length $\Theta(\log n)$ steps. The general problem, however, is of independent interest. In a related work[1], Becchetti et al. showed for $n$ tokens an upper bound on the maximum number of tokens which are enqueued at any node for the complete graph. An analysis of the behavior of this process for sparse random graphs for a large number of tokens, that is, $\Omega(n)$, would be of interest.

Regarding the results from Part II for diffusion based load balancing algorithms, we would like to note that any improvement on the bounds between the discrete and the idealized scheme would be interesting. However, to tighten

[1] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Gustavo Posta: *Self-Stabilizing Repeated Balls-into-Bins.* In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2015, pages 332–339.

these results, one would probably need different techniques. Therefore, any improvement would be a welcome contribution to the field of diffusion based load balancing.

Originally, we motivated load balancing with improved processing times of parallel computations in general and with finite element simulations in particular. We are not aware of any practical implementations of highly parallelized finite element solvers using load balancing based on discretized second order diffusion. Adapting an existing finite element software to use our load balancing algorithms would be, from the engineering point of view, an interesting, albeit possibly challenging endeavor.

Regarding the plurality consensus process presented in Part IV, there do exist various natural generalizations. It would be particularly interesting to generalize the results in the memory model to a broader range of graph classes and to dynamic networks. We believe that similar protocols should work on many graph classes in both, the synchronous and the asynchronous model, provided good expansion properties allow the rapid information dissemination to all other nodes. However, the resulting protocols might be vulnerable to an adversary who controls the initial assignment of opinions to nodes. It seems to be an interesting, albeit difficult, task to identify an initial bias which is large enough for the initial plurality color to win under presence of an adversary which may redistribute the opinions before the first round, or possibly in every round.

For the complete graph, an extension of the results to an even broader range of initial opinions should be possible. This intuition is supported by simulations of the two-choices protocol which were run successfully for $k = \sqrt{n}$.

Regarding the Check-Synchronicity procedure and the Shuffle Gadget used in the analysis of the asynchronous plurality consensus process in Chapter 19, we would like to note that these gadgets seem to be required only for the analysis. This observation is based on extensive simulations on up to $10^7$ nodes, which have essentially shown that a simple adaption of the synchronous algorithm to the asynchronous model without these gadgets achieves the same results. It therefore remains an open question whether our results could be maintained using a less complicated algorithm.

We nevertheless believe that the Check-Synchronicity procedure and the Shuffle Gadget are interesting in their own right. Therefore, a final open problem is to identify the generality of these procedures and to describe synchronous protocols in such a way that these gadgets can be used to construct population protocols from a broad class of synchronous algorithms. Such a general analysis could provide insights on the difficulties in bridging synchronous and asynchronous models.

**Acknowledgments.** The author would like to thank all anonymous reviewers who commented on the results presented in this thesis. Some of the open problems stated above are based on their feedback.

# Appendix

# Bibliography

[AD15] Mohammed Amin Abdullah and Moez Draief: *Global majority consensus by local majority polling on graphs of a given degree sequence*. In *Discrete Applied Mathematics*, volume 180, 2015, pages 1–10.

[AB12] Clemens P.J. Adolphs and Petra Berenbrink: *Distributed Selfish Load Balancing with Weights and Speeds*. In *Proc. PODC*, 2012, pages 135–144.

[ACL01] William Aiello, Fan Chung, and Linyuan Lu: *A Random Graph Model for Power Law Graphs*. In *Experimental Mathematics*, volume 10 (1), 2001, pages 53–66.

[ABEK14a] Hoda Akbari, Petra Berenbrink, Robert Elsässer, and Dominik Kaaser: *Discrete Load Balancing in Heterogeneous Networks with a Focus on Second-Order Diffusion*. In *CoRR*, volume abs/1412.7018, 2014. URL: http://arxiv.org/abs/1412.7018.

[ABEK14b] Hoda Akbari, Petra Berenbrink, Robert Elsässer, and Dominik Kaaser: *Load Balancing Visualization Video*. https://algorithms.cosy.sbg.ac.at/downloads/load-balancing-video.mkv. 2014.

[ABEK15] Hoda Akbari, Petra Berenbrink, Robert Elsässer, and Dominik Kaaser: *Discrete Load Balancing in Heterogeneous Networks with a Focus on Second-Order Diffusion*. In *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2015, pages 497–506. DOI: 10.1109/ICDCS.2015.57.

[ABS16] Hoda Akbari, Petra Berenbrink, and Thomas Sauerwald: *A Simple Approach for Adapting Continuous Load Balancing Processes to Discrete Settings*. In *Distributed Computing*, volume 29 (2), 2016, pages 143–161.

[AF02] David Aldous and James Allen Fill: *Reversible Markov Chains and Random Walks on Graphs*. Unpublished. http://www.stat.berkeley.edu/~aldous/RWG/book.html. 2002.

[AGV15] Dan Alistarh, Rati Gelashvili, and Milan Vojnović: *Fast and Exact Majority in Population Protocols*. In *Proc. PODC*, 2015, pages 47–56.

[AGGZ10] Dan Alistarh, Seth Gilbert, Rachid Guerraoui, and Morteza Zadimoghaddam: *How Efficient Can Gossip Be? (On the Cost of Resilient Information Exchange)*. In *Proc. ICALP*, 2010, pages 115–126.

[ABB+99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen: *LAPACK Users' Guide*, 3rd edition. SIAM, 1999.

[AAE08] Dana Angluin, James Aspnes, and David Eisenstat: *A simple population protocol for fast robust approximate majority*. In *Distributed Computing*, volume 21 (2), 2008, pages 87–102.

[AAER07] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert: *The computational power of population protocols*. In *Distributed Computing*, volume 20 (4), 2007, pages 279–304.

[AFJ06] Dana Angluin, Michael J. Fischer, and Hong Jiang: *Stabilizing Consensus in Mobile Networks*. In *Proc. DCOSS*, 2006, pages 37–50.

[AR07] James Aspnes and Eric Ruppert: *An Introduction to Population Protocols*. In *Bulletin of the EATCS*, volume 93, 2007, pages 98–117.

[AW04] Hagit Attiya and Jennifer Welch: *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*, 2nd edition. Wiley, 2004.

[ACF+15] Vincenzo Auletta, Ioannis Caragiannis, Diodato Ferraioli, Clemente Galdi, and Giuseppe Persiano: *Minority Becomes Majority in Social Networks*. In *Proc. WINE*, 2015, pages 74–88.

[BCN+15a] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Gustavo Posta: *Self-Stabilizing Repeated Balls-into-Bins*. In *Proc. SPAA*, 2015, pages 332–339.

[BCN+15b] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Riccardo Silvestri: *Plurality Consensus in the Gossip Model*. In *Proc. SODA*, 2015, pages 371–390.

[BCN+14] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, Riccardo Silvestri, and Luca Trevisan: *Simple Dynamics for Plurality Consensus*. In *Proc. SPAA*, 2014, pages 247–256.

[BCN+16] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan: *Stabilizing Consensus with Many Opinions*. In *Proc. SODA*, 2016, pages 620–635.

[BC78] Edward A. Bender and E. Rodney Canfield: *The Asymptotic Number of Labeled Graphs with Given Degree Sequences.* In *Journal of Combinatorial Theory, Series A*, volume 24 (3), 1978, pages 296–307.

[BTV09] Florence Bénézit, Patrick Thiran, and Martin Vetterli: *Interval consensus: From quantized gossip to voting.* In *Proc. ICASSP*, 2009, pages 3661–3664.

[BCO+14] Itai Benjamini, Siu-On Chan, Ryan O'Donnell, Omer Tamuz, and Li-Yang Tan: *Convergence, unanimity and disagreement in majority dynamics on unimodular graphs and random graphs.* In *CoRR*, volume abs/1405.2486, 2014. URL: http://arxiv.org/abs/1405.2486.

[BCF+15] Petra Berenbrink, Colin Cooper, Tom Friedetzky, Tobias Friedrich, and Thomas Sauerwald: *Randomized diffusion for indivisible loads.* In *Journal of Computer and System Sciences*, volume 81 (1), 2015, pages 159–185.

[BCEG10] Petra Berenbrink, Jurek Czyzowicz, Robert Elsässer, and Leszek Gąsieniec: *Efficient Information Exchange in the Random Phone-Call Model.* In *Proc. ICALP*, 2010, pages 127–138.

[BEF08] Petra Berenbrink, Robert Elsässer, and Tom Friedetzky: *Efficient Randomised Broadcasting in Random Regular Networks with Applications in Peer-to-Peer Systems.* In *Proc. PODC*, 2008, pages 155–164.

[BES14] Petra Berenbrink, Robert Elsässer, and Thomas Sauerwald: *Communication Complexity of Quasirandom Rumor Spreading.* In *Algorithmica*, 2014, pages 1–26.

[BFGK16] Petra Berenbrink, Tom Friedetzky, George Giakkoupis, and Peter Kling: *Efficient Plurality Consensus, or: The benefits of cleaning up from time to time.* In *Proc. ICALP*, 2016.

[BFH09] Petra Berenbrink, Tom Friedetzky, and Zengjian Hu: *A new analytical method for parallel, diffusion-type load balancing.* In *Journal of Parallel and Distributed Computing*, volume 69 (1), 2009, pages 54–61.

[BFK+16] Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, and Chris Wastell: *Plurality Consensus via Shuffling: Lessons Learned from Load Balancing.* In *CoRR*, volume abs/1602.01342, 2016. URL: http://arxiv.org/abs/1602.01342.

[BGKM16] Petra Berenbrink, George Giakkoupis, Anne-Marie Kermarrec, and Frederik Mallmann-Trenn: *Bounds on the Voter Model in Dynamic Networks.* In *Proc. ICALP*, 2016.

[Ber01] Eli Berger: *Dynamic Monopolies of Constant Size.* In *Journal of Combinatorial Theory, Series B*, volume 83 (2), 2001, pages 191–200.

[BT89] Dimitri Bertsekas and John Tsitsiklis: *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989.

[BG03] Andreas Blass and Yuri Gurevich: *Algorithms: A Quest for Absolute Definitions.* In *Bulletin of the EATCS*, volume 81, 2003, pages 195–225.

[Boi90] J. E. Boillat: *Load balancing and Poisson equation in a graph.* In *Concurrency and Computation: Practice and Experience*, volume 2 (4), 1990, pages 289–313.

[Bol80] Béla Bollobás: *A Probabilistic Proof of an Asymptotic Formula for the Number of Labelled Regular Graphs.* In *European Journal of Combinatorics*, volume 1 (4), 1980, pages 311–316.

[Bol01] Béla Bollobás: *Random Graphs*, 2nd edition. Cambridge University Press, 2001.

[BW01] Sherif Botros and Steve Waterhouse: *Search in JXTA and Other Distributed Networks.* In *Proc. P2P*, 2001, pages 30–35.

[BGPS06] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah: *Randomized Gossip Algorithms.* In *IEEE Transactions on Information Theory*, volume 52 (6), 2006, pages 2508–2530.

[BM91] Carl B. Boyer and Uta C. Merzbach: *A History of Mathematics*, 2nd edition. Wiley, 1991.

[BMPS04] Siddhartha Brahma, Sandeep Macharla, Sudebkumar Prasant Pal, and Sudhir Kumar Singh: *Fair Leader Election by Randomized Voting.* In *Proc. ICDCIT*, 2004, pages 22–31.

[CC12] Luca Cardelli and Attila Csikász-Nagy: *The Cell Cycle Switch Computes Approximate Majority.* In *Scientific Reports*, volume 2 (656), 2012.

[CS12] Keren Censor-Hillel and Hadas Shachnai: *Fast Information Spreading in Graphs with Large Weak Conductance.* In *SIAM J. on Computing*, volume 41 (6), 2012, pages 1451–1465.

[CP12] Jen-Yeu Chen and Gopal Pandurangan: *Almost-Optimal Gossip-Based Aggregate Computation.* In *SIAM Journal on Computing*, volume 41 (3), 2012, pages 455–483.

[CKO13] Flavio Chierichetti, Jon Kleinberg, and Sigal Oren: *On Discrete Preferences and Coordination.* In *Proc. EC*, 2013, pages 233–250.

[CLP10] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi: *Rumour Spreading and Graph Conductance.* In *Proc. SODA*, 2010, pages 1657–1663.

[CDFR16] Colin Cooper, Martin Dyer, Alan Frieze, and Nicolás Rivera: *Discordant voting processes on finite graphs.* In *CoRR*, volume abs/1604.06884, 2016. URL: http://arxiv.org/abs/1604.06884.

[CEOR13] Colin Cooper, Robert Elsässer, Hirotaka Ono, and Tomasz Radzik: *Coalescing Random Walks and Voting on Connected Graphs.* In *SIAM Journal on Discrete Mathematics*, volume 27 (4), 2013, pages 1748–1758.

[CER14] Colin Cooper, Robert Elsässer, and Tomasz Radzik: *The Power of Two Choices in Distributed Voting.* In *Proc. ICALP*, 2014, pages 435–446.

[CER+15] Colin Cooper, Robert Elsässer, Tomasz Radzik, Nicolás Rivera, and Takeharu Shiraga: *Fast Consensus for Voting on General Expander Graphs.* In *Proc. DISC*, 2015, pages 248–262.

[CFR09] Colin Cooper, Alan Frieze, and Tomasz Radzik: *Multiple Random Walks in Random Regular Graphs.* In *SIAM Journal on Discrete Mathematics*, volume 23 (4), 2009, pages 1738–1761.

[CRRS16] Colin Cooper, Tomasz Radzik, Nicolas Rivera, and Takeharu Shiraga: *Fast plurality consensus in regular expanders.* In *CoRR*, volume abs/1605.08403, 2016. URL: http://arxiv.org/abs/1605.08403.

[CG10] Gennaro Cordasco and Luisa Gargano: *Community Detection via Semi-Synchronous Label Propagation Algorithms.* In *Proc. BASNA*, 2010, pages 1–8.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: *Introduction to Algorithms*, 3rd edition. MIT Press, 2009.

[CH94] Alain Cournier and Michel Habib: *A New Linear Algorithm for Modular Decomposition.* In *Proc. CAAP*, 1994, pages 68–84.

[CG14] James Cruise and Ayalvadi Ganesh: *Probabilistic consensus via polling and majority rules.* In *Queueing Systems*, volume 78 (2), 2014, pages 99–120.

[CDS80] Dragos Cvetkovic, Michael Doob, and Horst Sachs: *Spectra of graphs: Theory and application.* Academic Press, 1980.

[Cyb89] George Cybenko: *Dynamic Load Balancing for Distributed Memory Multiprocessors.* In *Journal of Parallel and Distributed Computing*, volume 7 (2), 1989, pages 279–301.

[DGH+87] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry: *Epidemic Algorithms for Replicated Database Maintenance.* In *Proc. PODC*, 1987, pages 1–12.

[DP94] Xiaotie Deng and Christos Papadimitriou: *On the Complexity of Cooperative Solution Concepts.* In *Mathematics of Operations Research*, volume 19 (2), 1994, pages 257–266.

[DFM99] Ralf Diekmann, Andreas Frommer, and Burkhard Monien: *Efficient schemes for nearest neighbor load balancing.* In *Parallel Computing*, volume 25 (7), 1999, pages 789–812.

[DFF11] Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich: *Social Networks Spread Rumors in Sublogarithmic Time.* In *Proc. STOC*, 2011, pages 21–30.

[DFS09] Benjamin Doerr, Tobias Friedrich, and Thomas Sauerwald: *Quasirandom Rumor Spreading: Expanders, Push vs. Pull, and Robustness.* In *Proc. ICALP*, 2009, pages 366–377.

[DG13] Benjamin Doerr and Leslie Ann Goldberg: *Adaptive Drift Analysis.* In *Algorithmica*, volume 65 (1), 2013, pages 224–250.

[DGM+11] Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler: *Stabilizing Consensus With the Power of Two Choices.* In *Proc. SPAA*, 2011, pages 149–158.

[DW83] Peter Donnelly and Dominic Welsh: *Finite particle systems and infection models.* In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 94 (1), 1983, pages 167–182.

[Dot14] David Doty: *Timing in Chemical Reaction Networks.* In *Proc. SODA*, 2014, pages 772–784.

[DV12] Moez Draief and Milan Vojnović: *Convergence speed of binary interval consensus.* In *SIAM Journal on Control and Optimization*, volume 50 (3), 2012, pages 1087–1109.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi: *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press, 2009.

[DR98] Devdatt Dubhashi and Desh Ranjan: *Balls and Bins: A Study in Negative Dependence.* In *Random Structures & Algorithms*, volume 13 (2), 1998, pages 99–124.

[Els06] Robert Elsässer: *On the communication complexity of randomized broadcasting in random-like graphs.* In *Proc. SPAA*, 2006, pages 148–157.

[EK15] Robert Elsässer and Dominik Kaaser: *On the Influence of Graph Density on Randomized Gossiping.* In *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2015, pages 521–531. DOI: 10.1109/IPDPS.2015.32.

[EM03] Robert Elsässer and Burkhard Monien: *Load Balancing of Unit Size Tokens and Expansion Properties of Graphs.* In *Proc. SPAA*, 2003, pages 266–273.

[EMP02] Robert Elsässer, Burkhard Monien, and Robert Preis: *Diffusion Schemes for Load Balancing on Heterogeneous Networks.* In *Theory of Computing Systems*, volume 35 (3), 2002, pages 305–320.

[EMS06] Robert Elsässer, Burkhard Monien, and Stefan Schamberger: *Distributing Unit Size Workload Packages in Heterogeneous Networks.* In *Journal of Graph Algorithms and Applications*, volume 10 (1), 2006, pages 51–68.

[ES08] Robert Elsässer and Thomas Sauerwald: *The Power of Memory in Randomized Broadcasting.* In *Proc. SODA*, 2008, pages 218–227.

[ES09] Robert Elsässer and Thomas Sauerwald: *Cover Time and Broadcast Time.* In *Proc. STACS*, 2009, pages 373–384.

[ES10] Robert Elsässer and Thomas Sauerwald: *Discrete Load Balancing is (almost) as Easy as Continuous Load Balancing.* In *Proc. PODC*, 2010, pages 346–354.

[ER59] Paul Erdős and Alfréd Rényi: *On random graphs.* In *Publicationes Mathematicae (Debrecen)*, volume 6, 1959, pages 290–297.

[Eri14] Jeff Erickson: *Algorithms and Models of Computation.* lecture notes. 2014. URL: `http://jeffe.cs.illinois.edu/teaching/algorithms/`.

[FPRU90] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal: *Randomized broadcast in networks.* In *Random Structures & Algorithms*, volume 1 (4), 1990, pages 447–460.

[Fel68] William Feller: *An Introduction to Probability Theory and Its Applications*, 3rd edition. Wiley, 1968.

[FHP10] Nikolaos Fountoulakis, Anna Huber, and Konstantinos Panagiotou: *Reliable broadcasting in random networks and the effect of density.* In *Proc. INFOCOM*, 2010, pages 1–9.

[FWM94] Geoffrey C. Fox, Roy D. Williams, and Paul C. Messina: *Parallel Computing Works!* Morgan Kaufmann, 1994.

[FL94] Pierre Fraigniaud and Emmanuel Lazard: *Methods and problems of communication in usual networks.* In *Discrete Applied Mathematics*, volume 53 (1–3), 1994, pages 79–133.

[FGS12] Tobias Friedrich, Marti Gairing, and Thomas Sauerwald: *Quasirandom Load Balancing.* In *SIAM Journal on Computing*, volume 41 (4), 2012, pages 747–771.

[FS09] Tobias Friedrich and Thomas Sauerwald: *Near-Perfect Load Balancing by Randomized Rounding.* In *Proc. STOC*, 2009, pages 121–130.

[FKW13] Silvio Frischknecht, Barbara Keller, and Roger Wattenhofer: *Convergence in (Social) Influence Networks*. In *Proc. DISC*, 2013, pages 433–446.

[GP16] Mohsen Ghaffari and Merav Parter: *A Polylogarithmic Gossip Algorithm for Plurality Consensus*. In *Proc. PODC*, 2016.

[GM96] Bhaskar Ghosh and S. Muthukrishnan: *Dynamic Load Balancing by Random Matchings*. In *Journal of Computer and System Sciences*, volume 53, 3 1996, pages 357–370.

[Gia11] George Giakkoupis: *Tight bounds for rumor spreading in graphs of a given conductance*. In *Proc. STACS*, 2011, pages 57–68.

[Gia14] George Giakkoupis: *Tight Bounds for Rumor Spreading with Vertex Expansion*. In *Proc. SODA*, 2014, pages 801–815.

[GS12] George Giakkoupis and Thomas Sauerwald: *Rumor Spreading and Vertex Expansion*. In *Proc. SODA*, 2012, pages 1623–1641.

[Gif79] David Gifford: *Weighted Voting for Replicated Data*. In *Proc. SOSP*, 1979, pages 150–162.

[Gnu] Gnutella: *The Annotated Gnutella Protocol Specification v0.4*. URL: http://rfc-gnutella.sourceforge.net/developer/stable/index.html.

[Gol89] Eric Goles: *Local Graph Transformations Driven by Lyapunov Functionals*. In *Complex Systems*, volume 3 (1), 1989, pages 173–184.

[GM90] Eric Goles and Servet Martínez: *Neural and Automata Networks*. Kluwer, 1990.

[GO88] Eric Goles and Andrew M. Odlyzko: *Decreasing Energy Functions and Lengths of Transients for Some Cellular Automata*. In *Complex Systems*, volume 2 (5), 1988, pages 501–507.

[GO80] Eric Goles and J. Olivos: *Periodic behaviour of generalized threshold functions*. In *Discrete Mathematics*, volume 30 (2), 1980, pages 187–189.

[GFP85] Eric Goles-Chacc, Françoise Fogelman-Soulie, and Didier Pellegrin: *Decreasing energy functions as a tool for studying threshold networks*. In *Discrete Applied Mathematics*, volume 12 (3), 1985, pages 261–277.

[GV61] Gene H. Golub and Richard S. Varga: *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods*. In *Numerische Mathematik*, volume 3 (1), 1961, pages 147–156.

[Hae12] Bernhard Haeupler: *Tighter Worst-Case Bounds on Algebraic Gossip*. In *IEEE Communications Letters*, volume 16 (8), 2012, pages 1274–1276.

[Hae13] Bernhard Haeupler: *Simple, Fast and Deterministic Gossip and Rumor Spreading.* In *Proc. SODA*, 2013, pages 705–716.

[HR90] Torben Hagerup and Christine Rüb: *A guided tour of chernoff bounds.* In *Information Processing Letters*, volume 33 (6), 1990, pages 305–308.

[HP01] Yehuda Hassin and David Peleg: *Distributed Probabilistic Polling and Applications to Proportionate Agreement.* In *Information and Computation*, volume 171 (2), 2001, pages 248–268.

[Hea08] Thomas L. Heath: *The Thirteen Books of Euclid's Elements.* Vol. 1–3. Cambridge University Press, 1908.

[HL75] Richard Holley and Thomas Liggett: *Ergodic Theorems for Weakly Interacting Infinite Systems and the Voter Model.* In *The Annals of Probability*, volume 3 (4), 1975, pages 643–663.

[HKP+05] 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, 2005.

[Joh89] Barry W. Johnson, ed.: *Design & Analysis of Fault Tolerant Digital Systems.* Addison-Wesley, 1989.

[JK77] Norman Lloyd Johnson and Samuel Kotz: *Urn Models and Their Application: An Approach to Modern Discrete Probability Theory.* Wiley, 1977.

[KMN15] Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale: *Brief Announcement: On the Voting Time of the Deterministic Majority Process.* In *Proceedings of the 29th International Symposium on Distributed Computing (DISC)*, 2015.

[KMN16] Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale: *On the Voting Time of the Deterministic Majority Process.* In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2016. DOI: `10.4230/LIPIcs.MFCS.2016.55`.

[KSSV00] Richard Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking: *Randomized Rumor Spreading.* In *Proc. FOCS*, 2000, pages 565–574.

[KPW14] Barbara Keller, David Peleg, and Roger Wattenhofer: *How Even Tiny Influence Can Have a Big Impact!* In *Proc. FUN*, 2014, pages 252–263.

[KDG03] David Kempe, Alin Dobra, and Johannes Gehrke: *Gossip-Based Computation of Aggregate Information.* In *Proc. FOCS*, 2003, pages 482–491.

[KMG03] Anne-Marie Kermarrec, Laurent Massoulie, and Ayalvadi J. Ganesh: *Probabilistic Reliable Dissemination in Large-Scale Systems*. In *IEEE Transactions on Parallel and Distributed Systems*, volume 14 (3), 2003, pages 248–258.

[Knu97] Donald E. Knuth: *The Art of Computer Programming*, 3rd edition. Vol. 1: Fundamental Algorithms. Addison-Wesley, 1997.

[KPS13] Kishore Kothapalli, Sriram Pemmaraju, and Vivek Sardeshmukh: *On the Analysis of a Label Propagation Algorithm for Community Detection*. In *Proc. ICDCN*, 2013, pages 255–269.

[KS08] Ajay D. Kshemkalyani and Mukesh Singhal: *Distributed Computing. Principles, Algorithms, and Systems*. Cambridge, 2008.

[LN07] Nicolas Lanchier and Claudia Neuhauser: *Voter model and biased voter model in heterogeneous environments*. In *Journal of Applied Probability*, volume 44 (3), 2007, pages 770–787.

[Lig12] Thomas Liggett: *Interacting particle systems*. Springer Science & Business Media, 2012.

[Lig85] Thomas M. Liggett: *Interacting Particle Systems*. Springer, 1985.

[LM15] Yuezhou Lv and Thomas Moscibroda: *Local Information in Influence Networks*. In *Proc. DISC*, 2015, pages 292–308.

[Mal14] F. Mallmann-Trenn: *Bounds on the voting time in terms of the conductance*. Master's thesis. http://summit.sfu.ca/item/14502. MA thesis. Simon Fraser University, 2014.

[MNRS14] 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 *Proc. ICALP*, 2014, pages 871–882.

[Mil76] Gary L. Miller: *Riemann's Hypothesis and Tests for Primality*. In *Journal of Computer and System Sciences*, volume 13 (3), 1976.

[MU05] Michael Mitzenmacher and Eli Upfal: *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[MS06] Damon Mosk-Aoyama and Devavrat Shah: *Computing Separable Functions via Gossip*. In *Proc. PODC*, 2006, pages 113–122.

[MT14] Elchanan Mossel and Omer Tamuz: *Opinion Exchange Dynamics*. In *CoRR*, volume abs/1401.4770, 2014. URL: http://arxiv.org/abs/1401.4770.

[MR95] Rajeev Motwani and Prabhakar Raghavan: *Randomized Algorithms*. Cambridge University Press, 1995.

[MGS98] S. Muthukrishnan, B. Ghosh, and M. H. Schultz: *First- and Second-Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing.* In *Theory of Computing Systems*, volume 31 (4), 1998, pages 331–354.

[NIY99] Toshio Nakata, Hiroshi Imahayashi, and Masafumi Yamashita: *Probabilistic Local Majority Voting for the Agreement Problem on Finite Graphs.* In *Proc. COCOON*, 1999, pages 330–338.

[Oli12] Roberto I. Oliveira: *On the coalescence time of reversible random walks.* In *Transactions of the American Mathematical Society*, volume 364 (4), 2012, pages 2109–2128.

[Pel02] David Peleg: *Local majorities, coalitions and monopolies in graphs: a review.* In *Theoretical Computer Science*, volume 282 (2), 2002, pages 231–257.

[Pel14] David Peleg: *Immunity against Local Influence.* In *Language, Culture, Computation. Computing - Theory and Technology*, volume 8001, LNCS. Springer, 2014, pages 168–179.

[PVV09] Etienne Perron, Dinkar Vasudevan, and Milan Vojnović: *Using Three States for Binary Consensus on Complete Graphs.* In *Proc. INFOCOM*, 2009, pages 2527–2535.

[Pit87] Boris Pittel: *Linear Probing: The Probable Largest Search Time Grows Logarithmically with the Number of Records.* In *Journal of Algorithms*, volume 8 (2), 1987, pages 236–249.

[PS83] Svatopluk Poljak and Miroslav Sûra: *On periodical behaviour in societies with symmetric influences.* In *Combinatorica*, volume 3 (1), 1983, pages 119–121.

[PT86] Svatopluk Poljak and Daniel Turzík: *On an application of convexity to discrete systems.* In *Discrete Applied Mathematics*, volume 13 (1), 1986, pages 27–32.

[RS98] Martin Raab and Angelika Steger: *"Balls into Bins" — A Simple and Tight Analysis.* In *Proc. RANDOM*, 1998, pages 159–170.

[RSW98] Yuval Rabani, Alistair Sinclair, and Rolf Wanka: *Local Divergence of Markov Chains and the Analysis of Iterative Load-Balancing Schemes.* In *Proc. FOCS*, 1998, pages 694–703.

[Rab80] Michael O. Rabin: *Probabilistic Algorithm for Resting Primality.* In *Journal of Number Theory*, volume 12 (1), 1980.

[RAK07] Usha Raghavan, Réka Albert, and Soundar Kumara: *Near linear time algorithm to detect community structures in large-scale networks.* In *Physical Review E*, volume 76 (3), 2007, pages 036106.

[SSKÇ13] Ahmet Erdem Sarıyüce, Erik Saule, Kamer Kaya, and Ümit V. Çatalyürek: *Shattering and Compressing Networks for Betweenness Centrality.* In *Proc. SDM*, 2013, pages 686–694.

[SS12] Thomas Sauerwald and He Sun: *Tight Bounds for Randomized Load Balancing on Arbitrary Network Topologies.* In *Proc. FOCS*, 2012, pages 341–350.

[TT15] Omer Tamuz and Ran J. Tessler: *Majority Dynamics and the Retention of Information.* In *Israel Journal of Mathematics*, volume 206 (1), 2015, pages 483–507.

[Win08a] Peter Winkler: *Puzzled: Delightful Graph Theory.* In *Communications of the ACM*, volume 51 (8), 2008, pages 104.

[Win08b] Peter Winkler: *Puzzled: Solutions and Sources.* In *Communications of the ACM*, volume 51 (9), 2008, pages 103.

[Wor81a] Nicholas C. Wormald: *The asymptotic connectivity of labelled regular graphs.* In *Journal of Combinatorial Theory, Series B*, volume 31 (2), 1981, pages 156–167.

[Wor81b] Nicholas C. Wormald: *The asymptotic distribution of short cycles in random regular graphs.* In *Journal of Combinatorial Theory, Series B*, volume 31 (2), 1981, pages 168–182.

[Wor99] Nicholas C. Wormald: *Models of Random Regular Graphs.* In *Surveys in Combinatorics, 1999*, volume 267, London Mathematical Society Lecture Note Series. Cambridge University Press, 1999, pages 239–298.

# List of Figures

# List of Tables

# List of Algorithms

# List of Definitions

# List of Theorems