Sharing Resources in Distributed Systems

by

Rajmohan Rajaraman, B.Tech., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 1997

Sharing Resources in Distributed Systems

Approved by
Dissertation Committee:

To Ajuma and Mummy

Acknowledgments

First and foremost, I am indebted to my dissertation supervisor Greg Plaxton, who has been a true friend, philosopher, and guide ever since I have known him. A large part of what I have learnt in graduate school is due to Greg. His brilliance, enthusiasm, perseverance (exemplified by a 12-hour marathon meeting we once had), and his yearning for excellence are inspiring and I hope some of this has rubbed off on me.

The members of my dissertation committee, Bruce Maggs, Vijaya Ramachandran, Martin Wong, and David Zuckerman, have contributed a lot, either directly or indirectly, to my dissertation. Some of the key ideas that led to the result on static load balancing of Chapter 4 are due to Bruce Maggs. Bruce also made several useful comments that have significantly improved the presentation of Chapter 4. I would like to thank Vijaya Ramachandran for her insightful comments regarding the results of Chapter 3 and for the numerous hours she has devoted for discussing my doctoral work. I am grateful to Martin Wong, with whom I co-authored my first publication, for introducing me into computer science research. David Zuckerman has taught me a lot about randomness (in part, through his puzzles and card tricks). David also co-authored the static load balancing result of Chapter 4.

During the course of my graduate work, I have been fortunate to have collaborated with several other researchers who have also become close friends over these years. Johannes Gehrke made valuable contributions to the work concerning load balancing on rings (Chapter 5). Phil Mackenzie co-authored the work on contention resolution (Chapter 2). I am also grateful to Phil for my internship at Sandia National Labs during the summer of 1996. S. Muthukrishnan and Andréa Richa worked with me on the static load balancing result of Chapter 4. Muthu also came up with some of the main ideas in the dynamic load balancing result of Chapter 6. I would like to thank Torsten Suel for being always available for advice and discussions.

I have had very nice officemates who have tolerated my clumsiness which included regular encroachments of my material upon their desks. Special thanks to Shashidhar Thakur with whom I shared a wonderful office in Painter Hall for two years. Shashidhar and I had uncountable puzzle-solving sessions and I am glad that he is crunching out puzzles even after joining industry. The long discussions with the blokes at UA9, especially Rajeev Joshi, Kedar Namjoshi, and Carlos Puchol, and the hundreds of gallons of coffee that accompanied these discussions, will continue to stimulate me for a long time.

My six-year stint in Austin has been memorable thanks to some very special friends, but for whom this dissertation would have been completed in half the time (with apologies to P. G. Wodehouse)! All of my apartment mates over the years have been fun. I would especially remember Pawan Goyal and his culinary skills, Sanjay Gupta and his puns, Dinesh Muzumdar and our basketball duels, and Rahul Singh and his fixing abilities. The interminable debates with my real apartment mates and "virtual apartment mates" Om Damani, Rajesh Govindan, Divas Sanwal, Prashant Shenoy, and Ashis Tarafdar made me take the NBA slogan "Stay in School" a little too seriously. The company of Sanjay Agarwal, Madhukar Korupolu, Nishant Mittal, Chandramouli Nagarajan, Analbhai Shah, Renu Tewari, and Kishore Yellepeddy produced several memorable moments. I will always be nostalgic about the blood, sweat, tears, and the occasional baskets that were produced by our basketball gang during the numerous jam sessions we had this year.

Of course, I would not have been able to do any of the above without the financial support provided by NSF, Sandia National Labs, Schlumberger Corporation, TARP, and University of Texas at Austin. My sincere thanks to Gloria Ramirez, our graduate secretary, whose phenomenal efficiency can make even the most incompetent student meet their deadlines.

Undoubtably, the biggest credit goes to my family, especially my mother and my brother Rajesh, whose long-distance support contributed significantly to both my morale and the telephone industry. There was no better tonic for uplifiting my spirits than the royal treatment meted out by my folks at Calcutta and the nonstop encouragement provided by my sister-in-law Deepika.

Rajmohan Rajaraman

The University of Texas at Austin
December 1997

Sharing Resources in Distributed Systems

Publication No. ________________

Rajmohan Rajaraman, Ph.D.
The University of Texas at Austin, 1997

Supervisor: C. Greg Plaxton

An important goal of a distributed system is to effectively utilize the collective resources of the system, namely, the memory and the processors of the individual nodes. This dissertation addresses certain problems pertaining to sharing memory and processors in distributed systems.

In the first part of the dissertation, we study two important issues that arise while sharing memory in a distributed system: memory contention and faults. We adopt a model of computation in which each node can send or receive at most a constant number of objects per step. Our main result is a simple protocol for providing fast access to shared objects in an environment in which memory contention is unconstrained and a constant fraction of the nodes and communication links may be faulty at any time. Our protocol combines techniques for hashing and erasure codes with a new mechanism for on-line replication. We show that if each new access request is chosen according to a fixed probability distribution over the set of objects, then our protocol rapidly reaches a steady state in which each request is satisfied in an expected constant number of steps and the throughput of the protocol is asymptotically optimal. We also prove that the protocol continues to remain in a steady state if changes in the access pattern are moderate.

The second part of the dissertation studies load balancing, which is a mechanism for sharing processors in a distributed system. We analyze the effectiveness of a local load balancing algorithm in which each node repeatedly balances its load with its neighbors. Our main results concern the static version of the problem where we assume that the total load does not change with time. We analyze the local balancing algorithm in terms of both the imbalance of the initial load distribution and several parameters of the network, including the number of nodes, the maximum degree, and the edge expansion. We show that the algorithm is worst-case optimal for all networks. We improve this result for the special case of ring networks by showing that the local balancing approach is optimal (up to an additive linear term) for every initial distribution of load on the ring. Our results are also shown to hold for an asynchronous model of computation.

This dissertation demonstrates that a number of basic resource sharing problems admit efficient solutions in the form of simple local algorithms. Our algorithms are simple in the sense that the program running at each node can be expressed as a periodic process that repeatedly executes a small number of fixed operations. Our algorithms are local in the sense that each node either communicates with only its nearest neighbors or sends messages to only a small number of other nodes.

Contents

Chapter 1

Introduction

By a distributed system, we mean a collection of nodes that can communicate with one another. We assume that each node consists of a processor and some local memory. In order to optimize the overall performance of a distributed system, a need arises for effectively sharing the collective resources of the system, namely, the memory and the processors of the individual nodes. In this dissertation, we study two classes of problems, one arising in the context of sharing memory and the other arising in the context of sharing processors.

Sharing memory. A basic problem in a distributed system is to provide efficient access to shared objects (e.g., files, words of memory) that are stored in remote nodes. This is a complicated problem in general because of the large number of considerations involved, including: latency, bandwidth, faults, and network topology. Thus, for example, a request for a remote object may be delayed due to several reasons including: (i) the object may be a "hot spot", or (ii) the network may be congested, or (iii) the node holding the object may be temporarily faulty, or (iv) the node holding the object may be geographically distant. The first half of this dissertation focuses on developing protocols that efficiently handle faults and memory contention while providing access to shared objects. An overview of our results in this area is given in Section 1.1.

Sharing processors. Consider a distributed system where each node has a collection of tasks to execute. To minimize the total time taken to execute all of the tasks, one approach is to utilize the processors effectively by redistributing the tasks. For example, if there are some communication requirements among the tasks and communication is expensive, then it may be beneficial to distribute the tasks so that any two tasks that communicate with each other tend to be mapped to the same node or to nearby nodes. As another example, consider a scenario in which the tasks are independent and can be executed at any of the processors of the system. If the initial distribution of the tasks is uneven, then the total execution time can be reduced by transferring tasks from heavily loaded nodes to lightly loaded nodes. This technique, referred to as load balancing, is the focus of the second half of this dissertation. An overview of our results in this area is given in Section 1.2.

This dissertation demonstrates that a number of basic resource sharing problems admit efficient solutions in the form of *simple local* algorithms. Our algorithms are simple in the sense that the program running at each node can be expressed as a periodic process that repeatedly executes a small number of fixed operations. Our algorithms are local in the sense that each node either communicates with only its nearest neighbors or sends messages to only a small number of other nodes.

In addition to being easy to implement in practice, simple local algorithms are advantageous because they tend to be robust in the presence of faults and asynchrony. We provide some formal evidence in this regard by showing that most of our results hold in certain models that incorporate faults or asynchrony.

1.1 Sharing Memory

In the first part of the dissertation, we study two important issues that arise while providing access to shared data in a distributed system: *memory contention* and *faults*. Memory contention is the phenomenon in which an access request is delayed or rejected by a node because of other requests simultaneously sent to the node. This phenomenon may arise due to: (i) a number of nodes simultaneously attempting to access distinct objects residing at the same node (*exclusive access*), or (ii) a number of nodes simultaneously attempting to access the same object residing at some node (*concurrent access*). Even in the absence of memory contention, the response to an access request may be delayed if: (i) the remote node is faulty, or (ii) the requesting node is unable to communicate with the remote node due to faults in the communication network.

Our results in this part are presented in Chapters 2 and 3. Chapter 2 focuses on resolving memory contention that arises solely due to exclusive accesses and presents a protocol for supporting fast exclusive access to shared objects in a non-faulty distributed system. Chapter 3 considers the problem in its generality and presents a protocol for supporting fast fault-tolerant concurrent access to shared objects.

Chapters 2 and 3 are reviewed in Sections 1.1.2 and 1.1.3, respectively. We first describe in Section 1.1.1 the general aspects of the model of computation adopted in our study on sharing memory in a distributed system.

1.1.1 Model of Computation

We represent the distributed system as an *n-crossbar*. The n-crossbar (or simply, the crossbar) is a synchronous distributed machine that consists of n nodes and supports direct point-to-point communication between any two nodes. In order to address the issue of memory contention, we enhance the crossbar model as follows. A *c-arbitrary* crossbar is a crossbar in which: (i) the total number of messages sent or received by a single node in each step is at most c, and (ii) if the total number of messages destined to a node in a step exceeds c, then an adversary determines which subset of c messages is successfully received by the node.

We address faults by incorporating both static and dynamic node faults as well as a notion of faulty communication. Most importantly, our fault model assumes that at any given time: (i) a constant fraction of the nodes may be "down" (i.e., unable to communicate with any other nodes), and (ii) each "up" node may be unable to directly communicate (i.e., via a single point-to-point message) with a constant fraction of the other "up" nodes.

We now elaborate on three aspects of the model defined above: (i) synchrony, (ii) uniform topology, and (iii) the parameter c.

Synchrony. While our model allows messages to be lost due to communication failures and nodes to dynamically fail and recover, it disallows unpredictable delays in the delivery of messages and the execution of local computations. Our main reason for adopting a synchronous model is to obtain meaningful bounds on certain performance measures that concern time, e.g., the time taken to satisfy an access request. The task of devising an analytical model for deriving time bounds in an arbitrary asynchronous environment is a challenging open problem.

Uniform topology. In spite of the restrictive assumption of a uniform topology, the crossbar serves as a useful model for both tightly-coupled and loosely-coupled distributed systems. At one extreme, a system for which the interconnection is based on a dedicated switching network is precisely a crossbar. At the other extreme, the crossbar suitably models a wide-area network with a fast routing mechanism in studies where the particular topology of the network and the underlying routing mechanism are not of primary concern. In future work, we would like to extend our results to models that take into account the differing costs in communication among different nodes. (See [19, 74, 103] for recent work in this area.)

The parameter c. An important feature of the c-arbitrary crossbar is the parameter c which signifies the bandwidth limitation at each of the nodes of the network. This feature is motivated by the observation that with rapidly growing speeds and capacities of communication links, the network-node interface is increasingly limiting the performance of the network [41] [99, Chapter 9]. Restrictions on local bandwidth play a significant role in the BSP [115], LogP [41], and QSM(g) [58] models as well. The *gap* parameter of these models places an upper bound on the rate at which a single node may send or receive messages. However, BSP and LogP differ from the c-arbitrary crossbar on one significant point. In BSP and LogP, the gap places an upper bound on the number of messages *destined* to a single node in one step. On the other hand, the c-arbitrary crossbar model places no upper bound on the total number of messages destined to a single node in one step. Instead the model only limits to c the number of messages successfully *received* by a node in one step. The messages that are destined to a node and yet are not received by the node are assumed to be lost. Such a mechanism can be implemented, for example, by associating a time-out with each message so that every message that is not delivered at the end of a step times out.

Thus far, we have considered models that place restrictions only on the bandwidth available at the nodes. These models may not be appropriate for networks where the bandwidth of the network as a whole is the limiting factor. For such networks, we need to place global restrictions on the bandwidth, as is done in the PRAM(m) [90] and the QSM(m) [58] models. For a recent study on the implications of local and global bandwidth restrictions, see [3]. (We remark that our usage of the terms "local bandwidth" and "global bandwidth" is borrowed from [3].) The DRAM model [81] considers bandwidth restrictions in a more general form by explicitly accounting for congestion across every cut of the underlying network. Even though the DRAM provides a more accurate framework for evaluating performance, we have chosen a more abstract model for our study in order to simplify the analysis.

1.1.2 Exclusive Access to Shared Objects

We first consider the problem of resolving contention arising due to exclusive accesses in a fault-free c-arbitrary crossbar. Consider the scenario in which each node of the crossbar has a single access request. Since all the accesses are exclusive, each request is for a distinct object. Hence, we can represent the current set of access requests as a step of an EREW PRAM as follows: each processor of the PRAM maps to a distinct node of the crossbar, and each shared memory cell of the PRAM maps to a shared object of the crossbar. Our problem thus corresponds to the emulation of an n-processor EREW PRAM on a non-faulty c-arbitrary n-crossbar. In Chapter 2, we pinpoint the asymptotic complexity of a class of simple protocols that combine hashing and redundancy to obtain fast emulations of EREW PRAMs on c-arbitrary crossbars for all values for c. The results in this chapter are joint work with Phil Mackenzie and Greg Plaxton, and have appeared in [88].

A natural emulation scheme is to map each location of the EREW PRAM shared memory (and hence, each shared object) to the memory module of a randomly chosen node of the crossbar. We can easily see that the desired emulation corresponds to a random "balls and bins" experiment, in which each of n balls (representing the n memory accesses) are thrown independently and uniformly at random into n bins (representing the memory modules at the n nodes). It is straightforward to prove using Chernoff bounds [37] that if c is $O(1)$, then the preceding scheme requires $\Theta(\log n/\log\log n)$ time with high probability to emulate one step of the EREW PRAM. (By the phrase "with high probability", we mean "with probability $1 - 1/n^{\Omega(1)}$".)

In Chapter 2, we analyze a class of simple emulation protocols that replicate each object a constant number of times to achieve an $O(\log\log n)$ delay bound, and thus, an exponential improvement in delay. We remark that prior to our work, Karp, Luby, and Meyer auf der Heide [75], and subsequently, Dietzfelbinger and Meyer auf der Heide [47], presented protocols that achieve the $O(\log\log n)$ delay bound while incurring only a constant factor increase in space. Our results improve on previous results in two ways. First, our results hold for all values of c, while previous emulations required c to be sufficiently large. Second, we analyze a protocol that is more basic than those studied before. We are able to reduce our analysis to a study of a random balls and bins experiment that is similar to (but more complex than) the one mentioned above. We derive a sharp threshold phenomenon for the balls and bins experiment, which may be of independent interest.

1.1.3 Fault-Tolerant Concurrent Access to Shared Objects

The results reviewed in the preceding section concern a *static* set of exclusive accesses (i.e., with *no concurrency*) on a *fault-free* crossbar. In Chapter 3, we extend the ideas of hashing and replication that are used in Chapter 2, to address the more general setting of *dynamically* generated memory requests with *arbitrary concurrency* on a *faulty* distributed machine. Our study is largely motivated by the growing importance of wide-area network file systems which have become feasible due to high-speed networks. Since a completely general study is quite complicated, we make certain simplifying, yet realistic, assumptions that enable us to design and analyze a simple efficient protocol.

We represent the distributed system as a faulty $O(\log n)$-arbitrary crossbar, as defined in Section 1.1.1. We design and analyze a protocol for providing fast concurrent access to shared objects in this faulty network environment. Our protocol is based on on-line replication of objects and employs hashing and erasure codes. When a large number of clients attempt to read a given object at the same time, the object is rapidly replicated to an appropriate number of servers. Once the necessary level of replication has been achieved, each remaining request for the object is serviced within $O(1)$ expected steps.

We analyze our protocol and establish its optimality under two natural dynamic access patterns. In our first model, we assume that there is a fixed probability distribution from which each access request is independently drawn at all times. We show that our protocol reaches a steady state in $O(\log n)$ steps, in which each subsequent request is satisfied in $O(1)$ expected steps and $O(\log n)$ steps with high probability. The significance of this result is that it establishes the rapid adaptability of the protocol to arbitrary global access patterns. Since $\Omega(\log n)$ steps is a lower bound on the number of steps taken by any protocol to reach the steady state in the worst case, our analysis shows that the protocol reaches a steady state in an optimal number of steps. In the steady state, the expected time for satisfying any request is asymptotically optimal.

A drawback of the first model is that it does not allow, in general, dynamic changes in the "popularity" of an object (i.e., the number of users trying to access the object). In our second model, we assume that the popularity of an object can change arbitrarily, subject to the constraint that it is not more than a constant times the maximum popularity in the previous $\Theta(\log n)$ steps. For this model, we show that each request is satisfied in expected $O(1)$ steps and $O(\log n)$ steps with high probability. Moreover, for both the access pattern models, our protocol satisfies an asymptotically optimal number of requests per step while using an asymptotically optimal amount of communication.

The results in Chapter 3 are most suitable for applications in which: (i) reads occur much more frequently than writes, (ii) objects are not too small, and (iii) the popularity of any object does not change significantly over a short period of time. For example, the protocol might be appropriate for managing access to WWW pages on the Internet, since pages tend to be read far more often than they are written, the typical page size is thousands of bytes or more, and popular pages tend to remain popular over extended periods of time (e.g., for minutes, hours, or even days). In contrast, the protocol would probably be poorly-suited for use within a PRAM emulation scheme, where writes often account for a significant fraction of all accesses, the objects being accessed tend to be extremely small, and the popularity of an object may change arbitrarily from one time step to the next. The results in Chapter 3 are joint work with Greg Plaxton and have appeared in [102].

1.2 Sharing Processors

A natural approach towards harnessing the aggregate processing power of a distributed system is to balance the workload among the nodes of the system. In order to balance the workload, a simple strategy is to have each node periodically poll the other nodes to which it is connected and send some of its work to nodes with less pending work. The second part of this dissertation analyzes the effectiveness of this local load balancing strategy in the simplified scenario in which the load consists of a collection of independent unit-size jobs (called *tokens*) that can be executed on any other node.

The bulk of our results concerns *static* load balancing. In static load balancing, the total workload is available at the start of the computation, and no new load is added to the system. The main objective is to distribute the total load before initiating the computation such that the assignment of tasks to processors is balanced. Our main results in static load balancing are reviewed in Section 1.2.2.

Static load balancing is applicable in scenarios where information about the particular computation is known in advance. Parallel computations such as large-scale partial differential equations, finite element methods, branch-and-bound computations, and ray tracing, can be divided into a large number of small computational tasks and distributed among the processors at the start of the overall computation (see [77, 118] for some examples). Another important application of static load balancing arises in certain packet routing problems, where the initial distribution of packets may be irregular. In these problems, routing is performed by first redistributing the packets among the processors in a balanced manner, and then invoking standard routing techniques such as permutation routing or k-k-routing [97].

Our results for static load balancing also apply to certain problems where the load is *dynamic*, that is, the total load varies with time[1]. Dynamic load balancing is required in a wide variety of applications, including operating systems [48, 83], combinatorial optimization problems [80], adaptive mesh partitioning [68, 118], and fine-grain functional programming [59]. In certain problems arising in these applications, it is possible to divide the overall computation into phases, where the distributed system alternates between static load balancing and executing a portion of the workload.

Static load balancing may not be applicable in situations where the load is not known in advance. In such applications, it is required to have a continuous process that manages the distribution of load among nodes. Balancing dynamic load, however, is more difficult than balancing static load because of the potentially arbitrary nature of the on-line job arrival process. In order to make the study of dynamic load balancing somewhat tractable, most of the previous work has assumed either a particular statistical model of load variation or a specific network topology (for example, see [85, 111]). We adopt a different approach by proposing a model that allows an adversary to control the on-line job arrival process. Our results in this area are limited in scope. It is possible

[1]While we have used the term "static" or "dynamic" as a property of the load, some papers in the load balancing literature use the term as a property of the algorithm. These papers (for example, see [111]) define a static load balancing algorithm (resp., dynamic load balancing algorithm) to be an algorithm in which the decision of transferring load does not depend (resp., may depend) on the current system state.

that suitable enhancements of our model will be useful in the study of more realistic problems. Section 1.2.3 reviews our work on dynamic load balancing.

1.2.1 Model of Computation

We represent the distributed system as an arbitrary network G, where the vertices of G correspond to the nodes of the system and the edges of G correspond to the connections between nodes. We assume that in one unit of time, at most one token can be transmitted across an edge of the network in each direction. We focus primarily on algorithms with distributed control, that is, algorithms in which the action of a node depends only on the information available at the node and its neighbors.

The three defining characteristics of the model outlined above are: an arbitrary network topology, bounded edge capacity, and distributed control. We believe that these characteristics suitably model load balancing problems arising in practice. We also address the issue of asynchrony. The particular assumptions about asynchrony are discussed in the following section, where we review our results on static load balancing.

1.2.2 Static Load Balancing

We are given an arbitrary network G in which each node has an initial collection of tokens and no tokens are added or deleted while the tokens are being balanced. In each step, each node can transmit at most one token to each of its neighbors. The static load balancing problem is to design a distributed algorithm that reduces the *imbalance* of G, where the imbalance is defined to be the maximum difference between the number of tokens at any node and the average number of tokens per node. The performance of an algorithm is measured by the time it takes to balance the tokens, and by the final imbalance it achieves.

In Chapter 4, we tightly analyze a simple local algorithm of Aiello et al. [5], in which at each step each node sends a token to each of its neighbors with at least $2d$ fewer tokens (where d is the degree of the network). In particular, we prove that this algorithm, which we call the *multi-port algorithm*, balances to within $O(d^2 \log n/\alpha)$ tokens in $O(\Delta/\alpha)$ steps, where α is the edge expansion of the network and Δ is the imbalance associated with the initial token distribution. (We remark that there exist networks with diameter $\Omega(d^2 \log n/\alpha)$.) This upper bound is optimal in the following sense: Given any network with edge expansion α, there exists an initial load distribution with imbalance Δ for which any algorithm takes $\Omega(\Delta/\alpha)$ steps to balance to within even $\Delta/2$ tokens.

We extend our analysis to a slight variant of the multi-port algorithm for an asynchronous network model, in which edges are allowed to go "down" arbitrarily subject to the constraint that during any time unit the subgraph induced by the "up" edges has edge expansion α. We also obtain tight bounds on a randomized algorithm of Ghosh and Muthukrishnan [54] which balances load across edges chosen in a random matching.

Our main proof technique in Chapter 4 is a potential function argument based on the observation that, due to the expansion of the graph, there are many edges from "high" nodes (i.e., nodes having a large number of tokens) to "low" nodes. By assigning an exponentially higher potential to the "high" nodes, we show that a particular measure of the balance of the network improves rapidly throughout the course of the algorithm until the network is balanced to within $O((d^2 \log n)/\alpha)$ tokens. The results in Chapter 4 are joint work with Bhaskar Ghosh, Tom Leighton, Bruce Maggs, S. Muthukrishnan, Greg Plaxton, Andréa Richa, Robert Tarjan, and David Zuckerman, and have appeared in [52].

One shortcoming of our results in Chapter 4 is that the bounds on the final imbalance and the time taken to achieve the final imbalance are not optimal for *all* initial distributions. In Chapter 5, we establish a "universal" near-optimality result for the special class of ring networks. We show that for any initial token distribution b on a ring of n nodes, a particular local balancing algorithm converges to a completely balanced distribution within $4\mathrm{OPT}(b) + n$ steps, where $\mathrm{OPT}(b)$ is the time taken by an optimal centralized algorithm to balance b completely. More significantly, we generalize the preceding result to an asynchronous model in which local computations and messages may be arbitrarily delayed, subject to the constraint that each message is eventually delivered and each computation is eventually performed. The results in Chapter 5 are joint work with Johannes Gehrke and Greg Plaxton, and appear in [50].

1.2.3 Dynamic Load Balancing

In order to study the dynamic aspect of load balancing, we introduce the adversarial model in Chapter 6. In this model, tokens are created and/or destroyed in each step, and an adversary decides the number of these tokens and the location of each token. The goal of the balancing algorithm is to maintain a "small" imbalance at all times.

Clearly, some restrictions have to be placed on the adversary to disallow scenarios in which an unbounded amount of imbalance may be created, for example, by adding a large number of tokens at a single node. We place the following restriction: the adversary may insert/delete any number of tokens on any subset of nodes subject to the constraint that there exists an $\varepsilon > 0$ such that for every subset S of nodes, the change in the imbalance of S is at most $(1 - \varepsilon)$ times the number of edges coming out of S. In other words, the change in the imbalance of S is less than the number of edges connecting S to the rest of the network. It follows from an easy bisection argument that if the change in the imbalance of S is allowed to exceed the number of edges connecting S to the rest of the network, then an adversary may cause the imbalance of S to increase continuously with time.

Given the adversarial model, we seek *stable* algorithms. An algorithm is said to be stable if there exists Δ such that for all $t \geq 0$, the imbalance of G at the start of step t is at most Δ. In Chapter 6, we show that the local balancing algorithm of Aiello et al. [5] is stable for all networks for $\varepsilon > 0$. This result is joint work with S. Muthukrishnan.

1.3 Two Abbreviations for High Probability Bounds

Several claims in this dissertation are probabilistic and we differentiate between two kinds of high probability bounds. Let n denote the number of nodes in the system. Throughout this document, we use the abbreviation "whp" to mean "with high probability" or, more precisely, with probability $1 - n^{-c}$ for *some* constant c. Throughout this document, we use the abbreviation "wvhp" to mean "with very high probability" or, more precisely, with probability $1 - n^{-c}$, where c is a constant that can be set *arbitrarily large* by appropriately adjusting other constants defined within the relevant context.

Part I

Sharing Memory

Chapter 2

Fast Exclusive Access to Shared Objects

2.1 Introduction

In this chapter, we consider the problem of emulating an EREW PRAM on a c-arbitrary crossbar. We begin by reviewing the definitions of these two computational models. An *EREW PRAM* [49] is a collection of n processors along with a global shared memory. Input and output are provided in the shared memory. In a single computational step, each processor can read or write one memory location. The sole restriction is that no two processors are allowed to access the same memory location in a single step. (If two processors attempt to access the same memory location in a single step, then the machine halts.)

A c-arbitrary crossbar, which we introduced in Section 1.1.1, is a more realistic model of distributed computation in which the global shared memory is distributed over the n nodes of the machine. Input and output are provided in the distributed memory, and the nodes access the distributed memory via a communication network. Recall that the number of messages that a node of the c-arbitrary crossbar can send or receive in a single step is at most c. We now rephrase this property in terms of read/write operations to the distributed memory. We assume that each computational step of the crossbar consists of a read/write phase followed by an acknowledgment phase. During the read/write phase each node can issue one read or write request for a specific memory location. If the total number of read/write requests involving memory locations stored in any particular node A is less than or equal to c, then all requests involving A succeed and are acknowledged during the acknowledgment phase. On the other hand, if more than c processors attempt to access memory locations stored in the same node A, then an arbitrary subset of c of the requests succeed and are acknowledged.

We obtain our emulation results for the c-arbitrary crossbar by analyzing our emulations on a slight variant of the c-arbitrary crossbar, the *c-collision crossbar*. A c-collision crossbar is defined in the same manner as a c-arbitrary crossbar, except that if more than c requests are sent to a node A, then all requests involving A fail and no corresponding acknowledgments are sent. It is easy to see that given any emulation protocol for the c-collision crossbar, we can construct an equally efficient emulation protocol for the c-arbitrary crossbar.

The c-arbitrary and c-collision crossbars generalize some models that have been studied previously. The module parallel computer (MPC [9]) and the S*PRAM [116] correspond to the 1-arbitrary crossbar model. The local memory PRAM model of Anderson and Miller [10], later studied under the name OCPC (optical communication parallel computer) in [51, 60, 61], corresponds to the 1-collision crossbar model. The c-arbitrary and c-collision DMM models of [47, 75] are similar to the crossbar models in that the underlying communication network is a complete network. The DMM models, however, limit only the total number of *objects accessed* from a node to c and, hence, allow unbounded concurrent access to an object stored in a node, while the crossbar models limit the total number of *accesses satisfied* by a node to c.

As mentioned in Section 1.1.1, the c-arbitrary crossbar is a distributed memory model that addresses the issue of contention by placing a restriction on the bandwidth at each node. Among shared memory models, a well-studied model that addresses contention is the QRQW PRAM [57, 55, 56]. The QRQW PRAM extends the EREW

PRAM model by allowing simultaneous access to a cell in any step and allowing processor instructions to be pipelined. The amount of contention that occurs in a step is taken into account in the cost of performing the step.

2.1.1 Overview of the Results

Assume that we wish to emulate an EREW PRAM on a c-collision crossbar. If we map each EREW PRAM processor to a unique node of the crossbar, and employ a random hash function to map each location of the EREW PRAM shared memory to the memory module of some node of the crossbar, we can easily see the connection between the desired emulation and a random "balls and bins" experiment in which n balls are thrown independently and uniformly at random into n bins. Each of the at most n read or write requests generated in a single step of the EREW PRAM computation corresponds to a ball, and the memory module at each node corresponds to a bin.

If $c = O(1)$, we can conclude that the preceding scheme requires $\Omega(\lg n/\lg\lg n)$ time whp[1] to emulate one step of the EREW PRAM. On the other hand, Dietzfelbinger and Meyer auf der Heide [47] have recently shown that a bound of $O(\lg\lg n)$ time per EREW PRAM step is attainable for constant c. They presented a contention resolution protocol that minimizes the effect of the inevitable "hot-spot" memory modules (e.g., those memory modules receiving $\Theta(\lg n/\lg\lg n)$ requests under a given hash function) by storing three copies of each shared memory cell in the crossbar. Thus, at the expense of increasing the storage requirement by a factor of 3, the running time of the emulation is exponentially decreased.

In the protocol of [47], a read or write operation of memory location x by EREW PRAM processor i is emulated by having processor i of the c-collision crossbar access 2 out of the 3 copies corresponding to memory location x. The analysis presented in [47] requires some slack in the constants; in particular, they require $c \geq 3$, and are only able to analyze the protocol when it is used to emulate εn processors at a time, where ε is

[1] Recall that the term whp, which is defined in Section 1.3, means with probability $1 - n^{-\alpha}$ for some constant α.

a sufficiently small positive constant. (Thus, the overall running time of the protocol is increased by a factor of $1/\varepsilon$.)

The protocol of [47] is easily generalized to the "a out of b problem", in which b hash functions are used, and each node of the c-collision crossbar is required to access a out of b copies of a particular memory location. Our results in this chapter consist of the following bounds for the a out of b problem.

- In Section 2.4, we focus on the case $a = 1$, and pinpoint the asymptotic complexity of the resulting protocol on the c-collision crossbar for all possible choices of the parameter b and for c in $\{1, 2\}$. Furthermore, our analysis goes through with $\varepsilon = 1$, that is, we consider the most basic form of the protocol in which the action of all n EREW PRAM processors is emulated at once. For $c = 1$, we prove that the protocol runs in $\Theta(\lg \lg n)$ time whp if $b \geq 3$. For $c = 1$ and $b = 2$, we prove that the protocol runs in $\Omega(\lg n)$ time wvhp[1]. For $c = 2$ and $b \geq 2$, we prove that the protocol runs in $\Theta(\lg \lg n)$ time whp. Our results imply that for all constants $c \geq 2$, the 1 out of b protocol terminates in $\Theta(\log \log n)$ time whp on a c-collision crossbar. (The protocol will run faster for non-constant c. It would not be difficult to extend our analysis to obtain tight bounds for non-constant c.)

- In Section 2.5 we show that the above results hold even if the hash functions are only $O(\log^{\alpha} n)$-wise independent, where α is a sufficiently large real constant.

- In Section 2.6, we observe that any a out of b problem with $a > 1$ can be reduced to a sequence of 1 out of ℓ problems for an appropriate choice of ℓ. Thus, we are able to easily upper bound the complexity of a (new) protocol for essentially any a out of b problem. Our results yield, for example, $O(\log \log n)$ time protocols on 1-collision and 2-collision crossbars, and hence on 1-arbitrary and 2-arbitrary crossbars as well, using 5 and 3 hash functions, respectively. One might suspect that a reduction of this sort, while making the analysis easier, is only doing so at the expense of a significant

[1]Recall that the term wvhp, which is defined in Section 1.3, means with probability $1 - n^{-\alpha}$, where α is a constant that can be set arbitrarily large by appropriately adjusting other constants defined within the relevant context.

constant factor in performance. Interestingly, this is not the case; rather, as discussed in Section 2.6, our reduction yields a faster a out of b protocol than is obtained via the natural generalization of [47] for virtually all possible values of a and b.

2.1.2 Related Work

The ideas of hashing and replication have played a central role in almost all of the known shared memory simulations. Mehlhorn and Vishkin [91] proposed distributing the shared objects using universal hashing. Upfal and Wigderson [113] were the first to use replication in the context of shared memory simulations. They presented a deterministic protocol with delay $O(\log n(\log\log n)^2)$. One of their main observations, which has been used in several subsequent replication-based simulations, including ours, was that it is sufficient to access a majority of the copies. (The "majority trick" had been proposed earlier for concurrency control in replicated databases [112].)

Following the work of Upfal and Wigderson, several EREW PRAM emulations on the crossbar have been proposed. A deterministic simulation with delay $O(\log n)$ was presented in [9]; however, their result is non-constructive. Karp, Luby, and Meyer auf der Heide [75] presented a protocol that uses three hash functions and incurs $O(\log\log n)$ delay. Subsequently, Dietzfelbinger and Meyer auf der Heide [47] proved the same bound for a much simpler protocol which forms the basis for our work.

Subsequent to our work, faster protocols for EREW PRAM emulation on the closely related DMM model have been obtained. Czumaj, Meyer auf der Heide, and Stemann presented two randomized protocols for DMMs with $O(\log\log n/(\log\log\log n))$ [44] and $O(\log\log\log n \log^* n)$ [43] delay whp, respectively. Their protocols consist of certain preprocessing steps in which all the nodes of the DMM cooperate in estimating certain quantities associated with the PRAM step (e.g, the number of requests directed to each module). Hence, these protocols are more complicated than the ones we consider here. Moreover, the simulations of [44, 43] do not apply directly to the crossbar model which, as mentioned before, is not as powerful as the DMM model.

All of the protocols studied in this chapter and the ones discussed above are designed for distributed machines in which the underlying communication is by means of a complete network. Several researchers have studied simulations of shared memory on other topologies. Non-constructive simulations on networks of bounded degree were presented in [9]. Ranade devised a novel routing algorithm for the butterfly which leads to a CRCW PRAM simulation protocol with $O(\log n)$ delay [105]. Simulations of different PRAM models have been obtained on the mesh-connected computer [100] and its variants [82].

From the point of view of contention resolution, our work is related to the vast body of research on routing protocols for *multiple-access channels* (MACs) and optical computers. In a MAC, a set of distributed processes contend for a single shared resource, the channel, and the problem is to allocate the channel among the participating client processes. If two or more clients attempt to use the channel simultaneously, then there is a collision, and no client succeeds. The standard Ethernet local area network [92] and the classic ALOHA packet radio network [1] are two well-known examples of multiple access channels. (See [29, Chapter 4] for more examples.) An optical computer, which can be viewed as a collection of multiple access channels, is accurately modeled by a 1-collision crossbar.

There is a fundamental difference between the nature of contention arising in the context of routing for MACs and optical computers, on the one hand, and in shared memory simulations, on the other. While the "minimum unit of contention" in the former case is the channel or the memory module, in the latter case it is an object which is *replicable*. To illustrate this, consider a routing problem on a 1-collision crossbar in which h nodes send one message each to the same destination. It is easy to see that there is a lower bound of h for this problem. However, if we consider the analogous scenario of h nodes contending for objects that reside in the same memory module, a bound of h can be overcome by replication.

In the absence of replication, a natural approach to resolve contention for MACs is to use randomization to break the symmetry among the clients. This idea is central to many MAC protocols, including the Ethernet protocol [92] and the slotted ALOHA protocol [1], and routing protocols for the optical computer [10, 116, 51]. (For more work in this direction, see [64, 65].) Lower bounds for routing in optical computers appear in [62, 88].

2.2 The 1 out of ℓ Protocol

Consider the 1 out of ℓ problem where $\ell \geq 1$. Let the ℓ hash functions be labeled h_i, $0 \leq i < \ell$, and the shared memory request of node j be for cell x_j. Node j needs to successfully access one of the memory locations $h_i(x_j)$, $0 \leq i < \ell$. To solve the 1 out of ℓ problem, the following sequence of ℓ rounds can be repeated until each processor has had one successful access:

- In the ith round, where $0 \leq i < \ell$, if node j has not successfully accessed any copy of x_j, then node j attempts to access $h_i(x_j)$.

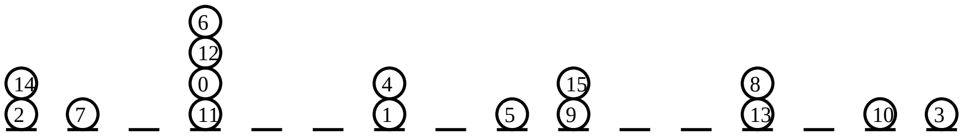
Each round is executed in a synchronous fashion. We refer to this protocol as the 1 out of ℓ protocol. (This is analogous to Access Schedule 2 of [47], defined for the 2 out of 3 problem.)

2.3 Sketch of the Analysis

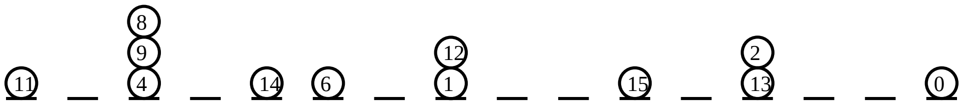
In order to understand the execution of the 1 out of ℓ protocol, let us consider the underlying process in an equivalent balls-and-bins setup. Assume for simplicity that the protocol is running on a 1-collision crossbar. Let T_i (resp., t_i) denote the set (resp., number) of outstanding requests at the end of round i.

2.3.1 Unbounded ℓ

To gain an initial intuition, let us assume that we have an infinite number of hash functions (i.e., ℓ is unbounded). Consider round 0 of the protocol. Since h_0 is chosen independently and uniformly at random, round 0 corresponds to a random throw of n balls into n bins. In a 1-collision crossbar, if a node receives more than one request, then the node rejects all of its requests. Therefore, t_0 is the number of *non-singletons*, where a non-singleton is a ball that lands into a bin with more than one ball. Following the same argument, we obtain that for all i, t_i is the number of non-singletons obtained in a random throw of t_{i-1} balls into n bins. Figure 2.1 illustrates rounds 0 and 1 of an instance of the random process that begins with 16 balls and 16 bins.



(a)



(b)

Figure 2.1: Rounds 0 and 1 of the 1 out of ℓ protocol, with $\ell \geq 2$ and $n = 16$. The 16 balls are numbered 0 through 15. Round 0 (part (a)) is an instance of a random throw of 16 balls into 16 bins. Since balls 3, 5, 7, and 10 are singletons we remove them before round 1. Round 1 (part (b)) is an instance of a random throw of the remaining 12 balls into 16 bins.

The process of throwing k balls into n bins is well understood and it is straightforward to derive expected and high probability bounds on the number of non-singletons. As an example, consider the case when k is n. The probability that a bin receives exactly one ball is $(1 - 1/n)^{n-1} \approx 1/e$. Thus, the expected number of bins that receive exactly one ball is approximately n/e. It follows that the expected number of non-singletons is approximately $(e-1)n/e$. More generally, we can show that if n/a balls are thrown independently and uniformly at random into n bins, then the number of non-singletons is $O(n/a^2)$ wvhp. (The intuition behind the preceding observation is the following: the probability that a ball lands in a bin holding at least one other ball is most $1/a$.) Using the above calculations, it is easy to show that there exist constants $\alpha > 1$ and $\beta > 1$ such that t_i is $O(n/\alpha^{\beta^i})$ wvhp for all i. Thus, we obtain that for $i = \Theta(\log \log n)$, t_i is zero wvhp, and hence all requests are satisfied within $\Theta(\log \log n)$ rounds wvhp.

2.3.2 Bounded ℓ

Unfortunately, the approach outlined in the preceding section does not carry through for the case of constant ℓ because the sequence of distributions arising in the analysis quickly deviates from the simple behavior associated with the process of throwing k balls randomly into n bins in round i. To see this, consider round ℓ of the protocol. In this round, each node j that has not successfully accessed a copy of x_j in any of the first ℓ rounds attempts to access $h_0(x_j)$. (Recall that the protocol cycles through the ℓ hash functions.) Even though h_0 is a hash function that is chosen independently and uniformly at random, it is not the case that round ℓ corresponds to a random throw of $t_{\ell-1}$ balls into n bins. This is because there is a correlation between the set $T_{\ell-1}$ of requests that are outstanding after round $\ell - 1$ and the set of hash functions used in the first ℓ rounds, which includes h_0. Figure 2.2 illustrates round 2 of the experiment initiated in Figure 2.1 under the assumption that ℓ is 2. The remaining rounds are shown in Figure 2.3.

Figure 2.2: Round 2 of the 1 out of 2 protocol for the example in Figure 2.1. We obtain the distribution in round 2 by removing the four singletons obtained in round 1 and the five singletons 0, 6, 11, 14, and 15 obtained in round 1 (Figure 2.1(b)) from the distribution obtained in round 0 (Figure 2.1(a)).

The key idea underlying our analysis is that we are able to characterize the process associated with round ℓ (and each of the subsequent rounds) in terms of certain "truncated k balls in n bins" distributions that are obtained by throwing k balls into n bins and removing balls until a certain condition holds. We then show that the truncated distributions can be approximated by suitably chosen instances of the standard balls-and-bins distributions.

Our approach can be illustrated by considering the relationship between rounds 0

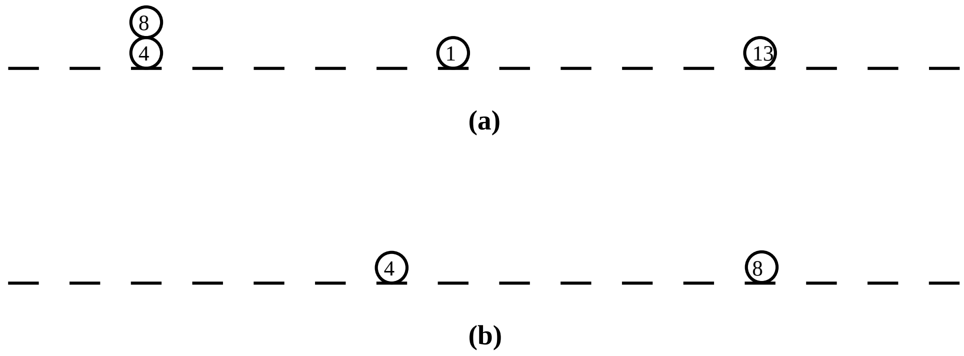


Figure 2.3: Rounds 3 and 4 of the 1 out of 2 protocol, for the example in Figure 2.1. We obtain the distribution in round 3 (part (a)) by removing the five singletons obtained in round 1 and the three singletons 2, 9, and 12 obtained in round 2 (Figure 2.2) from the distribution obtained in round 1 (Figure 2.1(b)). We obtain the distribution in round 4 (part (b)) by removing the three singletons obtained in round 3 and the two singletons 1 and 13 obtained in round 3 (Figure 2.2(b)) from the distribution obtained in round 2 (Figure 2.1(b)). Since there are no balls left after round 4, the protocol terminates.

and ℓ. As mentioned before, T_0 represents the set of non-singletons obtained in a random throw of a set (say S_0) of n balls into n bins. Since $T_{\ell-1}$ is a subset of T_0, we obtain that for any request in $T_{\ell-1}$, the location of the copy determined by h_0 is no longer random. The following crucial property, however, still holds: *the set $T_{\ell-1}$ is a random subset of* T_0. This property indicates that T_ℓ can be determined by the following experiment:

- Remove balls from S_0 at random until the number of balls remaining from T_0 is $t_{\ell-1}$. (Let S_ℓ denote the set of balls remaining from S_0.)

The set T_ℓ is precisely the set of non-singletons obtained at the end of the preceding experiment. Moreover, the set S_ℓ is a nearly random $|S_\ell|$-size subset of S_0. (We say "nearly random" because S_ℓ is biased by the stopping condition of the experiment. For example, the number of balls in S_ℓ that also belong to T_0 is $t_{\ell-1}$.) An experiment in which we first throw k balls into n bins at random and then remove a random subset of k' balls is equivalent to a random throw of $k - k'$ balls into n bins. Hence, t_ℓ is closely approximated by the number of non-singletons obtained in a random throw of $s_\ell = |S_\ell|$ balls into n bins. We have thus reduced the calculation of t_ℓ to that of s_ℓ. It is easy to see that the expected value of s_ℓ is approximately $nt_{\ell-1}/t_0$. It follows from a

Chernoff-type argument that s_ℓ is $\Theta(nt_{\ell-1}/t_0)$ wvhp. More generally, for all $i \geq \ell$, we can show that there exists $s_i = \Theta(s_{i-\ell}t_{i-1}/t_{i-\ell})$ such that t_i approximates the number of non-singletons obtained in a random throw of s_i balls into n bins.

We now have all the ingredients that together complete the analysis of the 1 out of ℓ protocol. The first ingredient is a sharp estimate of the number of non-singletons in a random throw of k balls into n bins. The second ingredient is the proof that t_i approximates the number of non-singletons obtained in a random throw of s_i balls into n bins. The third ingredient is that s_i can be determined from $s_{i-\ell}$, $t_{i-\ell}$, and t_{i-1}.

Using the first and second ingredients, we obtain a relation between s_i and t_i. Then, using the third ingredient, we obtain a recurrence relation among the t_i's. Finally, the recurrence relation yields a bound on the number of rounds it takes for t_i to be zero, and hence a bound on the running time of the protocol.

The formal analysis is somewhat more complicated than the informal description outlined above. For example, it turns out that both our estimate of the number of non-singletons as well as our bounds relating the quantities s_i, $s_{i-\ell}$, $t_{i-\ell}$, and t_{i-1}, need to be accurate to within a $1 - o(1)$ factor. Fortunately, the theory of martingales provides a mechanism to obtain bounds that have the required level of accuracy. Another complication arising in the analysis is that, as indicated above, t_i is only approximately given by the number of non-singletons in a random throw of s_i balls into n bins. We make the notion of approximation precise by deriving suitably tight upper and lower bounds for t_i.

2.3.3 Summary

The basic ideas sketched in Section 2.3.2 can be generalized to derive a bound on the running time of the protocol for all values of ℓ and c. For example, when c is 1, we obtain that the running time is $\Theta(\log \log n)$ for all $\ell \geq 3$.

Finally, as mentioned in Section 2.1.1, we derive a protocol for the a out of b problem by running a sequence of an appropriate number of 1 out of ℓ protocols. The analysis for the a out of b problem follows from the analysis for the 1 out of ℓ problem in a straightforward manner.

2.4 Analysis of the 1 out of ℓ Protocol

In this section we analyze the 1 out of ℓ protocol under the assumption that each hash function is chosen independently and uniformly at random. We begin our analysis by presenting the 1 out of ℓ protocol in an equivalent balls-and-bins setup. Let n balls labeled 0 through $n-1$ represent the accesses, and n bins labeled 0 through $n-1$ represent the memory modules. Each hash function, a random function from $[n]$ to $[n]$, is equivalent to a random throw of n balls uniformly and independently into n bins. Let h^A denote the function h with domain restricted to the set $A \subseteq [n]$. Let R_i denote the set of balls remaining after round i. For convenience, define R_{-1} to be the set of balls left before round 0, i.e., $R_{-1} = [n]$. Note that for $i \geq 0$, R_i is the subset of R_{i-1} given by the following recurrence:

$$R_i = \{j \in R_{i-1} : |f^{-1}(f(j))| > c\}, \text{ where } f = h_{i \bmod \ell}^{R_{i-1}}.$$

Recall that a bag (or multiset) is an unordered set in which repetition is allowed. For any set A we define a bag B to be an A-*bag* if every element of B is also an element of A.

For nonnegative integers m and n, let $\mathcal{F}_{m,n}$ denote the set of functions from $[m]$ to $[n]$. For each $f \in \mathcal{F}_{m,n}$, note that the bag $\{f(j) : j \in [m]\}$ is an m-size $[n]$-bag. For convenience, given any $f \in \mathcal{F}_{m,n}$ and $A \subseteq [m]$, let the term *bag* $f(A)$ denote the bag $\{f(x) : x \in A\}$. The uniform distribution over $\mathcal{F}_{m,n}$ induces a probability distribution, which we denote $\mathcal{D}_{m,n}$, over the set of all m-size $[n]$-bags. For any bag B and $A \subseteq [n]$, let $\mathcal{B}_{A,B} = \{f^A : f \in \mathcal{F}_{n,n} \text{ and bag } f(A) = B\}$.

Let S_i and T_i denote the bags $h_{i \bmod \ell}(R_{i-1})$ and $h_{i \bmod \ell}(R_i)$, respectively. Let $t_i = |T_i| = |R_i|$ (thus $t_{-1} = n$) and $s_i = |S_i|$. Note that $\langle t_i \rangle$ is a nonincreasing sequence. The protocol terminates after the first round i for which $t_i = 0$. The protocol fails to terminate if and only if $t_i = t_{i+\ell} > 0$ for some $i \geq -1$. (In such a case, the protocol enters an infinite loop with $t_j = t_i$ for all $j \geq i$.) The goal of our analysis is twofold: (i) to bound the probability that the protocol fails to terminate, and (ii) to analyze the number of rounds required by the protocol when it does terminate. We begin our analysis by establishing some properties of $\mathcal{D}_{m,n}$ and $\mathcal{B}_{A,B}$.

Let random variable X be drawn from $\mathcal{D}_{m,n}$, B be an arbitrary $[n]$-bag of size m, and m_i be the number of copies of element i in B, $0 \leq i < n$. We have:

$$\Pr[X = B] = \frac{m!}{m_0! \cdots m_{n-1}!} \cdot \frac{1}{n^m}. \tag{2.1}$$

Lemma 2.1: *Let m and n be integers such that $0 \leq m < n$ and assume that X is a random variable drawn from $\mathcal{D}_{m+1,n}$. Let x be chosen uniformly at random from X. If Y is the random variable $X \setminus \{x\}$, then the probability distribution of Y is $\mathcal{D}_{m,n}$.*

Proof: Let B be any m-size $[n]$-bag and $B_i = B \cup \{i\}$, $0 \leq i < n$. Let the number of copies of element i in B be m_i. (Hence $\sum_{i=0}^{n-1} m_i = m$.) Using Equation 2.1 we have

$$\begin{aligned}
\Pr[Y = B] &= \sum_{i=0}^{n-1} \Pr[X = B_i] \cdot \frac{m_i + 1}{m+1} \\
&= \sum_{i=0}^{n-1} \left(\frac{1}{n^{m+1}} \cdot \frac{(m+1)!}{(m_i+1)!} \prod_{j \neq i} \frac{1}{m_j!} \right) \frac{m_i + 1}{m+1} \\
&= \sum_{i=0}^{n-1} \frac{1}{n^{m+1}} \cdot \frac{m!}{m_0! \cdots m_{n-1}!} \\
&= \frac{m!}{m_0! \cdots m_{n-1}!} \cdot \frac{1}{n^m}.
\end{aligned}$$

□

Corollary 2.1.1: *Let a, m, and n be integers such that $0 \leq a \leq m \leq n$. Let X be a random variable drawn from $\mathcal{D}_{m,n}$. If Y is a random a-size subbag of X, then the probability distribution of Y is $\mathcal{D}_{a,n}$.* □

Lemma 2.2: *Let R be an arbitrary subset of $[n]$ and B be an arbitrary $[n]$-bag. Let h be a function drawn uniformly at random from $\mathcal{B}_{R,B}$. For an arbitrary subset A of R, the bag $h(A)$ is a random $|A|$-size subbag of B.*

Proof: Consider an arbitrary element $x \in R$. Clearly $h(x)$ is a random element of B. Applying this for each element in A, we obtain that the bag $h(A)$ is a random $|A|$-size subbag of B. ∎

For any random variable X and any event E which occurs with non-zero probability, let $X \mid E$ denote the random variable whose probability distribution is the conditional probability distribution of X given E. Using Corollary 2.1.1 and Lemma 2.2, we prove the following claims about the 1 out of ℓ protocol.

Lemma 2.3: *Let R be an arbitrary subset of $[n]$ and T be an arbitrary $[n]$-bag. For all $i \geq 0$, if $\Pr[\{R_i = R, T_i = T\}]$ is non-zero, then the random variable $h^R_{i \bmod \ell} \mid \{R_i = R, T_i = T\}$ is drawn uniformly at random from $\mathcal{B}_{R,T}$.*

Proof: The random variable $h_{i \bmod \ell}$ is drawn uniformly at random from $\mathcal{F}_{n,n}$. Therefore, given that $R_i = R$ and bag $h_{i \bmod \ell}(R) = T_i = T$, $h^R_{i \bmod \ell}$ is drawn uniformly at random from the set of functions whose domain is R and the range, viewed as a bag, is the $[n]$-bag T. This set of functions is precisely $\mathcal{B}_{R,T}$. ∎

Lemma 2.4: *For $0 \leq j < i$, let $\widehat{R}_j$, $\widehat{T}_j$, and $\widehat{t}_j$ be an arbitrary subset of $[n]$, an arbitrary $[n]$-bag, and an arbitrary integer, respectively, such that $\widehat{t}_j = |\widehat{R}_j| = |\widehat{T}_j|$. Let S'_i denote the random variable $S_i \mid \{R_j = \widehat{R}_j, T_j = \widehat{T}_j, t_j = \widehat{t}_j : 0 \leq j < i\}$. Let i be a nonnegative integer and $\Pr[\{R_j = \widehat{R}_j, T_j = \widehat{T}_j, t_j = \widehat{t}_j : 0 \leq j < i\}]$ be non-zero. If $0 \leq i < \ell$, then S'_i is drawn from $\mathcal{D}_{\widehat{t}_{i-1},n}$; otherwise, S'_i is a $\widehat{t}_{i-1}$-size random subbag of $\widehat{T}_{i-\ell}$.*

Proof: By definition, S_i equals the bag $h_{i \bmod \ell}(R_{i-1})$. If $0 \leq i < \ell$, then S'_i equals the bag $h_{i \bmod \ell}(\widehat{R}_{i-1})$ because $h_{i \bmod \ell}$ is independent of all events associated with rounds 0 through $i-1$. Let C be an arbitrary n-bag and let h' denote the random variable $h_{i \bmod \ell} \mid \{h_{i \bmod \ell}([n]) = C\}$. The probability distribution of bag $h_{i \bmod \ell}([n])$ is $\mathcal{D}_{n,n}$ and hence h' is drawn uniformly at random from $\mathcal{B}_{[n],C}$. Applying Lemma 2.2 with $([n], C, h', \widehat{R}_{i-1})$ for (R, B, h, A), we obtain that bag $h'(\widehat{R}_{i-1})$, i.e., $S'_i \mid \{h_{i \bmod \ell}([n]) = C\}$, is a random $\widehat{t}_{i-1}$-size subbag of C. Since the preceding statement holds for all C, we apply Corollary 2.1.1 with $(\widehat{t}_{i-1}, n, n, h_{i \bmod \ell}([n]), S'_i)$ for (a, m, n, X, Y) to obtain that S'_i is drawn from $\mathcal{D}_{\widehat{t}_{i-1},n}$.

We now consider the case $i \geq \ell$. Let h'' denote the random variable $h_{i \bmod \ell}^{\widehat{R}_{i-\ell}} \mid \{R_{i-\ell} = \widehat{R}_{i-\ell}, T_{i-\ell} = \widehat{T}_{i-\ell}\}$. Since h'' is independent of all events associated with rounds $i-\ell+1$ through $i-1$, S'_i equals the bag $h''(\widehat{R}_{i-1})$. Applying Lemma 2.3 with $(\widehat{R}_{i-\ell}, \widehat{T}_{i-\ell}, i-\ell)$ for (R, T, i), we obtain that h'' is drawn uniformly at random from $\mathcal{B}_{\widehat{R}_{i-\ell}, \widehat{T}_{i-\ell}}$. Applying Lemma 2.2 with $(\widehat{R}_{i-\ell}, \widehat{T}_{i-\ell}, h'', \widehat{R}_{i-1})$ for (R, B, h, A), we obtain that bag $h''(\widehat{R}_{i-1})$, i.e., S'_i, is a random $\widehat{t}_{i-1}$-size subbag of $\widehat{T}_{i-\ell}$. $\square$

We are now ready to describe the protocol in terms of the S_i's and T_i's alone. Let *RandomBag*(m, n) return a bag drawn from $\mathcal{D}_{m,n}$. Let *RandomSubbag*(B, m) return a new bag that is a random m-size subbag of B. Let *PrunedBag*(B, c) return a bag that contains exactly those elements of S that have more than c copies. **Alg1**(n, ℓ, c) algorithmically describes the random process established by Lemma 2.4 regarding the 1-out-of-ℓ protocol on a c-collision n-crossbar.

Alg1(n, ℓ, c)

$i := 0$;
repeat
  if $i < \ell$ **then**
    $S_i :=$ *RandomBag*$(|T_{i-1}|, n)$
  else
    $S_i :=$ *RandomSubbag*$(T_{i-\ell}, |T_{i-1}|)$;
  $T_i :=$ *PrunedBag*(S_i, c);
  $i := i + 1$
until $|T_{i-1}| = 0$

In order to analyze **Alg1** we will estimate the size of T_i after round i. We propose a modified version of the above algorithm that simplifies the estimation of $|T_i|$. Observe that for $0 \leq i < \ell$, S_i is the bag obtained by throwing $|S_i|$ balls at random into n bins, and T_i is *PrunedBag*(S_i, c). Below we present the modified algorithm **Alg2**(n, ℓ, c) that approximately maintains this invariant after *every* round, under a suitable redefinition of S_i. The analysis in Section 2.4.3 will make this precise. **Alg2** is the same as **Alg1** except that Lines (1.5) and (1.6) are replaced by Lines (2.1) through (2.7), stated below.

(2.1) **else** {

(2.2) $\quad S_i, T_i := S_{i-\ell}, T_{i-\ell}$;

(2.3) $\quad$ **while** $|T_i| > |T_{i-1}|$ {

(2.4) $\qquad$ "Select x at random from S_i";

(2.5) $\qquad S_i, T_i := S_i \setminus \{x\}, T_i \setminus \{x\}$

(2.6) $\quad$ }

(2.7) };

Since each element x in line (2.4) is selected at random from S_i, any element selected from T_i is also random in T_i. Moreover exactly $|T_{i-1}|$ of the elements from $T_{i-\ell}$ are retained after the execution of the **while** loop.

Lemma 2.5: *Let S_i^1, T_i^1 (resp., S_i^2, T_i^2) denote bags S_i, T_i in* **Alg1** *(resp.,* **Alg2***) after round i, $i \geq 0$. Then T_i^1 and T_i^2 have the same probability distribution.*

Proof: We use induction on the number of rounds. For the basis, we observe that $T_0, \ldots, T_{\ell-1}$ in **Alg1** and **Alg2** are obtained in exactly the same way. (Lines (1.5) and (1.6) of **Alg1** and the corresponding lines (2.1) through (2.7) of **Alg2** are not executed.)

Consider round $i \geq \ell$. By the induction hypothesis T_j^1 and T_j^2 have the same probability distribution, $0 \leq j < i$. In Line (1.6), **Alg1** computes S_i^1 by selecting a random subbag of size $|T_{i-1}^1|$ from the subbag $T_{i-\ell}^1$. In Lines (2.3) through (2.6), **Alg2** computes S_i^2 by removing at random elements from $S_{i-\ell}^2$ until $|T_{i-1}^2|$ elements are retained from subbag $T_{i-\ell}^2$. Thus T_i^2 is a $|T_{i-1}^2|$-size subbag chosen randomly from $T_{i-\ell}^2$. By the induction hypothesis, the probability distribution of $T_{i-\ell}^1$ (resp., T_{i-1}^1) is the same as that of $T_{i-\ell}^2$ (resp., T_{i-1}^2). Therefore, S_i^1 after Line (1.6) of **Alg1** and T_i^2 after Line (2.6) of **Alg2** have the same probability distribution. Let S' (resp., T') denote S_i^2

(resp., T_i^2) after Line (2.6) of **Alg2**. Since $T_{i-\ell}^2$ contains all elements of $S_{i-\ell}^2$ with more than c copies, T' contains all elements of S' with more than c copies.

After Line (1.7), T_i^1 is the subbag of S_i^1 containing all elements with more than c copies, and T_i^2 is the subbag of S' containing all elements with more than c copies. Since T' contains all elements of S' with more than c copies, T_i^2 is the subbag of T' containing all elements with more than c copies. Therefore, the probability distribution of T_i^1 after round i is the same as that of T_i^2 after round i. $\square$

Corollary 2.5.1: *The probability that* $\mathbf{Alg1}(n, \ell, c)$ *terminates after round* i, $i \geq 0$, *is equal to the probability that* $\mathbf{Alg2}(n, \ell, c)$ *terminates after round* i. $\square$

In the remainder of this section, we analyze **Alg2** for the cases where c is in $\{1, 2\}$ and ℓ is arbitrary. The analysis can be easily generalized to apply for all values of c. We begin by presenting, in Section 2.4.1, some results on large deviations of certain probability distributions. In Section 2.4.2, we analyze certain "balls and bins" experiments. Section 2.4.3 uses these analyses to obtain tight bounds on the running times of $\mathbf{Alg2}(n, \ell, 1)$ and $\mathbf{Alg2}(n, \ell, 2)$. Among other results, we show that $\mathbf{Alg2}(n, 3, 1)$ and $\mathbf{Alg2}(n, 2, 2)$ both terminate in $\Theta(\log \log n)$ rounds whp.

2.4.1 Large Deviations

For our analysis, we make frequent use of bounds on the tails of the binomial and hypergeometric distributions [8, 37, 38, 70]. These bounds are stated in Appendix A. Lemmas 2.6 and 2.7 are obtained from bounds on the tails of the hypergeometric and binomial distributions, respectively.

Lemma 2.6: *Let* S *be a set of* s *balls, and* T *be a subset of* S, $t = |T|$. *Let* s' *balls be chosen uniformly at random from* S, *and* t' *be the random variable representing the number of balls that are chosen from* T. *Then,*

$$\begin{aligned} \Pr[t' \geq (1 + 1/(2\log^3 n))s't/s] &\leq e^{-s't^2/(2s^2 \log^6 n)}, \text{ and} \\ \Pr[t' \leq (1 - 1/(2\log^3 n))s't/s] &\leq e^{-s't^2/(2s^2 \log^6 n)}. \end{aligned}$$

Proof: Apply Theorem A.2 with $\varepsilon = t/(2s \log^3 n)$. $\square$

Lemma 2.7: *Let S be a set of s balls and T be a subset of S, $t = |T|$. Let s' balls be chosen at random from S, and let t' be the random variable representing the number of balls that are chosen from T. If $s't/s \geq \log^2 n$, then $t' \geq s't/(3s)$ wvhp.*

Proof: Let $p = t/s$. Consider the s' balls being chosen in s' rounds (one ball in each round). If the number of balls chosen from bag T in rounds $1, \ldots, i-1$ is less than $ps'/3$, the probability that a ball from T is chosen in round i is at least $2p/3$. Let X be a random variable drawn from $B(s', 2p/3)$. The probability that $t' \geq ps'/3$ is at least the probability that $X \geq ps'/3$. By Equation (A.1) of Theorem A.1, $\Pr[X \geq ps'/3] \geq 1 - e^{-ps'/12}$. Since $ps' \geq \log^2 n$, the lemma is proven. $\square$

In $\mathbf{Alg2}(n, \ell, 1)$, T_i is that subbag of S_i, each element of which has at least 2 copies. We call such elements (as well as the associated balls) *non-singletons*. Similarly, in $\mathbf{Alg2}(n, \ell, 2)$ each element of T_i has at least 3 copies. We call such elements (as well as the associated balls) *non-pairs*. In Section 2.4.3, we show that the probability distribution of S_i is approximately $\mathcal{D}_{s_i,n}$. Thus, in $\mathbf{Alg2}(n, \ell, 1)$ (resp., $\mathbf{Alg2}(n, \ell, 2)$) t_i is approximately the number of non-singletons (resp., non-pairs) in a random bag drawn from $\mathcal{D}_{s_i,n}$. In order to get sharp estimates on the number of non-singletons and non-pairs in a random bag drawn from $\mathcal{D}_{m,n}$, we use a martingale analysis. Appendix B defines a martingale and states certain useful bounds on large deviations for martingales.

2.4.2 Lemmas on Balls and Bins

In this section, we estimate the number of non-singletons and non-pairs in a random bag with distribution $\mathcal{D}_{m,n}$ using the large deviation bounds mentioned in Section 2.4.1. By linearity of expectation, the expected number of non-singletons (resp., non-pairs) of a random bag X drawn from $\mathcal{D}_{m,n}$ is given by $f(m,n)$ (resp., $g(m,n)$), where

$$f(m,n) = m\left(1 - \left(1 - \frac{1}{n}\right)^{m-1}\right), \text{ and}$$

$$g(m,n) \quad = \quad m\left(1-\left(1-\frac{1}{n}\right)^{m-1}-\frac{m-1}{n}\left(1-\frac{1}{n}\right)^{m-2}\right).$$

Throughout this section n will be fixed, so we use $f(m)$ (resp., $g(m)$) to denote $f(m,n)$ (resp., $g(m,n)$). In Section C.1 of Appendix C, we derive certain properties of f and g. In particular, Lemmas C.1 and C.2 show that $f(m) = \Theta(m^2/n)$, and $g(m) = \Theta(m^3/n^2)$. Let

$$\begin{aligned} \delta &= 1 - 1/\log^3 n, \text{ and} \\ \Delta &= 1 + 1/\log^3 n. \end{aligned}$$

We now bound the probability that the number of non-singletons in a random bag drawn from $D_{m,n}$ deviates from the mean $f(m)$. Lemma 2.8 is used to bound deviations to within a $o(1)$ factor for m suitably large, and Lemma 2.9 bounds deviations to within a constant factor for all m.

Lemma 2.8: *Let m and n be integers such that $3 \le m \le n$, and $h : [m] \to [n]$ be a random function drawn from $\mathcal{F}_{m,n}$, and $t(h)$ be the number of non-singletons in bag $h([m])$. If $m \ge n^{2/3}\log^3 n$, then $\delta f(m) \le t(h) \le \Delta f(m)$ wvhp.*

Proof: Consider the martingale $X_0, \ldots, X_m$ defined as:

$$X_i(h) = E[t(p) \mid p \text{ and } h \text{ agree on balls in } [i]].$$

If two functions p and p' differ only on ball i, $t(p)$ and $t(p')$ differ by at most 2. We apply Theorem B.2 by scaling the random variable t by 2 and thus obtain, $|X_{i+1} - X_i| \le 2$. Similarly, after scaling X_i's by 2, we apply Theorem B.1 to get

$$\Pr[|X_m - X_0| > 2\lambda\sqrt{m}] < 2e^{-\lambda^2/2}. \tag{2.2}$$

The expected value X_0 of t, is $f(m)$. For a function h, $t(h)$ is $X_m(h)$. By Equation 2.2 with $\lambda = f(m)/(2\sqrt{m}\log^3 n)$, we find that

$$\Pr\left[|t(p) - f(m)| > \frac{f(m)}{\log^3 n}\right] < 2e^{-f(m)^2/(8m\log^6 n)}.$$

Since $f(m) \geq m^2/3n$, for all $m > 2$

$$\Pr\left[|t(p) - f(m)| > \frac{f(m)}{\log^3 n}\right] < 2e^{-m^3/(72n^2 \log^6 n)}.$$

For $m \geq n^{2/3} \log^3 n$, $m^3/(72^2 \log^6 n) \geq (\log^3 n)/72$. Therefore, $\delta f(m) \leq t(p) \leq \Delta f(m)$ wvhp. $\square$

Corollary 2.8.1: *Let m and n be integers such that $3 \leq m \leq n$, S be a random bag drawn from $\mathcal{D}_{m,n}$, and t be the number of non-singletons in S. If $m \geq n^{2/3} \log^3 n$, then $\delta f(m) \leq t \leq \Delta f(m)$ wvhp.* $\square$

Lemma 2.9: *Let m and n be integers such that $3 \leq m \leq n$ and S be a random bag drawn from $\mathcal{D}_{m,n}$. Let t represent the number of non-singletons in S. Then,*

1. *The probability that a particular ball is a non-singleton is at most m/n.*

2. *For $\sqrt{n} \log^5 n \leq m \leq n$, we have $t \leq 4m^2/n$ wvhp.*

3. *For $m \leq \sqrt{n} \log^5 n$, we have $t \leq 4 \log^{10} n$ wvhp.*

Proof: Let the m balls be thrown one-by-one. Since the balls occupy at most m bins, when a ball i is thrown the probability that i falls into a bin that is non-empty before i is thrown (referred to as a "non-empty bin" henceforth in this proof) is at most m/n. Thus, the probability that a particular ball is a non-singleton is at most m/n. This establishes Part 1 of the lemma.

Let X be the random variable representing the number of balls that fall into non-empty bins. Thus, the number of non-singletons is at most $2X$. Moreover, X is stochastically dominated by the random variable Y drawn from $B(m, m/n)$. The expected value of Y is m^2/n.

For $m \geq \sqrt{n} \log^5 n$, we apply Equation A.2 with $\varepsilon = 1$, and obtain $\Pr[Y \geq 2s^2/n] \leq e^{-m^2/3n} \leq e^{-(\log^{10} n)/3}$. Therefore the number of non-singletons is at most $4m^2/n$ wvhp, proving Part 2 of the lemma.

For $m \le \sqrt{n}\log^5 n$, we upper bound t by the number of non-singletons in a bag drawn from $\mathcal{D}_{\sqrt{n}\log^5 n, n}$. By Part 2, $t \le 4\log^{10} n$ wvhp, proving Part 3 of the lemma. ∎

The following two lemmas establish bounds on the number of non-pairs. We omit the proofs since they follow the lines of Lemmas 2.8 and 2.9, respectively.

Lemma 2.10: *Let m and n be integers such that $6 \le m \le n$. Let $h : [m] \to [n]$ be a random function drawn from $\mathcal{F}_{m,n}$, and $t(h)$ be the number of non-pairs in bag $h([m])$. If $m \ge n^{4/5}\log^3 n$, then $\delta g(m) \le t(p) \le \Delta g(m)$ wvhp.* ∎

Corollary 2.10.1: *Let m and n be integers such that $6 \le m \le n$, S be a random bag drawn from $D_{m,n}$, and t be the number of non-pairs in S. If $m \ge n^{4/5}\log^3 n$, then $\delta g(m) \le t \le \Delta g(m)$ wvhp.* ∎

Lemma 2.11: *Let m and n be integers such that $6 \le m \le n$, and S be a random bag drawn from $D_{m,n}$. Let t be the random variable denoting the number of non-pairs in S. Then,*

1. *The probability that a particular ball is a non-pair is at most the maximum of $3m^2/n^2$ and $3(\log^{10} n)/n$.*
2. *For $n^{2/3}\log^3 n \le m \le n$, t is at most $12m^3/n^2$ wvhp.*
3. *For $m \le n^{2/3}\log^3 n$, t is at most $12\log^9 n$ wvhp.* ∎

2.4.3 Analysis of Algorithm Alg2

In this section, we analyze the number of rounds taken by **Alg2** before termination. For $0 \le i < \ell$, we have $s_i = t_{i-1}$. Corollaries 2.12.1 and 2.12.2, and Lemma 2.13 establish bounds on s_i in terms of the s'_j's, $0 \le j < i$.

Lemma 2.12: *In* **Alg2**(n, ℓ, c) *let $i \ge \ell$, $s_+ = \Delta s_{i-\ell} t_{i-1}/t_{i-\ell}$ and $s_- = \delta s_{i-\ell} t_{i-1}/t_{i-\ell}$. Then,*

$$\begin{aligned} \Pr[s_i \ge s_+] &\ge e^{-s_+ t_{i-\ell}^2/(2s_{i-\ell}^2 \log^6 n)}, \text{ and} \\ \Pr[s_i \le s_-] &\le e^{-s_- t_{i-\ell}^2/(2s_{i-\ell}^2 \log^6 n)}. \end{aligned}$$

Proof: In round i, **Alg2** removes elements at random from $S_{i-\ell}$ until t_{i-1} elements are left from the subbag $T_{i-\ell}$ of $S_{i-\ell}$. Hence, $\Pr[s_i \geq s_+]$ equals the probability that less than t_{i-1} elements are left from $T_{i-\ell}$ after $s_{i-\ell} - s_+$ elements are removed. This is equal to the probability that less than t_{i-1} elements are chosen from $T_{i-\ell}$ in a random selection of s_+ elements from $S_{i-\ell}$. Applying Lemma 2.6 with $(s,t,s') = (s_{i-\ell}, t_{i-\ell}, s_+)$, the desired probability is at most $e^{-s_+ t_{i-\ell}^2/(2s_{i-\ell}^2 \log^6 n)}$. (Here we use the inequality $(1 - 1/(2\log^3 n))\Delta \geq 1$ for n sufficiently large.)

Similarly, $\Pr[s_i \leq s_-]$ equals the probability that more than t_{i-1} elements are left from $T_{i-\ell}$ after $s_{i-\ell} - s_-$ elements are removed from $S_{i-\ell}$. This is equal to the probability that more than t_{i-1} elements are chosen from $T_{i-\ell}$ in a random selection of s_- elements from $S_{i-\ell}$. Applying Lemma 2.6 with $(s,t,s') = (s_{i-\ell}, t_{i-\ell}, s_-)$, the desired probability is at most $e^{-s_- t_{i-\ell}^2/(2s_{i-\ell}^2 \log^6 n)}$. (Here we use that inequality $(1 + 1/(2\log^3 n))\delta \leq 1$ for n sufficiently large.) ∎

Corollary 2.12.1: *In* **Alg2**(n,ℓ,c)*, if* $i \geq \ell$*,* $s_{i-\ell} t_{i-1}/t_{i-\ell} \geq 2n^{2/3} \log^3 n$ *and* $t_{i-\ell} \geq s_{i-\ell}^2/4n$*, then wvhp,*

$$\delta s_{i-\ell} t_{i-1}/t_{i-\ell} \leq s_i \leq \Delta s_{i-\ell} t_{i-1}/t_{i-\ell}.$$

Proof: Let s_+, s_- be as defined in Lemma 2.12. By Lemma 2.12, we have

$$\Pr[s_i \geq \Delta s_{i-\ell} t_{i-1}/t_{i-\ell}] \leq e^{-s_+ t_{i-\ell}^2/(2s_{i-\ell}^2 \log^6 n)}.$$

Since $s_+, s_{i-\ell} \geq 2n^{2/3} \log^3 n$ and $t_{i-\ell} \geq s_{i-\ell}^2/4n$, the right hand side of the above inequality is at most $e^{-s_+ s_{i-\ell}^2/32n^2 \log^6 n} \leq e^{-\log^3 n/4}$. Similarly, we can prove the desired lower bound on s_i wvhp using the lower bound in Lemma 2.12. (Note that $s_- \geq 2\delta n^{2/3} \log^3 n \geq n^{2/3} \log^3 n$ for n sufficiently large.) ∎

Corollary 2.12.2: *In* **Alg2**(n,ℓ,c)*, if* $s_{i-\ell} t_{i-1}/t_{i-\ell} \geq 2n^{4/5} \log^3 n$ *and* $t_{i-\ell} \geq s_{i-\ell}^3/13n^2$*, then wvhp,*

$$\delta s_{i-\ell} t_{i-1}/t_{i-\ell} \leq s_i \leq \Delta s_{i-\ell} t_{i-1}/t_{i-\ell}.$$

Proof: Let s_+, s_- be as defined in Lemma 2.12. By Lemma 2.12, we have

$$\Pr[s_i \leq \Delta s_{i-\ell} t_{i-1}/t_{i-\ell}] \leq e^{-s_+ t_{i-\ell}^2/(2s_{i-\ell}^2 \log^6 n)}.$$

Since $s_+, s_{i-\ell} \geq 2n^{4/5}\log^3 n$ and $t_{i-\ell} \geq s_{i-\ell}^3/13n^2$, the right hand side of the above inequality is at most $e^{-s_+ s_{i-\ell}^4/(2\cdot 13^2 n^4 \log^6 n)} \leq e^{-(16\log^9 n)/13^2}$. Similarly, we can prove the desired lower bound on s_i wvhp using the lower bound in Lemma 2.12. (Note that $s_- \geq 2\delta n^{4/5}\log^3 n \geq n^{4/5}\log^3 n$ for n sufficiently large.) ∎

Lemma 2.13: *Let $i \geq \ell$. In* **Alg2**(n, ℓ, c)*, if $t_{i-1} \geq \log^2 n$, then $s_i \leq 3s_{i-\ell}t_{i-1}/t_{i-\ell}$ wvhp. If $t_{i-1} \leq \log^2 n$, then $s_i \leq 3s_{i-\ell}(\log^2 n)/t_{i-\ell}$ wvhp.*

Proof: In **Alg2**, $\Pr[s_i \leq 3s_{i-\ell}t_{i-1}/t_{i-\ell}]$ is equal to the probability that more than t_{i-1} elements are selected from $T_{i-\ell}$ in a random selection of $3s_{i-\ell}t_{i-1}/t_{i-\ell}$ elements from $S_{i-\ell}$. If $t_{i-1} \geq \log^2 n$, then we apply Lemma 2.7 with $(s, t, s') = (s_{i-\ell}, t_{i-\ell}, 3s_{i-\ell}t_{i-1}/t_{i-\ell})$ to establish that $s_i \leq 3s_{i-\ell}t_{i-1}/t_{i-\ell}$ wvhp. Similarly, $\Pr[s_i \leq 3s_{i-\ell}(\log^2 n)/t_{i-\ell}]$ is equal to the probability that more than t_{i-1} elements are selected from $T_{i-\ell}$ in a random selection of $3s_{i-\ell}(\log^2 n)/t_{i-\ell}$ elements from $S_{i-\ell}$. If $t_{i-1} \leq \log^2 n$, then we apply Lemma 2.7 with $(s, t, s') = (s_{i-\ell}, t_{i-\ell}, 3s_{i-\ell}(\log^2 n)/t_{i-\ell})$ to establish that $s_i \leq 3s_{i-\ell}(\log^2 n)/t_{i-\ell}$ wvhp. ∎

We now relate t_i to s_i for $i \geq \ell$. First, we prove the following lemma.

Lemma 2.14: *Let m balls be thrown independently and uniformly at random into n bins and S be the associated random bag. Let balls be removed at random from S until the remaining bag, denoted by S', satisfies a given condition C. Let X denote the set of balls that are non-singletons, m' denote $|S'|$, and t' denote $|X|$. Let condition C be such that there exist integers d and u satisfying $d \leq m' \leq u$ wvhp.*

1. *If $d, u \geq n^{2/3}\log^3 n$, then $\delta f(d) \leq t' \leq \Delta f(u)$ wvhp.*
2. *If $u \geq \sqrt{n}\log^5 n$, then $t' \leq 4u^2/n$ wvhp.*
3. *If $u \leq \sqrt{n}\log^5 n$, then $t' \leq 4\log^{10} n$ wvhp.*

4. For any ball x, $\Pr[x \in X] \leq u^2/(mn) + 1/n^c$ for any real constant $c \geq 0$.

Proof: Consider the experiment of removing balls one-by-one at random from S. Let S_1 (resp., X_1) be the bag (resp., set of non-singleton balls) obtained when $m-u$ balls have been removed and S_2 be the bag obtained when $m-d$ balls have been removed. Therefore $|S_1| = u$ and $|S_2| = d$. Also, S_2 is a subbag of S_1. Wvhp, the condition C occurs after $m-u$ balls are removed and before $m-d$ balls are removed from S. Thus wvhp, S' is a subbag of S_1 and a superbag of S_2. Let t_1 (resp. t_2) denote the number of non-singletons in S_1 (resp., S_2). Hence $t_2 \leq t' \leq t_1$ wvhp. Note that by Corollary 2.1.1, S_1 and S_2 have probability distributions $\mathcal{D}_{u,n}$ and $\mathcal{D}_{d,n}$, respectively.

1. If $d, u \geq n^{2/3} \log^3 n$, then by Corollary 2.8.1, $t_2 \geq \delta f(d)$ and $t_1 \leq \Delta f(u)$ wvhp, thus establishing Part 1 of the lemma.

2. If $u \geq \sqrt{n} \log^5 n$, then by Part 2 of Lemma 2.9, $t_1 \leq 4u^2/n$ wvhp, thus establishing Part 2 of the lemma.

3. If $u \leq \sqrt{n} \log^5 n$, then by Part 3 of Lemma 2.9, $t_1 \leq 4 \log^{10} n$ wvhp. Hence $t' \leq 4 \log^{10} n$ wvhp, thus establishing Part 3 of the lemma.

4. For any ball x, $\Pr[x \in X] \leq \Pr[x \in X_1] + 1/n^c$ for any $c \geq 0$. By symmetry, the probability that x remains when u balls are left is u/m. Since S_1 is drawn uniformly at random from $\mathcal{D}_{u,n}$, by Part 1 of Lemma 2.9, $\Pr[x \in X_1] \leq (u/m)(u/n) = u^2/(mn)$, thus establishing Part 4 of the lemma.

$\square$

Corollary 2.14.1: *In* **Alg2**$(n, \ell, 1)$*, let $i \geq \ell$ and $d, u \geq 0$ be integers such that $d \leq s_i \leq u$ wvhp. If $t' = t_i$, then Parts 1 through 3 of Lemma 2.14 hold. Also, for any ball $x \in [n]$, the probability that x remains after round i is at most $(u^2/n^2) + 1/n^c$ for any real constant $c \geq 0$.*

Proof: Fix integer $i \geq \ell$. Let $k = i \bmod \ell$. Consider the sequence of bags $\{S_{j\ell+k} \mid j \geq 0\}$ in **Alg2**. Bag S_k is obtained by throwing t_{k-1} (n if $k = 0$) balls into n bins. Bag

$S_{j\ell+k}$, $j > 0$, is obtained by removing balls at random from $S_{(j-1)\ell+k}$ until $t_{(j-1)\ell+k-1}$ balls are left in a particular subbag $T_{(j-1)\ell+k}$ of $S_{(j-1)\ell+k}$.

Bag S_k can be obtained equivalently the following way: remove $n - t_{k-1}$ (0 if $k = 0$) balls at random from S that is a random bag drawn from $\mathcal{D}_{n,n}$. Thus each bag $S_{j\ell+k}$, $j \geq 0$ (S_i, in particular), can be viewed as having been obtained from bag S by removing balls at random until a certain condition (say C) holds. For bag S_i thus obtained, it is given that $d \leq |S_i| \leq u$ wvhp. We invoke Lemma 2.14, substituting $(S, S_i, s_i, t', n, d, u, C)$ for $(S, S', m', t', m, d, u, C)$, to establish the desired claims. $\square$

Lemma 2.15 is the analogue of Corollary 2.14.1 for $\mathbf{Alg2}(n, \ell, 2)$ and can be proved along the same lines.

Lemma 2.15: *In* $\mathbf{Alg2}(n, \ell, 2)$, *let* $i \geq \ell$ *and* $d, u \geq 0$ *be integers such that* $d \leq s_i \leq u$ *wvhp.*

1. *If* $d, u \geq n^{4/5} \log^3 n$, *then* $\delta g(d) \leq t_i \leq \Delta g(u)$ *wvhp.*
2. *If* $u \geq n^{2/3} \log^3 n$, *then* $t_i \leq 12u^3/n^2$ *wvhp.*
3. *If* $u \leq n^{2/3} \log^3 n$, *then* $t_i \leq 12 \log^9 n$ *wvhp.*
4. *For any* $x \in [n]$ *the probability that* x *remains after round* i *is at most the maximum of* $3u^3/n^3 + 1/n^c$ *and* $(u \log^{10} n)/n^2 + 1/n^c$ *for any real constant* $c \geq 0$. $\square$

2.4.3.1 Analysis for the 1-Collision Crossbar

Using the results of Section 2.4.2, we show that the probability that $\mathbf{Alg2}(n, \ell, 1)$ deviates significantly from the "expected behavior" is polynomially small. Let s'_i be defined as follows:

$$s'_i = \begin{cases} n & \text{if } i = 0, \\ f(s'_{i-1}) & \text{if } 0 < i < \ell, \text{ and} \\ s'_{i-\ell} \cdot \frac{f(s'_{i-1})}{f(s'_{i-\ell})} & \text{otherwise.} \end{cases}$$

Let $t'_i = f(s'_i)$ for all $i \geq 0$. (Note that for all $i \geq 0$, s'_i is the expected value of s_i given that $(s_j, t_j) = (s'_j, t'_j)$ for $0 \leq j < i$. Similarly, t'_i is the expected value of t_i given that $(s_j, t_j) = (s'_j, t'_j)$ for $0 \leq j < i$ and $s_i = s'_i$.)

Lemma 2.16: *In* $\mathbf{Alg2}(n, \ell, 1)$*, for all* $0 \leq i \leq \frac{5}{2}\log_3 \log n$*, if* $s'_i \geq 4n^{2/3}\log^3 n$ *and* n *is sufficiently large, then wvhp,*

$$\delta^{3^i} s'_i \leq s_i \leq \Delta^{3^i} s'_i, \text{ and} \tag{2.3}$$

$$\delta^{2\cdot 3^i+1} t'_i \leq t_i \leq \Delta^{2\cdot 3^i+1} t'_i. \tag{2.4}$$

Proof: We use induction on i. For the basis, $i = 0$ and $s_0 = n = s'_0$. By Lemma 2.8, $\delta f(n) \leq t_0 \leq \Delta f(n)$ wvhp. Since $t'_0 = f(n)$, the desired claims hold for $i = 0$.

Assume the claim holds for all $j < i$. We first establish Equation 2.3 from which we then derive Equation 2.4. We consider two cases. If $i < \ell$, then $s_i = t_{i-1}$. Since $s'_{i-1} \geq s'_i \geq 4n^{2/3}\log^3 n$, we obtain from the induction hypothesis that $\delta^{2\cdot 3^{i-1}+1} t'_{i-1} \leq t_{i-1} \leq \Delta^{2\cdot 3^{i-1}+1} t'_{i-1}$ wvhp. Since $3^i \geq 2 \cdot 3^{i-1} + 1$ for $i < \ell$, $\delta^{c^i} s'_i \leq s_i \leq s'_i \Delta^{c^i}$ wvhp.

If $i \geq \ell$, we use Corollary 2.12.1 to bound s_i. By the induction hypothesis and using the inequality $\min\{s'_{i-1}, s'_{i-\ell}\} \geq 4n^{2/3}\log^3 n$,

$$\begin{aligned}
\delta^{c^{i-\ell}} s'_{i-\ell} &\leq s_{i-\ell} \leq \Delta^{c^{i-\ell}} s'_{i-\ell}, \\
\delta^{2\cdot 3^{i-\ell}+1} t'_{i-\ell} &\leq t_{i-\ell} \leq \Delta^{2\cdot 3^{i-\ell}+1} t'_{i-\ell}, \text{ and} \\
\delta^{2\cdot 3^{i-1}+1} t'_{i-1} &\leq t_{i-1} \leq \Delta^{2\cdot 3^{i-1}+1} t'_{i-1}.
\end{aligned}$$

Substituting appropriate bounds on $s_{i-\ell}, t_{i-\ell}$, and t_{i-1}, we get the following bounds on $s = s_{i-\ell} t_{i-1}/t_{i-\ell}$ wvhp:

$$\frac{\delta^{2\cdot 3^{i-1}+3^{i-\ell}+1} s'_{i-\ell} t'_{i-1}}{\Delta^{2\cdot 3^{i-\ell}+1} t'_{i-\ell}} \leq s \leq \frac{\Delta^{2\cdot 3^{i-1}+3^{i-\ell}+1} s'_{i-\ell} t'_{i-1}}{\delta^{2\cdot 3^{i-\ell}+1} t'_{i-\ell}}.$$

Since $\delta \leq \Delta^{-1}$ and $\Delta^2 \geq \delta^{-1}$ for n sufficiently large, we have

$$\frac{\delta^{2\cdot 3^{i-1}+3\cdot 3^{i-\ell}+2} s'_{i-\ell} t'_{i-1}}{t'_{i-\ell}} \leq s \leq \frac{\Delta^{2\cdot 3^{i-1}+5\cdot 3^{i-\ell}+3} s'_{i-\ell} t'_{i-1}}{t'_{i-\ell}}. \tag{2.5}$$

Since $3^\ell \geq 2 \cdot 3^{\ell-1} + 9$ for $\ell \geq 3$, $3^i \geq 2 \cdot 3^{i-1} + 3 \cdot 3^{i-\ell} + 2$. Therefore $s \geq \delta^{3^i} s'_{i-\ell} t'_{i-1}/t'_{i-\ell} = \delta^{3^i} s'_i$ wvhp. Since $i \leq \frac{5}{2}\log_3 \log n$, we have $3^i \leq \log^{5/2} n$. Hence, $\delta^{3^i} \geq \alpha$ for any real

$\alpha < 1$ for n sufficiently large. We thus have $s \geq 2n^{2/3}\log^3 n$. We next show that $t_{i-\ell} \geq s_{i-\ell}^2/4n$ wvhp. By the induction hypothesis, $t_{i-\ell} \geq \delta^{2\cdot 3^{i-\ell}+1} t'_{i-\ell} = \delta^{2\cdot 3^{i-\ell}+1} f(s'_{i-\ell})$ wvhp. Since $f(s'_{i-\ell}) \geq (s'_{i-\ell})^2/3n$ and $s_{i-\ell} \leq \Delta^{3^{i-\ell}} s'_{i-\ell}$ wvhp, we have

$$t_{i-\ell} \geq \frac{\delta^{2\cdot 3^{i-\ell}+1} s_{i-\ell}^2}{3\Delta^{2\cdot 3^{i-\ell}} n} \geq \frac{\delta^{4\cdot 3^{i-\ell}+1} s_{i-\ell}^2}{3n}$$

wvhp. In the last equation we use $\delta \leq \Delta^{-1}$. For any real $\alpha < 1$, $\delta^{3^{i-\ell}} \geq \delta^{3^i} \geq \alpha$ for n sufficiently large. Therefore, $\delta^{4\cdot 3^{i-\ell}+1} \geq 3/4$ for n sufficiently large and thus it follows that $t_{i-\ell} \geq s_{i-\ell}^2/4n$ wvhp.

We now apply Corollary 2.12.1 to obtain $\delta s \leq s_i \leq \Delta s$ wvhp. By Equation 2.5, wvhp,

$$\frac{\delta^{2\cdot 3^{i-1}+3\cdot 3^{i-\ell}+3} s'_{i-\ell} t'_{i-1}}{t'_{i-\ell}} \leq s_i \leq \frac{\Delta^{2\cdot 3^{i-1}+5\cdot 3^{i-\ell}+4} s'_{i-\ell} t'_{i-1}}{t'_{i-\ell}}.$$

Since for $\ell \geq 3$, $3^\ell \geq 2 \cdot 3^{\ell-1} + 9$, we have $3^i \geq 2 \cdot 3^{i-1} + 5 \cdot 3^{i-\ell} + +4$ and $3^i \geq 2 \cdot 3^{i-1} + 3 \cdot 3^{i-\ell} + 3$. Since $s'_i = s'_{i-\ell} t'_{i-1}/t'_{i-\ell}$, Equation 2.3 holds wvhp.

We now invoke Part 1 of Corollary 2.14.1 to obtain bounds on t_i. Note that $\delta^{3^i} s'_i, \Delta^{3^i} s'_i \geq n^{2/3}\log^3 n$ for n sufficiently large. Thus wvhp,

$$\delta f(\delta^{3^i} s'_i) \leq t_i \leq \Delta f(\Delta^{3^i} s'_i),$$

and hence by Corollary C.3.1,

$$\delta^{2\cdot 3c^i+1} f(s'_i) \leq t_i \leq \Delta^{2\cdot 3^i+1} f(s'_i).$$

Since $t'_i = f(s'_i)$, Equation 2.4 follows wvhp. $\square$

Lemma 2.16 implies that we can analyze $\mathbf{Alg2}(n, \ell, 1)$ by studying how s'_i decreases as i increases.

Lemma 2.17: *For all $0 \leq i < \ell$, we have*

$$\prod_{0 \leq j < i+1} s'_j = s'_0 \prod_{0 \leq j < i} f(s'_j),$$

and for $i \geq \ell$, we have

$$\prod_{i-\ell+1 \leq j < i+1} s'_j = s'_0 \prod_{i-\ell+1 \leq j < i} f(s'_j).$$

Proof: For $0 \le i < \ell$, the desired claim follows directly from the definition of s'_j, $0 \le j < i+1$. Observe that for $i = \ell - 1$, we have $\prod_{i-\ell+1 \le j < i+1} s'_j = s'_0 \prod_{i-\ell+1 \le j < i} f(s'_{j-1})$. We use this equality as a basis for the case $i \ge \ell$. Assume that for $\ell - 1 \le k < i$, we have $\prod_{k-\ell+1 \le j < k+1} s'_j = s'_0 \prod_{k-\ell+1 \le j < k} f(s'_j)$. Then

$$\begin{aligned}
\prod_{i-\ell+1 \le j < i+1} s'_j &= \frac{s'_i}{s'_{i-\ell}} \prod_{i-\ell \le j < i} s'_j \\
&= \frac{s'_0 f(s'_{i-1})}{f(s'_{i-\ell})} \prod_{i-\ell \le j < i-1} f(s'_j) \\
&= s'_0 \prod_{i-\ell+1 \le j < i} f(s'_j).
\end{aligned}$$

∎

Lemma 2.18: *For all $1 \le i < \ell$, if s'_{i-1} and n are sufficiently large, then*

$$\frac{1}{3^{i-1}} \prod_{0 \le j < i} \frac{s'_j}{n} \le \frac{s'_i}{n} \le \prod_{0 \le j < i} \frac{s'_j}{n}.$$

For $i \ge \ell$, if s'_{i-1} and n are sufficiently large, then

$$\frac{1}{3^{\ell-1}} \prod_{i-\ell+1 \le j < i} \frac{s'_j}{n} \le \frac{s'_i}{n} \le \prod_{i-\ell+1 \le j < i} \frac{s'_j}{n}.$$

Proof: By Lemma 2.17 and Lemma C.1, if s'_{i-1} and n are sufficiently large, then for all $0 \le i < \ell$, we have

$$\frac{s'_0}{3^{i-1}} \prod_{0 \le j < i} \frac{(s'_j)^2}{n} \le \prod_{0 \le j < i+1} s'_j \le s'_0 \prod_{0 \le j < i} \frac{(s'_j)^2}{n},$$

and the claim of the lemma follows after dividing by $s'_0 \prod_{0 \le j < i} s'_j$. By Lemma 2.17 and Lemma C.1, if s'_{i-1} and n are sufficiently large, then for all $i \ge 0$, we have

$$\frac{s'_0}{3^{\ell-1}} \prod_{i-\ell+1 \le j < i} \frac{(s'_j)^2}{n} \le \prod_{i-\ell+1 \le j < i+1} s'_j \le s'_0 \prod_{i-\ell+1 \le j < i} \frac{(s'_j)^2}{n},$$

and the claim of the lemma follows after dividing by $s'_0 \prod_{i-\ell+1 \le j < i} s'_j$. ∎

Lemma 2.18 can be used to analyze $\mathbf{Alg2}(n, \ell, 1)$ for any $\ell \ge 2$. Let $w_i = \log_r(n/s_i)$ and $w'_i = \log_r(n/s'_i)$, where $r = n/f(n)$. (Note that $e/(e-1) \le r \le 2$ for all $n \ge 2$.)

Lemma 2.19: *In* $\mathbf{Alg2}(n, \ell, 1)$*, for all* $i > 0$*, if* s'_{i-1} *and* n *are sufficiently large, then*

$$\sum_{1 \le j \le \min\{i, \ell-1\}} w'_{i-j} \le w'_i \le 2\log_r 3 + \sum_{1 \le j \le \min\{i, \ell-1\}} w'_{i-j}.$$

Proof: Follows directly from the definition of w'_i and Lemma 2.18. ∎

We are now ready to place a bound on the number of rounds taken by $\mathbf{Alg2}(n, \ell, 1)$ before termination. We first show that for $\ell \ge 3$, $\mathbf{Alg2}(n, \ell, 1)$ terminates in $O(\log\log n)$ rounds whp. To prove this upper bound, it is enough to consider the case $\ell = 3$. The upper bound for $\ell > 3$ follows from the bound for $\ell = 3$.

Lemma 2.20: *In* $\mathbf{Alg2}(n, 3, 1)$ *for all* $i > 0$*, if* s'_{i-1} *and* n *are sufficiently large, then* $w'_i \ge p_1^{i-1}$*, where* $p_1 > 1$ *satisfies the following inequality:*

$$p_1^2 - p_1 - 1 \le 0 \tag{2.6}$$

Proof: The proof is by induction on i. For the induction basis, i is 1. We have $s'_1 = n/r$, hence $w'_1 = 1 = p_1^0$.

Let the claimed lower bound on w'_i hold for all $0 < j < i$, $i > 1$. By Lemma 2.19 and the induction hypothesis,

$$w'_i \ge p_1^{i-3} + p_1^{i-2}.$$

It thus follows from Equation 2.6 that $w'_k \ge p_1^{k-1}$. ∎

We now place an upper bound on the number of rounds taken by $\mathbf{Alg2}(n, 3, 1)$ before termination.

Lemma 2.21: *There exists an integer* $j = O(\log\log n)$ *such that* $s_j \le n^{2/5}$ *wvhp in* $\mathbf{Alg2}(n, 3, 1)$.

Proof: Let $\phi = (1+\sqrt{5})/2$. Since $\phi^2 - \phi - 1 = 0$, Lemma 2.20 implies that $w'_i \ge \phi^{i-1}$ for all $i > 0$. Let $k = \min\{i : w'_i \ge \log_r(\frac{n^{1/3}}{4\log^3 n})\}$. For $i = \lceil \log_\phi \log_r \frac{n^{1/3}}{4\log^3 n} \rceil + 1$, $w'_i \ge \log_r(\frac{n^{1/3}}{4\log^3 n})$. Therefore, $k \le \log_\phi \log_r \frac{n^{1/3}}{4\log^3 n} + 2$. (Also note that since $w'_2 \le 1 + 2\log_r 3$, $k \ge 3$ for n sufficiently large.) Since $\phi^{5/2} > 3$, $k \le 5/2 \log_3 \log n$ for n sufficiently large. Thus, Equations 2.3 and 2.4 of Lemma 2.16 hold for all $i < k$. (Also note that $s'_k = n/r^{w'_k} \leq 4n^{2/3}\log^3 n$.)

By Lemma 2.16, $t_{k-1} \geq \delta^{2\cdot 3^{k-1}+1} t'_{k-1}$ wvhp. Since $t'_{k-1} = f(s'_{k-1}) \geq (s'_{k-1})^2/3n$, we have

$$t_{k-1} \geq 16\delta^{2\cdot 3^{k-1}+1}(n^{1/3}\log^6 n)/3 \geq \log^2 n$$

wvhp for n sufficiently large. By Lemma 2.13, $s_k \leq 3s_{k-3}t_{k-1}/t_{k-3}$ wvhp. Substituting appropriate bounds on s_{k-3}, t_{k-3}, and t_{k-1} from Lemma 2.16, we have wvhp

$$s_k \quad \leq \quad 3\Delta^{2\cdot 3^{k-1}+5\cdot 3^{k-3}+4} s'_k \leq 3\Delta^{3^k} s'_k \leq 4s'_k. \tag{2.7}$$

The last equation follows from the inequality $\Delta^{3^k} < \alpha$ for any real $\alpha < 1$ and n sufficiently large. We consider two cases depending on the value of s'_k.

Case 1: $s'_k \leq \sqrt{n}\log^5 n$. By Equation 2.7, $s_k \leq 4\sqrt{n}\log^5 n$ wvhp. Therefore, by Part 3 of Lemma 2.14.1, $t_k \leq 64\log^{10} n$ wvhp. We consider two cases. If $t_k \geq \log^2 n$, by Lemma 2.13, $s_{k+1} \leq 3s_{k-2}t_k/t_{k-2}$ wvhp. If $t_k \leq \log^2 n$, then $s_{k+1} \leq 3s_{k-2}\log^2 n/t_{k-2}$. In any case, $s_{k+1} \leq (192 s_{k-2}\log^{10} n)/t_{k-2}$ wvhp. We now substitute appropriate bounds on s_{k-2} and t_{k-2} from Lemma 2.16 and obtain that wvhp,

$$\begin{aligned} s_{k+1} \quad &\leq \quad \frac{192\Delta^{3^{k-2}} s'_{k-2}\log^{10} n}{\delta^{2\cdot 3^{k-2}+1} t'_{k-2}} \\ &\leq \quad \frac{576 n\Delta^{5\cdot 3^{k-2}+2}\log^{10} n}{s'_{k-2}} \\ &\leq \quad 144\Delta^{3^k} n^{1/3}\log^7 n \\ &\leq \quad n^{2/5} \end{aligned}$$

for n sufficiently large. (Note: The penultimate equation follows from the inequalities $3^k \geq 5\cdot 3^{k-2}+2$ and $s'_{k-2} \geq 4n^{2/3}\log^3 n$.)

Case 2: $s'_k \geq \sqrt{n}\log^5 n$. By Equation 2.7, $s_k \leq 4s'_k$ wvhp. We again consider two cases, depending on whether $t_k \leq \log^2 n$ or $t_k \geq \log^2 n$.

If $t_k \leq \log^2 n$ then Lemma 2.13 implies that $s_{k+1} \leq 3s_{k-2}\log^2 n/t_{k-2}$ wvhp. Arguing as in Case 2, s_{k+1} is at most $n^{2/5}$ wvhp.

If $t_k \geq \log^2 n$ then Lemma 2.13 implies that $s_{k+1} \leq 3s_{k-2}t_k/t_{k-2}$ wvhp. Since $s_k \leq 4s'_k$, by Part 2 of Lemma 2.14.1, $t_k \leq 64(s'_k)^2/n \leq 192t'_k$ wvhp. Substituting this upper bound on t_k and appropriate bounds on s_{k-2} and t_{k-2} obtained from Lemma 2.16, we have $s_{k+1} \leq 1000s'_{k+1}$ wvhp for n sufficiently large. We now derive an upper bound on s'_{k+1}.

By Lemma 2.19, $w'_k \leq w'_{k-1} + w'_{k-2} + 2\log_r 3$. Since $w'_{k-1} \geq w'_{k-2}$, we have

$$w'_{k-1} \geq \frac{1}{2}(w'_k - 2\log_r 3) \geq \frac{\log_r n}{6} - \frac{3\log_r \log n}{2} - \log_r 6.$$

Thus, by Lemma 2.19,

$$\begin{aligned} w'_{k+1} &\geq w'_k + w'_{k-1} \\ &\geq \frac{\log_r n}{2} - \frac{9\log_r \log n}{2} - \log_r 24, \text{ and} \\ s'_{k+1} &\leq \sqrt{n}\log^5 n, \end{aligned}$$

for n sufficiently large. We now apply an analysis similar to Case 2 with k replaced by $k+1$ to establish that s_{k+2} is at most $n^{2/5}$ wvhp.

Cases 1 and 2 establish that after $j = k+2 = O(\log\log n)$ rounds, s_j is at most $n^{2/5}$ wvhp. □

Lemma 2.22: *For any ball $x \in [n]$, the probability that x remains after $O(\log\log n)$ rounds of* **Alg2**$(n,3,1)$ *is at most $2/n^{6/5}$ for n sufficiently large.*

Proof: By Lemma 2.21, after $j = O(\log\log n)$ rounds, $s_j \leq n^{2/5}$ wvhp. By Corollary 2.14.1, the probability that x remains after round j is at most $2n^{4/5}/n^2$ for n sufficiently large. Since $2n^{4/5}/n^2 = 2/n^{6/5}$, the desired claim follows. □

The following theorem is an easy consequence of the above lemma.

Theorem 2.1: **Alg2**$(n,3,1)$ *terminates in $O(\log\log n)$ rounds whp.* □

Corollary 2.1.1: *For $\ell \geq 3$,* **Alg2**$(n,\ell,1)$ *terminates in $O(\log\log n)$ rounds whp.* □

We now establish a lower bound on the number of rounds taken by $\mathbf{Alg2}(n,\ell,1)$ before termination. We first place an upper bound on w'_i that complements the lower bound of Lemma 2.20. (Note that the following lemma applies for all ℓ, while ℓ equals 3 in Lemma 2.20.)

Lemma 2.23: *In* $\mathbf{Alg2}(n,\ell,1)$ *for all* $i>0$, *if* s'_{i-1} *and* n *are sufficiently large, then* $w'_i \le p_2^{i-1}$, *where* $p_2>1$ *satisfies the following inequality:*

$$p_2^k - 2\log_r 3 - \sum_{0\le j<k} p_2^j \;\ge 0 \quad \textit{for all } k\le \ell-1 \tag{2.8}$$

Proof: We first note that $w'_0=0$. The proof of the lemma is by induction on i. For the induction basis, $i=1$. We have $s'_1=n/r$, and hence $w'_1=1=p_2^0$.

Let the claimed upper bound on w'_i hold for all $0<j<i$, $i>1$. By Lemma 2.19 and the induction hypothesis, we have:

$$w'_i \;\le\; 2\log_r 3 + \sum_{1\le j\le \min\{i-1,\ell-1\}} p_2^{i-j-1}.$$

By Equation 2.8 and the inequality $p_2>1$, $2\log_r 3+\sum_{1\le j\le\min\{i-1,\ell-1\}} p_2^{i-j-1}\le p_2^{i-1}$. This completes the proof of the desired claim. ∎

Theorem 2.2: *For any* $\ell\ge 3$, $\mathbf{Alg2}(n,\ell,1)$ *terminates in* $\Omega(\log\log n)$ *rounds wvhp.*

Proof: One solution to Equation 2.8 is $p_2=2\log_r 3+1=O(1)$. Thus, by Lemma 2.20, $w'_i\le p_2^{i-1}$ for all $i>0$. After $k=\lfloor\log_{p_2}((\log_r n)/4)\rfloor$ rounds, $w'_k\le(\log_r n)/4$ and $s'_k\ge n^{3/4}$. For n sufficiently large, $n^{3/4}\ge 4n^{2/3}\log^3 n$. Therefore, by Lemma 2.16, $t_k\ge \delta^{2\cdot 3^k+1}t'_k\ge\delta^{2\cdot 3^k+1}(s'_k)^2/3n>0$ for n sufficiently large. This shows that $\mathbf{Alg2}(n,\ell,1)$ executes at least $\log_{p_2}((\log_r n)/4)\ge\log_{p_2}((\log n)/4)=\Omega(\log\log n)$ rounds before termination. ∎

The recurrence in Lemma 2.18 for $\ell=2$ yields $s'_{i+1}/n\ge s'_i/3n$ for all $i\ge 0$. Thus $w'_i=O(i)$. Using an analysis similar to the above theorem we establish an $\Omega(\log n)$ lower bound for $\mathbf{Alg2}(n,2,1)$.

Theorem 2.3: $\mathbf{Alg2}(n,2,1)$ *terminates in* $\Omega(\log n)$ *rounds wvhp.* ∎

2.4.3.2 Analysis for the 2-Collision Crossbar

The analysis of **Alg2** for the 2-collision crossbar is similar to that for the 1-collision crossbar. We begin by defining s'_i as follows:

$$s'_i = \begin{cases} n & \text{if } i = 0, \\ g(s'_{i-1}) & \text{if } 0 < i < \ell, \\ s'_{i-\ell} \cdot \frac{g(s'_{i-1})}{g(s'_{i-\ell})} & \text{otherwise.} \end{cases}$$

For all $i \geq 0$ let $t'_i = g(s'_i)$. As in Section 2.4.3.1, we next show that s'_i and t'_i are good approximations for s_i and t_i, respectively (Lemma 2.24). Lemmas 2.25 and 2.26 determine the rate at which s'_i decreases with increasing i. Using Lemma 2.24, we can then determine the rate of change of s_i as i increases. The proofs of Lemmas 2.24, 2.25, and 2.26 are analogous to those of Lemmas 2.16, 2.17, and 2.18.

Lemma 2.24: *Let c be the positive root of $c^2 = 4c + 13$. In* **Alg2**$(n, \ell, 2)$*, for all $0 \leq i \leq (11/4)\log_c \log n$, if $s'_i \geq 4n^{4/5}\log^3 n$, then wvhp,*

$$\delta^{c^i} s'_i \leq s_i \leq \Delta^{c^i} s'_i, \text{ and} \tag{2.9}$$

$$\delta^{4c^i+1} t'_i \leq t_i \leq \Delta^{4c^i+1} t'_i. \tag{2.10}$$

□

Lemma 2.25: *For all $0 \leq i < \ell$, we have*

$$\prod_{0 \leq j < i+1} s'_j = s'_0 \prod_{0 \leq j < i} g(s'_j),$$

and for $i \geq \ell$, we have

$$\prod_{i-\ell+1 \leq j < i+1} s'_j = s'_0 \prod_{i-\ell+1 \leq j < i} g(s'_j).$$

□

Lemma 2.26: *For all $1 \leq i < \ell$, if s'_{i-1} and n are sufficiently large, then*

$$\frac{1}{12^{i-1}} \prod_{0 \leq j < i} \left(\frac{s'_j}{n}\right)^2 \leq \frac{s'_i}{n} \leq \prod_{0 \leq j < i} \left(\frac{s'_j}{n}\right).$$

For $i \geq \ell$, if s'_{i-1} and n are sufficiently large, then

$$\frac{1}{12^{\ell-1}} \prod_{i-\ell+1 \leq j < i} \left(\frac{s'_j}{n}\right)^2 \leq \frac{s'_i}{n} \leq \prod_{i-\ell+1 \leq j < i} \left(\frac{s'_j}{n}\right)^2.$$

∎

Let $w_i = \log_r(n/s_i)$ and $w'_i = \log_r(n/s'_i)$, where $r = n/g(n)$. (Note that $e/(e-2) \leq r \leq 9$ for $n \geq 3$.)

Lemma 2.27: *In* **Alg2**$(n, \ell, 2)$*, for all $i > 0$, if s'_{i-1} and n are sufficiently large, then*

$$\sum_{1 \leq j \leq \min\{i, \ell-1\}} 2w'_{i-j} \leq w'_i \leq \log_r 12 + \sum_{1 \leq j \leq \min\{i, \ell-1\}} 2w'_{i-j}.$$

Proof: Follows directly from the definition of w'_i and Lemma 2.26. ∎

We are now ready to place a tight bound on the number of rounds taken by **Alg2**$(n, \ell, 2)$ before termination. We first show that **Alg2**$(n, 2, 2)$ terminates in $O(\log \log n)$ rounds whp. The upper bound for $\ell > 2$ follows from the bound for $\ell = 2$.

Lemma 2.28: *In* **Alg2**$(n, 2, 2)$ *for all $i > 0$, if s'_{i-1} and n are sufficiently large, then $w'_i \geq 2^{i-1}$.*

Proof: The proof is by induction on i. For the induction basis, $i = 1$. We have $s'_1 = n/r$, hence $w'_1 = 1 = 2^0$. Let the claimed lower bound on w'_i hold for all $0 < j < i$, $i > 1$. By Lemma 2.19 and the induction hypothesis, $w'_i \geq 2 \cdot 2^{i-2} = 2^{i-1}$. ∎

We now place an upper bound on the number of rounds taken by **Alg2**$(n, 2, 2)$ before termination.

Lemma 2.29: *There exists an integer $j = O(\log \log n)$ such that $s_j \leq n^{5/8}$ wvhp in* **Alg2**$(n, 2, 2)$.

Proof: By Lemma 2.28, $w'_i \geq 2^{i-1}$ for all $i > 0$. Let $k = \min\{i \mid w'_i \geq \log_r \frac{n^{1/3}}{4\log^3 n}\}$. Therefore $k \leq \lceil \log \log_r \frac{n^{1/5}}{4\log^3 n} \rceil + 1 \leq \log \log_r \frac{n^{1/5}}{4\log^3 n} + 2$. (Also note that since $w'_1 = 1$, we have $k \geq 2$ for n sufficiently large.) Now we apply Lemma 2.24 with $\ell = 2$. Let α

be the root of the equation $\alpha^2 = 4\alpha + 13$. Since $2^{11/4} > \alpha$, $k \leq (11/4)\log_\alpha \log_r n$ for n sufficiently large. Therefore by Lemma 2.24, $t_{k-1} \geq \delta^{4\alpha^{k-1}+1} t'_{k-1}$ wvhp. Since $t'_{k-1} \geq (s'_{k-1})^3/(12n^2)$, we have $t_{k-1} \geq \delta^{4\alpha^{k-1}+1}(16n^{2/5}\log^9 n)/3 \geq \log^2 n$ for n sufficiently large. By Lemma 2.13, $s_k \leq 3s_{k-2}t_{k-1}/t_{k-2}$ wvhp. Substituting the appropriate bounds on s_{k-2}, t_{k-1} and t_{k-2} given by Lemma 2.24, we have wvhp,

$$\begin{aligned} s_k &\leq \frac{3\Delta^{4\alpha^{k-1}+\alpha^{k-2}+1} s'_{k-2} t'_{k-1}}{\delta^{4\alpha^{k-2}+1} t'_{k-2}} \\ &\leq \frac{3\Delta 4\alpha^{k-1} + 9\alpha^{k-2} + 3 s'_{k-2} t' k - 1}{t'_{k-2}}. \end{aligned}$$

Since $\alpha^2 = 4\alpha + 13$, we have $\alpha^k \geq 4\alpha^{k-1} + 9\alpha^{k-2} + 3$. Therefore,

$$s_k \leq 3\Delta^{\alpha^k} s'_k \leq 4s'_k, \tag{2.11}$$

wvhp for n sufficiently large.

We consider two cases, depending on whether $t_k \leq \log^2 n$ or $t_k > \log^2 n$.

If $t_k \leq \log^2 n$, then by Lemma 2.13, $s_{k+1} \leq 3s_{k-1}\log^2 n/t_{k-1}$ wvhp. Substituting appropriate bounds on s_{k-1} and t_{k-1} given by Lemma 2.24, we have wvhp,

$$\begin{aligned} s_{k+1} &\leq \frac{3\Delta^{\alpha^{k-1}} s'_{k-1}\log^2 n}{\delta^{4\alpha^{k-1}+1} t'_{k-1}} \\ &\leq \frac{36\Delta^{9\alpha^{k-1}+2} n^2 \log^2 n}{(s'_{k-1})^2} \\ &\leq \frac{9\Delta^{\alpha^{k+1}} n^{2/5}}{4\log^4 n} \\ &\leq \frac{92^\alpha n^{2/5}}{4\log^4 n} \\ &\leq n^{5/8} \end{aligned}$$

for n sufficiently large. (The second equation follows from the lower bound on t'_{k-1} given by Lemma C.2. In the third equation we use $s'_{k-1} \geq n^{4/5}\log^3 n$. And in the penultimate equation we use $\Delta^{\alpha^k} \leq 2$ for n sufficiently large.)

If $t_k > \log^2 n$ then by Lemma 2.13, $s_{k+1} \leq 3s_{k-1}t_k/t_{k-1}$ wvhp. If $s'_k \leq (n^{2/3}\log^3 n)/4$, then since $s_k \leq 4s'_k$ wvhp, by Part 3 of Lemma 2.15, $t_k \leq 12\log^9 n$

wvhp. Hence, as in the case $t_k \leq \log^2 n$ above, we can establish that t_{k+1} is zero whp. If $s'_k \geq (n^{2/3}\log^3 n)/4$, then by Lemma 2.15, $t_k \leq 768(s'_{k-1})^3/n^2$ wvhp. Therefore, by Lemma C.2, $t_k \leq 12 \cdot 768 t'_k$. Substituting this bound on t_k and appropriate bounds on s_{k-1} and t_{k-1} given by Lemma 2.24, we have wvhp,

$$\begin{aligned} s_{k+1} &\leq \frac{36 \cdot 768 \Delta^{\alpha^{k-1}} s'_{k-1} t'_k}{\delta^{4\alpha^{k-1}+1} t'_{k-1}} \\ &\leq 36 \cdot 768 \Delta^{9\alpha^{k-1}+2} s'_{k+1} \\ &\leq 36 \cdot 768 \cdot 2^{\alpha} s'_{k+1}. \end{aligned}$$

By Lemma 2.27 with $\ell = 2$, $w'_{k+1} \geq 2w'_k$. Thus $w'_{k+1} \geq 2\log_r(\frac{n^{1/5}}{4\log^3 n})$, and $s'_{k+1} \leq 16n^{3/5}\log^6 n$. Hence, $s_{k+1} \leq n^{5/8}$ for n sufficiently large. ∎

In Lemma 2.30 we place a bound on the probability that a particular ball remains after $O(\log\log n)$ rounds.

Lemma 2.30: *For any ball $x \in [n]$, the probability that x remains after $O(\log\log n)$ rounds of* **Alg2**$(n,3,1)$ *is at most $4/n^{9/8}$ for n sufficiently large.*

Proof: By Lemma 2.29, after $j = O(\log\log n)$ rounds, $s_j \leq n^{5/8}$ wvhp. By Part 4 of Lemma 2.15, the probability that x remains after round j is at most $4n^{15/8}/n^3$ for n sufficiently large. Since $4n^{15/8}/n^3 = 4/n^{9/8}$, the desired claim follows. ∎

The following theorem follows easily from Lemma 2.30.

Theorem 2.4: **Alg2**$(n,2,2)$ *terminates in $O(\log\log n)$ rounds whp.* ∎

Corollary 2.4.1: *For $\ell \geq 2$,* **Alg2**$(n,\ell,2)$ *terminates in $O(\log\log n)$ rounds whp.* ∎

We now establish a lower bound on the number of rounds taken by **Alg2**$(n,\ell,2)$ before termination. We first place a upper bound on w'_i that complements the lower bound of Lemma 2.28. The proof of the following lemma follows the lines of Lemma 2.23 and is omitted here.

Lemma 2.31: *In* $\mathbf{Alg2}(n, \ell, 2)$ *for all* $i > 0$*, if* s'_{i-1} *and* n *are sufficiently large, then* $w'_i \leq p^{i-1}$*, where* $p > 1$ *satisfies the following inequality:*

$$p^k - \log_r 12 - 2 \sum_{0 \leq j < k} p^j \geq 0 \text{ for all } k \leq \ell - 1. \tag{2.12}$$

□

Theorem 2.5: *For all* $\ell \geq 1$*,* $\mathbf{Alg2}(n, \ell, 2)$ *terminates in* $\Omega(\log \log n)$ *rounds wvhp.*

Proof: One solution to Equation 2.12 is $p = \log_r 12 = O(1)$. Therefore, by Lemma 2.28, $w'_i \leq p^{i-1}$ for all $i > 0$. After $k = \lfloor \log_p((\log_r n)/6) \rfloor$, rounds $w'_k \leq (\log_r n)/6$, and $s'_k \geq n^{5/6}$. For n sufficiently large $n^{5/6} \geq 4n^{4/5} \log^3 n$. Therefore, by Lemma 2.24, $t_k \geq \delta^{4c^k+1} t'_k \geq \delta^{4\alpha^k+1} (s'_k)^3 / 12n^2 > 1$ wvhp for n sufficiently large. (Here α is the positive root of $\alpha^2 = 4\alpha + 13$. Note that $\delta^{4\alpha^k+1} > 12\sqrt{n}$ for n sufficiently large.) This shows that $\mathbf{Alg2}(n, 2, 2)$ executes at least $\lfloor \log_p((\log_r n)/6) \rfloor \geq \lfloor \log_p((\log_9 n)/6) \rfloor = \Omega(\log \log n)$ rounds wvhp before termination. □

2.5 Limited Independence

In this section we analyze the 1 out of ℓ protocol when the ℓ hash functions are chosen from a k-wise independent family of hash functions. We show that for any c-collision crossbar, the probability that a particular memory request remains after r rounds of the k-wise independent 1 out of ℓ protocol is close to that of the fully independent protocol for $r = O(\log \log n)$, even when $k \ll n$. Importing the results in Lemmas 2.22 and 2.30, we obtain the following main theorems.

Theorem 2.6: *For integers* $\ell \geq 3$ *and* $c \geq 1$*, the* 1 *out of* ℓ *problem is solved on a* c*-collision crossbar in* $O(\log \log n)$ *rounds whp, when the* ℓ *hash functions are chosen independently and uniformly at random from a* k*-wise independent family of hash functions for any* $k = \Omega(\log^\alpha n)$*, where* α *is a real constant chosen sufficiently large.* □

Theorem 2.7: *For integers $\ell \geq 2$ and $c \geq 2$, the 1 out of ℓ problem is solved on a c-collision crossbar in $O(\log\log n)$ rounds whp, when the ℓ hash functions are chosen independently and uniformly at random from a k-wise independent family of hash functions for any $k = \Omega(\log^{\alpha} n)$ where α is a real constant chosen sufficiently large.* □

Let $\mathcal{F}^k_{m,n}$ denote a k-wise independent family of functions from $[m]$ to $[n]$. That is, for $\{x_i : i \in [j]\} \subseteq [m]$, $y_0, \ldots, y_{\ell-1} \in [n]^j$, $j \in [k+1]$, it holds that if h is drawn uniformly at random from $\mathcal{F}^k_{m,n}$, then

$$\Pr[h(x_i) = y_i \text{ for all } i \text{ in } [j]] = 1/n^j.$$

If $k \leq \sqrt{n}$, $\mathcal{F}^k_{m,n}$ can be constructed as in [75] using the families $\overline{H}_{n^d,n}$ and H^1_{m,n^d} defined in [35] and [107] respectively. (Here d is an appropriate constant.) A hash function h chosen uniformly at random from $\mathcal{F}^k_{m,n}$ is defined as $r \circ s$, where r and s are chosen uniformly at random from $\overline{H}_{n^d,n}$ and H^1_{m,n^d} respectively. Both r and s can be evaluated in constant time [107, 35], and hence the same is true of h.

In order to analyze the 1 out of ℓ protocol, we restrict our attention to the atmost n memory requests of the processors. The hash functions with the domain restricted to this set of requests can be viewed as mapping $m \leq n$ memory locations into n memory modules k-wise independently. First, we establish a few simple properties of k-wise independent hash functions.

Lemma 2.32: *Let k, m, and n be integers such that $0 < k \leq m \leq n$. Let h be drawn uniformly at random from $\mathcal{F}^k_{m,n}$. For any $A \subseteq [n]$, $|A| \leq (k-1)/e^2$, we have*

$$\Pr[h^{-1}(A) = \emptyset] \leq (1 - |A|/n)^m (1 + e^{-(k-1)/3}).$$

Proof: If k is even, let $k' = k$; otherwise, let $k' = k - 1$. By inclusion-exclusion we have:

$$\begin{aligned}
\Pr[h^{-1}(A) = \emptyset] &= 1 + \sum_{i=1}^{m} \sum_{0 \leq x_0 < \ldots < x_{i-1} < m} (-1)^i \Pr[h(x_0), \ldots, h(x_{i-1}) \in A] \\
&\leq 1 + \sum_{i=1}^{k'} \sum_{0 \leq x_0 < \ldots < x_{i-1} < m} (-1)^i \Pr[h(x_0), \ldots, h(x_{i-1}) \in A]
\end{aligned}$$

$$
\begin{aligned}
&= 1 + \sum_{i=1}^{k'} \sum_{0 \le x_0 < \ldots < x_{i-1} < m} (-1)^i (|A|/n)^i \\
&= 1 + \left(\sum_{i=1}^{k'-1} \sum_{0 \le x_0 < \ldots < x_{i-1} < m} (-1)^i (|A|/n)^i \right) + \binom{m}{k'} (|A|/n)^{k'} \\
&\le 1 + \left(\sum_{i=1}^{m} \sum_{0 \le x_0 < \ldots < x_{i-1} < m} (-1)^i (|A|/n)^i \right) + \binom{m}{k'} (|A|/n)^{k'} \\
&\le (1 - |A|/n)^m + \binom{m}{k'} (|A|/n)^{k'} \\
&\le (1 - |A|/n)^m (1 + (em/k')^{k'} (|A|/n)^{k'} e^{2m|A|/n}) \\
&\le (1 - |A|/n)^m (1 + (e|A|/k')^{k'} e^{2m|A|/n}) \\
&\le (1 - |A|/n)^m (1 + e^{-k'} e^{2|A|}) \\
&\le (1 - |A|/n)^m (1 + e^{-k'/3}).
\end{aligned}
$$

(In the seventh equation we use the inequalities $1 - x \ge e^{-2x}$ for $0 \le x \le 1/2$ and $|A| \le k'/e^2 \le n/2$. The last equation follows since $|A| \le k'/e^2$.) □

Lemma 2.33: *Let k, m, and n be integers such that $0 < k \le m \le n$. Let h be drawn uniformly at random from $\mathcal{F}^k_{m,n}$. Let $B \subseteq [n]$ satisfy $|B| \le k/\beta$, where real $\beta > 0$. If $S = h^{-1}(B)$, then $\Pr[|S| \ge \beta|B|] \le (e/\beta)^{\beta|B|}$.*

Proof: By the definition of S, $\Pr[|S| \ge \beta|B|]$ is the probability that there exists a set $T \subseteq [m]$, $|T| = \beta|B|$, such that $h(T) \subseteq B$. Since $\beta|B| \le k$ and h is chosen uniformly from a k-wise independent family of hash functions, the desired probability is at most $\binom{m}{\beta|B|}(|B|/n)^{\beta|B|} \le (me/(\beta n))^{\beta|B|} \le (e/\beta)^{\beta|B|}$. □

Corollary 2.33.1: *Let k, m, n, and p be integers such that $0 \le p < k \le m \le n$. Let h be drawn uniformly at random from $\mathcal{F}^k_{m,n}$. For i in $[p]$, let $X = \{x_i : i \in [p]\} \subseteq [m]$ and $y = (y_0, \ldots, y_{p-1}) \in [n]^p$. Let E be the event that for all i in $[p]$, $h(x_i) = y_i$. Let $A \subseteq [n]$, $|A| \le min\{p, (k-p-1)/e^2\}$, and E' be the event that for all $x \notin X$, $h(x) \notin A$. Let $B \subseteq [n]$ satisfy $|B| \le (k-p-1)/\beta$, where real $\beta \ge 0$. If $S = h^{-1}(B)$, then $\Pr[|S| \ge \beta|B| + p \mid E \cap E'] \le 2e^{2|A|}(e/\beta)^{\beta|B|}$.*

Proof: Let Y denote $[m] \setminus X$. Thus, $g = h^Y \mid E$ is drawn uniformly from a $(k-p)$-wise independent family of functions from Y to $[n]$. The event $E' \mid E$ is equivalent to the event that $g^{-1}(A) = \emptyset$. If $k-p$ is odd, let $k' = k-p$; otherwise, let $k' = k-p-1$. By inclusion-exclusion we have

$$\begin{aligned}
\Pr[E' \mid E] &\geq (1-|A|/n)^{m-p} - \binom{m-p}{k'}(|A|/n)^{k'} \\
&\geq e^{-2|A|(m-p)/n} - (e|A|/(k'))^{k'} \\
&\geq e^{-2|A|} - (2e)^{-2|A|} \\
&\geq e^{-2|A|}/2.
\end{aligned}$$

(For the second equation we use the inequality $(1-|A|/n) \geq e^{-2|A|/n}$, since $|A| \leq n/2$. The third equation follows from the inequality $k' \geq 2e^2|A| \geq 2|A|$.)

$$\begin{aligned}
\Pr[(|S| \geq \beta|B|+p) \mid E \cap E'] &= \frac{\Pr[((|S| \geq \beta|B|+p) \cap E') \mid E]}{\Pr[E' \mid E]} \\
&\leq \frac{\Pr[(|S| \geq \beta|B|+p) \mid E]}{\Pr[E' \mid E]} \\
&\leq \frac{\Pr[g^{-1}(B) \geq \beta|B|]}{\Pr[E']} \\
&\leq 2e^{2|A|}(e/\beta)^{\beta|B|}.
\end{aligned}$$

(For the last equation we invoke Lemma 2.33 substituting $(m-p, n, k-p, \beta, B, S)$ for (m, n, k, α, B, S).) $\square$

For the rest of this section, we fix integers $\ell, c \geq 1$, and analyze the 1 out of ℓ protocol on the c-collision crossbar. Let $\vec{h} = (h_0, \ldots, h_{\ell-1})$ represent a tuple of ℓ hash functions, where $h_i : [m] \to [n]$ for all i in $[\ell]$. For $x \in [m]$, let $AFFECT_i(\vec{h}, x)$ denote the set of memory requests that could affect the success of request x in round j for all j in $[i+1]$. Formally, we define

$$AFFECT_i(\vec{h}, x) = \begin{cases} \{x\} & \text{if } i = -1, \\ \{z \in [m] : h_{i \bmod \ell}(z) = h_{i \bmod \ell}(y) & \\ \quad \text{for some } y \in AFFECT_{i-1}(\vec{h}, x)\} & \text{otherwise.} \end{cases}$$

Lemma 2.34: *Let k, m, and n be integers such that $0 \le k \le m \le n$. Let $\vec{h} = (h_0, \ldots, h_{\ell-1})$ denote ℓ hash functions chosen independently and uniformly at random from $\mathcal{F}^k_{m,n}$, For any $x \in [m]$ and $i \ge 0$, if $k \ge max\{4\log^2 n, 10|AFFECT_{i-1}(\vec{h}, x)|\}$, then $|AFFECT_i(\vec{h}, x)| \le max\{4\log^2 n, 10|AFFECT_{i-1}(\vec{h}, x)|\}$ wvhp.*

Proof: In the following we use A_i as a shorthand for $AFFECT_i(\vec{h}, x)$. Fix i and let $j = i \bmod \ell$. Let $A_{i-1} = \{x_0, \ldots, x_{p-1}\}$, where $p \in [m+1]$. If $i \ge \ell - 1$, let $A = A_{i-1} \setminus A_{i-\ell}$; otherwise, let $A = A_{i-1}$. Let $B = h_j(A_{i-\ell})$, $C = h_j(A)$, and $S = A_i \setminus A_{i-1}$. Thus $S \subseteq h_j^{-1}(C)$. Fix $y = (y_0, \ldots, y_{p-1}) \in [n]^p$ and let E be the event that $(h_j(x_0), \ldots, h_j(x_{p-1})) = y$. Let E' be the event that for all $x \notin A_{i-1}$ $h_j(x) \notin C$. Set $\beta = \max\{(e^2 p)/|C|, (\log^2 n)/|C|\}$. We now apply Corollary 2.33.1, substituting $(k, m, n, h_j, p, X, y, B, C, S, E, E', \beta)$ for $(k, m, n, h, p, X, y, A, B, S, E, E', \beta)$, to obtain $|S| \le \beta|C| + p$ with probability at least $1 - 2e^{2|B|}(e/\beta)^{\beta|C|}$. Since $\beta \ge e^2 p/|C| \ge e^2$, we have

$$
\begin{aligned}
2e^{2|B|}(e/\beta)^{\beta|C|} &\le 2e^{2|B| - \beta|C|} \\
&\le 2e^{-\beta|C|/2} \\
&\le 2e^{-(\log^2 n)/2}.
\end{aligned}
$$

(For the second equation we use the inequality $2|B| \le 2p \le 2\beta|C|/e^2 \le \beta|C|/2$. The last equation follows from the definition of β.) Thus,

$$
|A_i| \le |A_{i-1}| + |S| \le \max\{e^2 p, 2\log^2 n\} + 2p \le \max\{4\log^2 n, 10|A_{i-1}|\}
$$

wvhp. ∎

For $r \ge 0$, $\vec{h} = (h_0, \ldots, h_{\ell-1})$, $h_i : [m] \to [n]$ for all i in $[\ell]$, and $x \in [m]$, define $ASSIGN_r(\vec{h}, x)$ as $\{(x', h_0(x'), \ldots, h_{\ell-1}(x')) : x' \in AFFECT_r(\vec{h}, x)\}$. We note that $ASSIGN_r(\vec{h}, x)$ completely determines whether x succeeds within r rounds under $\vec{h}$. Any element in the set $[m] \times [n]^\ell$ of $(\ell+1)$-dimensional vectors is referred to as an *assignment*.

In the following, let $\Pr_k[EVENT(\vec{h})]$ denote the probability of $EVENT(\vec{h})$ when each hash function in $\vec{h}$ is chosen independently and uniformly from $\mathcal{F}^k_{m,n}$.

Lemma 2.35: *Let k, m, n, and p be integers such that $0 \leq k \leq m \leq n$ and $0 \leq p \leq (k-1)/(e^2+1)$. Let x_i, for all i in $[p]$, be arbitrary distinct integers from $[m]$ and $y_{i,j}$, for all i in $[p]$ and all j in $[\ell]$, be arbitrary integers from $[n]$. Let $A = \{(x_i, y_{i,0}, \ldots, y_{i,\ell-1}) : i \in [p]\}$. For arbitrary $x \in [m]$ and integer $r \geq 0$, we have*

$$\Pr_k[ASSIGN_r(\vec{h}, x) = A] \leq \Pr_m[ASSIGN_r(\vec{h}, x) = A](1 + e^{-(k-p)/3})^\ell.$$

Proof: Let E be the event that $h_j(x_i) = y_{i,j}$ for all i in $[p]$ and all j in $[\ell]$. Let $X = \{x_i : i \in [p]\}$ and $Y_j = \{y_{i,j} : i \in [p]\}$, $j \in [\ell]$. (Note that $|Y_j| \leq p$ for j in $[\ell]$.) Let E' be the event that $AFFECT_r(\vec{h}, x) = X$. Thus $ASSIGN_r(\vec{h}, x) = A$ if and only if E and E' occur.

We now consider E' under the assumption that E occurs. Let $\vec{g}$ denote the tuple $(h_0^X, \ldots, h_{\ell-1}^X)$. Let E'_0 be the event that $AFFECT_r(\vec{g}, x) = X$. Let E'_1 be the event that for j in $[\ell]$, $h_j^{-1}(B_j)$ is a subset of X, where for all j in $[\ell]$, B_j is determined as follows: If $r < j$ then $B_j = \emptyset$; otherwise, we consider two cases. If $r \bmod \ell \leq j$, then $B_j = h_j^X(AFFECT_{(\lfloor r/\ell \rfloor - 1)\ell + j}(\vec{g}, x))$; otherwise, $B_j = h_j^X(AFFECT_{\lfloor r/\ell \rfloor \ell + j}(\vec{g}, x))$. We now show that given E, E' is equivalent to $E'_0 \cap E'_1$. First, by the definition of $AFFECT$, E' implies E'_0 and E' implies E'_1. Second, event E'_0 implies that $AFFECT_r(\vec{h}, x) \supseteq X$. Since the domain of $\vec{g}$ is X, by the definitions of B_j and $AFFECT$, if $h_j^{-1}(B_j) \subseteq X$ for all j, then the calculation of $AFFECT_r(\vec{h}, x)$ will be identical to that of the calculation of $AFFECT_r(\vec{g}, x)$, i.e., $AFFECT_r(\vec{h}, x) = AFFECT_r(\vec{g}, x)$. Therefore, $E'_0 \cap E'_1$ implies E'.

Since the occurrence of event E completely determines $\vec{g}$, and hence whether E'_0 occurs, it follows that $\Pr_k[E'_0 \mid E] = \Pr_m[E'_0 \mid E]$. It remains to bound $\Pr_k[E'_1 \mid E \cap E'_0]$, which is the same as $\Pr_k[E'_1 \mid E]$.

For any $p \leq q \leq m$, if h_j is drawn from a q-wise independent family of hash functions from $[m]$ to $[n]$, then $f_j = h_j^{[m] \setminus X} \mid E$ is drawn from a $(q-p)$-wise independent family of hash functions from $[m] \setminus X$ to $[n]$. We invoke Lemma 2.32, substituting $(k-p, m-p, n, f_j, B_j)$ for (k, m, n, h, A), to obtain that for all j in $[\ell]$:

$$\Pr_k[h_j^{-1}(B_j) \subseteq X \mid E] \quad = \quad \Pr_k[f_j^{-1}(B_j) = \emptyset]$$

$$\begin{aligned} &\leq \Pr_m[f_j^{-1}(B_j) = \emptyset](1 + e^{-(k-p)/3}) \\ &= \Pr_m[h_j^{-1}(B_j) \subseteq X \mid E](1 + e^{-(k-p)/3}). \end{aligned}$$

Since $\Pr_q[E' \mid E]$ is at most $\prod_{0 \leq j < \ell} \Pr_q[h_j^{-1}(B_j) \subseteq X \mid E]$ for q in $[m+1]$, we have

$$\begin{aligned} \Pr_k[E \cap E'] &= \Pr_k[E]\Pr_k[E'_0 \mid E]\Pr_k[E'_1 \mid E] \\ &= \Pr_k[E]\Pr_k[E'_0 \mid E] \prod_{0 \leq j < \ell} \Pr_k[h_j^{-1}(B_j) \subseteq X \mid E] \\ &= \Pr_m[E]\Pr_m[E'_0 \mid E] \prod_{0 \leq j < \ell} \Pr_k[h_j^{-1}(B_j) \subseteq X \mid E] \\ &\leq \Pr_m[E]\Pr_m[E'_0 \mid E] \prod_{0 \leq j < \ell} \Pr_m[h_j^{-1}(B_j) \subseteq X \mid E](1 + e^{-(k-p)/3})^\ell \\ &= \Pr_m[E]\Pr_m[E'_0 \mid E]\Pr_m[E'_1 \mid E](1 + e^{-(k-p)/3})^\ell \\ &= \Pr_m[E \cap E'](1 + e^{-(k-p)/3})^\ell. \end{aligned}$$

$\square$

Lemma 2.36: *Let k, m, and n be integers such that $0 \leq k < m \leq n$. For any real $\gamma \geq 0$, $x \in [m]$ and integer $r \leq \gamma \log\log n$, if $k \geq 8 \log^{4\gamma+2} n$, then*

$\Pr_k[x$ *remains after* r *rounds under* $\vec{h}] \leq \Pr_m[x$ *remains after* r *rounds under* $\vec{h}] + 1/n^2$

for n sufficiently large.

Proof: Let $\mathcal{A}$ be $\{ASSIGN_r(\vec{h}, x) : \vec{h} \in \mathcal{F}_{m,n}$ and x remains after r rounds under $\vec{h}\}$. By Lemma 2.34, $|ASSIGN_r(\vec{h}, x)| \leq 4(\log^2 n)10^r\} \leq 4 \log^{4\gamma+2} n$ wvhp. By Lemma 2.35, for any assignment A such that $|A| \leq 4 \log^{4\gamma+2} n$,

$$\begin{aligned} \Pr_k[ASSIGN_r(\vec{h}, x) = A] &\leq \Pr_m[ASSIGN_r(\vec{h}, x) = A](1 + e^{-(k-|A|)/3})^\ell \\ &\leq \Pr_m[ASSIGN_r(\vec{h}, x) = A](1 + 1/n^3), \end{aligned}$$

for n sufficiently large. (Here we use the inequality $k \geq 8 \log^{4\gamma+2} n \geq 2|A|$.) Thus,

$\Pr_k[x$ remains after r rounds under $\vec{h}]$

$$
\begin{aligned}
&\leq \Pr_k[ASSIGN_r(\vec{h},x) \in \mathcal{A}] \\
&\leq \Pr_k[(ASSIGN_r(\vec{h},x) \in \mathcal{A}) \text{ and } |ASSIGN_r(\vec{h},x)| \leq 4\log^{4\gamma+2} n] + 1/n^3 \\
&\leq \sum_{\substack{A\in\mathcal{A} \\ |A|\leq 4\log^{4\gamma+2} n}} \Pr_k[ASSIGN_r(\vec{h},x) = A] + 1/n^3 \\
&\leq \sum_{\substack{A\in\mathcal{A} \\ |A|\leq 4\log^{4\gamma+2} n}} \Pr_m[ASSIGN_r(\vec{h},x) = A](1+1/n^3) + 1/n^3 \\
&\leq \Pr_m[ASSIGN_r(\vec{h},x) \in \mathcal{A}](1+1/n^3) + 1/n^3 \\
&\leq \Pr_m[x \text{ remains after } r \text{ rounds under } \vec{h}] + 1/n^2,
\end{aligned}
$$

for n sufficiently large. ∎

By Lemma 2.22, for any $x \in [n]$, $\Pr_n[x$ remains after $O(\log\log n)$ rounds$]$ is at most $2/n^{6/5}$ in the 1 out of 3 protocol on the 1-collision crossbar. Similarly, for the 1 out of 2 protocol on the 2-collision crossbar, Lemma 2.30 implies that $4/n^{9/8}$ is a lower bound on $\Pr_n[x$ remains after $O(\log\log n)$ rounds$]$ for any $x \in [n]$. We now apply Lemma 2.36 to establish Theorems 2.6 and 2.7.

2.6 The Emulation Protocols

Alg1 can be generalized to apply to any a out of b problem by changing the routines *RandomSubbag* and *PrunedBag* appropriately; after each step, we need to keep track of how many successes each processor has had, and only those processors with fewer than a successes participate. In the following discussion, we refer to this protocol as the *generic* protocol. For given a and b, the analysis of the generic protocol can be done using the approach of Subsection 2.4.3, but involves more complicated calculations and recurrences. A different analysis of this protocol for the 2 out of 3 case is given in [47], where an $O(\log\log n)$ upper bound is shown when the collision factor is greater than 3. In this section, we present a simple variant of the generic protocol that solves any a out of b problem on a 2-collision crossbar, and hence on a 2-arbitrary crossbar as well, in $O(\log\log n)$ time whp.

In particular, we can solve any a out of $a+1$ problem by running $\mathbf{Alg1}(n, 2, 1)$ with $\binom{a+1}{2}$ different hash-function pairs. Since each run fails with a polynomially small probability and there are only a constant number of runs, the entire algorithm succeeds whp. For instance, in the case of 2 out of 3, we simply perform 3 runs of **Alg1**. At first glance, it may appear that this revised protocol is only of interest because it is simpler to analyze. Actually, the new protocol is competitive with the generic one for small a and is much faster for large a. Comparing it for the 2 out of 3 problem, we first note that since each of the 3 runs use 2 hash functions only while the generic protocol uses 3, the revised protocol will be at most twice as slow as the generic one. Moreover, the 1 out of 2 problem is clearly a simpler problem than the 2 out of 3 problem. So each run will involve a fewer rounds than in the generic algorithm. For large a, this phenomenon is pronounced. For a generic a out of $a+1$ protocol to make "progress", a number of processors must have a large number of successes. But at the outset, the fraction of processors that have succeeded on $d \leq a$ hash functions decreases exponentially with d. Therefore, while the revised protocol experiences only a quadratic slowdown in running time, the generic protocol will suffer an exponential increase in running time with increasing a. (We remark here that our notion of the generic protocol does not include the protocol studied in [94] that is shown to have a running time that is polynomial in a when a, b, and c are suitably chosen.)

The basic idea outlined above can be used to solve any a out of b problem by choosing any $a+1$ hash functions and solving the corresponding a out of $a+1$ problem.

Theorem 2.8: *For integer constants a and b with $1 \leq a < b$, the corresponding a out of b problem can be solved on a 2-collision or a 2-arbitrary crossbar in $O(\log\log n)$ time whp.* □

The above generalizations can be made for the 1-collision and the 1-arbitrary crossbars as well. Since **Alg1** solves the 1 out of 3 problem on a 1-collision crossbar in $O(\log\log n)$ time whp, any a out of $a+2$ problem can be solved in the same asymptotic time bound by running $\mathbf{Alg1}(n, 3, 1)$ on $\binom{a+2}{3}$ different triples of hash functions.

Theorem 2.9: *For integer constants a and b with $1 \leq a < b - 1$, the corresponding a out of b problem can be solved on a 1-collision or a 1-arbitrary crossbar in $O(\log \log n)$ time whhp.* ∎

2.7 Concluding Remarks

In this chapter, we have analyzed a class of simple local protocols for emulating an EREW PRAM on a c-arbitrary crossbar. As shown in Section 2.6, the delay associated with the emulations is $O(\log \log n)$ whp. One way to reduce delay is to introduce parallel slackness by emulating a non-constant number of EREW PRAM processors on a node of the crossbar. The concept of parallel slackness has been used by [47, 63, 75] to obtain *work-optimal* emulations. A work-optimal emulation with delay $d(n)$ is a protocol that emulates a $d(n)$-processor EREW PRAM on an n-processor crossbar in $O(d(n))$ time. It would be interesting to see if the techniques developed in this chapter can be applied to obtain tight analyses of simple work-optimal emulations.

An alternative approach to reduce delay is to allow the nodes of the crossbar to execute asynchronously. In other words, once a node has successfully accessed its request, it need not wait for the successful completion of the access requests of other nodes before proceeding to its next access request. Such an approach not only reduces delay, but is also preferable since no explicit synchronization is required. We adopt this dynamic approach in the following chapter, where we build on the techniques of hashing and replication used here to develop a protocol that provides efficient access to shared objects in the presence of faults and concurrency.

Our main technical contribution in this chapter is the derivation of sharp threshold phenomena associated with certain random allocation experiments. Several recent papers have studied similar processes that arise in dynamic resource allocation and parallel load balancing [2, 25, 108].

Chapter 3

Fast Fault-Tolerant Concurrent Access to Shared Objects

3.1 Introduction

In this chapter, we design and analyze a simple local protocol for providing fast concurrent access to shared objects in a faulty distributed network. We model the network as a faulty $O(\log n)$-arbitrary crossbar. In particular, our model consists of n nodes communicating via point-to-point messages, subject to the following constraints: In a single step, (i) a node can only send or receive $O(\log n)$ words[1], (ii) a constant fraction of the nodes may be "down" (i.e., unable to communicate with any other nodes), and (iii) each "up" node may be unable to directly communicate (i.e., via a single point-to-point message) with a constant fraction of the other "up" nodes. See Section 3.2 for a precise definition of our model of computation.

How can we provide efficient concurrent access to a given popular object A in a network that supports only partially reliable point-to-point communication? In a conventional distributed file system, a single "server" process (residing on a particular physical node) is assigned the responsibility for storing the object A, and any "client"

[1]Throughout this chapter, we use the term "word" to refer to an $O(\log n)$-bit string.

process wishing to read A sends a message to this server; the server then responds with a message containing a copy of the object A. Unfortunately, this scheme suffers from both low fault-tolerance (if a given client cannot connect to the server due to a network fault, an event that occurs with constant probability in our model, then that client cannot access the object) and high latency (since A is assumed to be a popular object, a long time is needed for the server to sequentially service each of the incoming requests for A).

Thus, to obtain either fault-tolerance or fast concurrent access we are led to consider schemes in which each object is replicated across a number of different nodes. In Chapter 2, we observed that a constant number of copies suffices to provide efficient access to shared objects under the assumption that there are no faults and no concurrent accesses. Fault-tolerance considerations alone, however, would seem to imply that each object should be replicated $\Omega(\log n)$ times if we wish to guarantee access with a failure probability that is polynomially small in n, the number of nodes in the network, since each node can fail with constant probability. Unfortunately, this results in an $\Omega(\log n)$-fold increase in the space needed to store each object. The theory of erasure codes, however, provides a convenient method for achieving fault-tolerance while paying only a constant factor space penalty. For example, using Rabin's Information Dispersal Algorithm [104] (IDA), for any $k > m$, a given b-bit string can be encoded as a set of k (b/m)-bit strings of length m, with the property that any m of the (b/m)-bit strings suffice to reconstruct the original b-bit string. Thus, IDA can be used to obtain fault-tolerance with only a constant factor space penalty by setting m to $\Theta(\log n)$ and k to $\Theta(m)$, e.g., $k = 2m$. This powerful technique is used by Aumann et al. [17] as part of an efficient scheme for emulating large-grained PRAM programs on an asynchronous parallel machine. In our protocol, we use the same technique to store the "primary" copy of each object.

Of course, the IDA technique alone is not sufficient to guarantee fast (e.g., $O(\log n)$ time) concurrent access to an object that is extremely popular. For example, suppose that the popularity of some object is $\sqrt{n}$, and that IDA has been used to encode the object in $\Theta(\log n)$ "fragments", each of which is stored in a separate node.

Assuming point-to-point communication, and assuming that a single node cannot send or receive more than $O(\log n)$ messages in a single time step, it is clear that without further replication (either of the individual fragments, or of the object as a whole), $\Omega(\sqrt{n}/\log n)$ steps are needed in order to service all $\sqrt{n}$ requests.

A central part of our protocol is a mechanism for on-line replication. At a high level, the replication mechanism provides fast concurrent access by enforcing the following two invariants. *Invariant 1:* While the popularity of a given object exceeds the number of "server copies" (i.e., the number of server processes holding a copy of the object), the number of server copies increases geometrically. *Invariant 2:* When the popularity of a given object does not exceed the number of server copies by more than a constant factor, each outstanding request is independently serviced with constant probability at the current step. Thus, if the popularity of the object does not change during subsequent steps, each of the outstanding requests is serviced in $O(1)$ expected steps, and in $O(\log n)$ steps wvhp[1].

3.1.1 Overview of the Results

The design of our protocol is presented in Sections 3.3, 3.4, and 3.7. Section 3.3 gives an informal overview of our protocol. Section 3.4 contains a formal definition of our protocol for read-only objects. Section 3.7 discusses write operations.

In order to establish the fast performance of our protocol, we consider two natural models for dynamic access patterns.

- Our first model is the *fixed* model in which there is a fixed probability distribution from which each access request is independently drawn at all times. We show that our protocol reaches a *nice* state in $O(\log n)$ steps, after which: (i) in each step, a constant fraction of all requests are satisfied wvhp, and (ii) each request is satisfied in expected $O(1)$ steps and $O(\log n)$ steps wvhp.

[1]Recall that the term wvhp, which is defined in Section 1.3, means with probability $1-n^{-c}$, where c is a constant that can be set arbitrarily large by appropriately adjusting other constants defined within the relevant context.

- Our second model is the *dynamic* model that allows constrained fluctuations in the access pattern over time. More precisely, in the dynamic model the number of new access requests for any object in any step may change arbitrarily subject to the constraint that it does not grow beyond a constant factor times the maximum number of accesses to the same object in the previous $O(\log n)$ steps. For the dynamic model, we show that the protocol is always in a nice state.

While the result for the fixed model establishes the rapid convergence of our protocol to a nice state in the presence of a fixed global access pattern, the result for the dynamic model shows that once a nice state is reached, the protocol tolerates "non-volatile" changes in the access pattern. In Section 3.5, we formally define the two access pattern models and state the performance bounds of the protocol. Section 3.6 contains an analysis of the protocol, in which we first establish Invariants 1 and 2 under suitable assumptions, and then, by making use of the invariants, establish the bounds for the fixed and dynamic models.

3.1.2 Related Work

Most of the details of our protocol are concerned with ensuring fast access to popular objects. A variety of other well-known methods have been used for solving this problem, including broadcast, combining [105], and multicast [45]. However, the class of architectures that support the efficient implementation of these methods is restricted. For example, a single-bus network can efficiently support broadcast, which enables an arbitrary subset of the processors to obtain copies of a single object at the same time. On the other hand, the cost of implementing broadcast in a distributed network with point-to-point connections is significant.

Our hashing techniques are loosely related to Valiant's hashing-based combining mechanism for simulating CRCW PRAM algorithms on parallel computers [114]. In other related work, Gibbons, Matias, and Ramachandran [55] adopt a different approach to account for contention in parallel algorithms. They introduce the QRQW PRAM model, which permits concurrent reading and writing but at a cost proportional to the number of readers/writers to a memory location in a given step. The focus of our algorithm design and analysis is different. While [55] and [114] are primarily concerned with the problem of PRAM emulation, we have optimized our protocol to obtain fast performance (e.g., expected $O(1)$ time) on a more restricted class of access patterns.

The particular assumptions of our model and the design choices of our protocol are largely influenced by the characteristics and requirements of wide-area network file systems. There is a growing need for efficient protocols to access objects across wide-area networks and several solutions have been recently proposed. While all of these solutions incorporate replication of objects, they differ on the mechanism used to determine where the copies of a given object are stored. Very broadly, existing protocols can be classified into two categories.

The first category consists of implementations where a client accesses a given object by consulting the "manager" of the object to locate a copy (examples include [31], [67], and xFS [11]). The main drawback with this approach is that the manager is usually implemented as a process running at a single node and thus constitutes a sequential bottleneck. Instead of sending each request for a given object through a central server, the protocols in the second category forward the request along a path in a tree of nodes, the root of which is the "owner" of the object (examples include [74] and Harvest [33, 36]). An advantage of this approach is that if several copies of a given object exist, then a request to the object is satisfied within a small number of forwardings along the path. A request for an object with low popularity, however, may be significantly delayed because the request may have to be forwarded through all the nodes in the relevant path to the owner of the object. Our protocol overcomes the above drawbacks by sending each request to $O(\log n)$ nodes in parallel. By choosing the $O(\log n)$ nodes judiciously, we are able to ensure that if the number of copies of an object is at least a constant fraction of the number of requests, then the expected time for accessing the object is $O(1)$.

3.2 Model of Computation

In this section we define our model of computation. We assume a synchronous network consisting of n nodes, each with its own local memory. We specify the model by characterizing: (i) communication, (ii) faults, (iii) object size, (iv) cache size, and (v) local computation.

Communication. Nodes communicate with one another by sending messages. Each message contains at least one word, and at most $O(\log n)$ words, where a "word" is defined as an $O(\log n)$-bit string.

Sending messages. The total number of words in all messages sent by a single node in one step is required to be $O(\log n)$ (even if some or all of these messages are not successfully transmitted due to faults in the network, which are discussed below).

Receiving messages. The total number of words in all messages received by a single node in one step is required to be $O(\log n)$. We place no upper bound on the total number of words in all messages destined to a single node in one step; instead, we only limit to $c_0 \log n$ the number of words in all messages successfully received by a node in one step, where c_0 is some positive constant. We assume that a worst-case adversary determines which subset of the messages of total size $c_0 \log n$ are successfully received by a given node if the $c_0 \log n$ limit on total size would otherwise be exceeded.

Message types. Our protocol makes use of a constant number of different types of messages. At times the protocol may result in, say, $O(\log n)$ messages of type α and $\Theta(\sqrt{n})$ messages of type β being sent to a particular node. In such a scenario, the adversary referred to above has the freedom to decide that none of the messages of type α get through. On the other hand, it may be important for the correctness of the protocol that the type α messages be given priority over the type β messages. One way to accomplish this is to modify the model stated above by associating a numeric priority with each message type to resolve contention among messages of different types. Since our protocol only makes use of a constant number of different message types, we could avoid introducing such priorities by modifying the protocol to ensure that only one type of message is ever sent in a single step. We prefer the former solution since it is more compatible with an asynchronous view of the protocol.

Faults. As mentioned in Section 3.1, our model of computation also allows for the possibility of faults in the network. More specifically, we assume that the network is subject to the following three classes of faults.

Random static node faults. After we have fixed our initial storage layout for the objects, we assume that a (sufficiently small) constant fraction ϕ_0 of the nodes are selected at random and marked as "dead". Such dead nodes cannot send or receive any messages throughout the course of the computation.

Dynamic node faults. An oblivious adversary selects, for each step, a (sufficiently small) constant fraction ϕ_1 of the nodes and marks them "down". Such down nodes cannot send or receive any messages in the current step.

Dynamic link faults. For each pair of up nodes (i.e., neither dead nor down) i and j in the network, an oblivious adversary determines whether communication between nodes i and j is to be allowed in step t. In each step t, each up node must be allowed the possibility of communicating with a (sufficiently large) constant fraction $(1 - \phi_2)$ of the other nodes.

With regard to the dynamic link faults, we should emphasize that the set of faulty links determined by the adversary are not provided to the non-dead nodes at execution time. The only way that a non-dead node can find out whether it is possible to communicate with some other non-dead node in step t is by attempting to send a message in step t, with the hope of subsequently receiving some form of acknowledgment in a later step. (Of course, any acknowledgment message is itself subject to possible faults.)

Object size. Each object consists of $\Theta(\log n)$ words. Note that this assumption can be enforced by simply breaking up larger data items into $\Theta(\log n)$-word pieces, and padding out smaller data items to $\Theta(\log n)$ words. The main reason for assuming a uniform object length is that it simplifies our presentation and analysis. In a practical implementation, we would modify the protocol to handle messages of varying lengths;

for larger objects, the associated optimizations can be expected to provide substantial constant factor savings in overhead per object-word accessed.

Cache size. We assume that each node of the network has a cache in which extra copies of objects are stored. In our analysis, it is convenient to assume that the capacity of each cache is $\Omega(\log n)$ objects.

Local computation. In each step, a node is allowed to perform an arbitrary amount of local computation. Although the model of computation allows an arbitrary amount of local computation in each step, our protocol does not perform any particularly complex local operations in a single step.

3.3 Overview of the Protocol

In this section we provide an informal overview of our protocol for accessing read-only objects. Our discussion is formalized in Section 3.4. See Section 3.7 for a discussion of write operations. As mentioned in Section 3.1, our protocol relies on maintaining Invariants 1 and 2.

Enforcing Invariant 1. With each object we associate a number of disjoint *blocks* of servers. The ith block contains $\Theta(2^i \log n)$ servers, $0 \leq i < \log(n/\Theta(\log n))$, so that the total number of servers in all blocks is n, the number of nodes in the network. A hash function is used to map these logical blocks of servers to the physical nodes of the network. The hash function is distributed to all nodes so that any node can rapidly compute the physical node corresponding to the jth server of the ith block of a given object. The $\Theta(\log n)$ servers of block 0 of an object are used to store the primary copy of that object, i.e., the $\Theta(\log n)$ fragments computed using IDA. Each server in the higher-numbered blocks (block 1 onwards) of an object is used to store a whole copy of that object. We choose to replicate whole copies of objects, as opposed to fragments, so that the encode-decode overhead associated with IDA can be avoided on retrieval of popular objects. This may be viewed as a minor optimization since the overhead of IDA is actually quite small [104].

In our protocol, a client process attempting to read a particular object A sends $\Theta(\log n)$ messages, one to each of the $\Theta(\log n)$ servers in block 0 of A, and $O(1)$ messages to a randomly chosen set of servers in each of the $\Theta(\log n)$ other blocks associated with A. (See Figure 3.1.) If the popularity of A is low (i.e., $O(\log n)$ where the hidden constant is sufficiently small), then wvhp a sufficiently high constant fraction of the messages sent to block 0 are successfully transmitted, and at the next step a sufficiently high number of fragments are returned to the client, allowing the client to reconstruct a copy of the desired object using IDA. (Note that a node can send $O(\log n)$ copies of a fragment in a single step, since a fragment only consists of a constant number of words.)

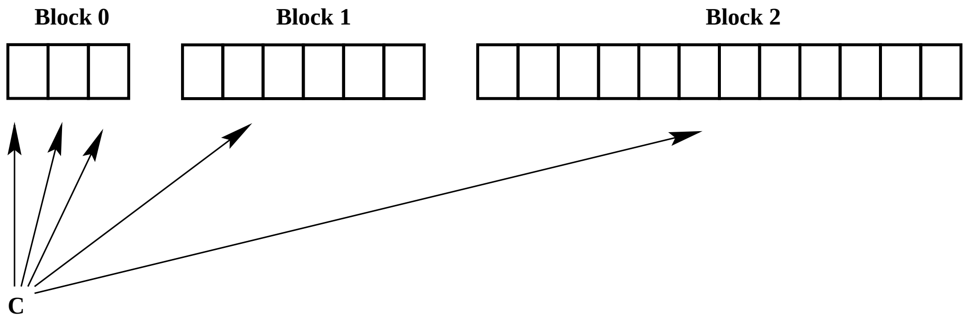


Figure 3.1: The request messages sent by a client. Client C attempting to read a particular object A sends messages to $O(\log n)$ of the n servers associated with A. Each cell shown above represents a server associated with object A. The client sends one request message to each server in block 0, and $O(1)$ messages to a randomly chosen set of servers in each of the other blocks.

If the popularity of A is high (i.e., $\Omega(\log n)$ where the hidden constant is sufficiently high), then so many clients attempt to access A that the servers in block 0 of A are "flooded" with incoming messages requesting fragments of A. As a result, most of these messages are not successfully transmitted, and few if any of the clients receive (on the next step) sufficiently many fragments to reconstruct A using IDA. On the other hand, a sufficiently high constant fraction of the servers in block 0 of A receive $\Theta(\log n)$ messages requesting a fragment of A.

One might believe that *all* of the servers in block 0 receive $\Theta(\log n)$ such messages; this is not necessarily the case, however, since some of these servers may be mapped to the same node as, for example, the servers in block 0 of one or more other popular objects, so that the messages associated with A might be "swamped out" by the messages associated with other objects. A critical part of our analysis is geared towards proving that wvhp a sufficiently high constant fraction of the nodes in block 0 of A is not the destination of more than a total of $O(\log n)$ messages associated with other objects at the current step; these are the nodes that wvhp receive $\Theta(\log n)$ requests for A.

Each server in block 0 of A that detects a high level of popularity for A at a particular step reacts by attempting to send a copy of the fragment of A that it holds to all $O(\log n)$ servers in block 1 of A. (See Figure 3.2.) Although the servers in block 1 may all be flooded with client requests for A (since the popularity of A is assumed to be high), the fragment messages sent from servers in block 0 are not swamped out by such client requests because the fragment messages are given a higher priority. (Of course, we need to argue that these fragment messages are not swamped out by same-priority fragment messages associated with other objects; this follows by essentially the same argument as was mentioned in the preceding paragraph.) As a result of the fragment messages sent from servers in block 0 (the constant fraction detecting a high popularity for A) to servers in block 1, wvhp a sufficiently high constant fraction of the servers in block 1 of A reconstruct a copy of A using IDA.

Thus, if the popularity of A is sufficiently high at time t, then at time $t+1$, a constant fraction of the servers in block 1 of A hold a copy of A wvhp. A minor variant of the above process is used to ensure that, if a sufficiently large constant fraction of the servers in block 1 hold a copy of A at time t, and if the popularity of A is $\Omega(\log n)$, then a constant fraction of the servers in block 2 hold a copy of A at time $t+1$. The idea is that a server in block 1 "detects a high popularity" for A if it receives more than a certain constant threshold number of requests for A. Rather than sending $O(\log n)$ fragments of A to servers of block 2 (as were sent from servers of block 0 to servers of block 1 earlier), each server of block 1 detecting a high popularity for A sends $O(1)$ copies of A to a randomly chosen set of servers in block 2 of A. (Note that $O(1)$ copies of A require $O(\log n)$ words.)

More generally, suppose that at time t a sufficiently high constant fraction of the servers in each of blocks 1 through i holds a copy of A, and that the popularity of A is $\Omega(2^i \log n)$, where the hidden constant is sufficiently large. Then a constant fraction of the servers in block i receive more than a certain constant threshold number of requests for A, and react by sending $O(1)$ copies of A to randomly chosen servers in block $i+1$. (See Figure 3.3.) As a result, at time $t+1$, a constant fraction of the servers in block $i+1$ of A hold a copy of A wvhp.

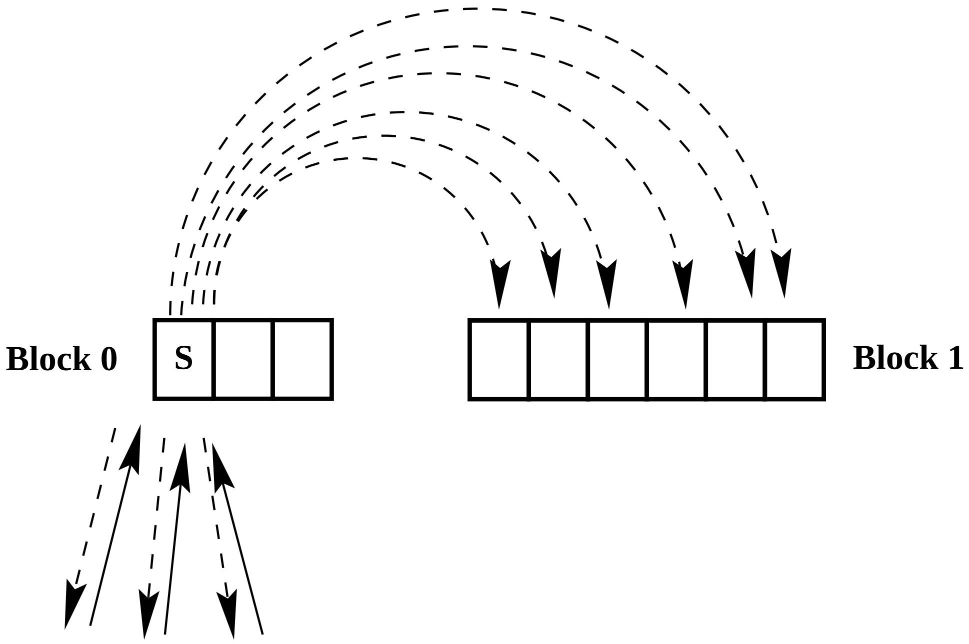


Figure 3.2: The response of a server in block 0. The unbroken arrows represent request messages from clients. The broken arrows represent responses from server S in block 0. If the number of requests to S exceeds a certain $\Omega(\log n)$ threshold, then S sends a copy of the fragment it holds to each server in block 1. This helps to enforce Invariant 1. Server S also sends a copy of the fragment it holds to each of the requesting clients. This helps to enforce Invariant 2.

Enforcing Invariant 2. The total number of requests received by a server for object fragments is $O(\log n)$ per step, simply because a node cannot receive more than $O(\log n)$ messages per step. Thus, in the following step (assuming it is not subject to a dynamic node fault), a server can respond to each such request with a copy of the desired fragment.

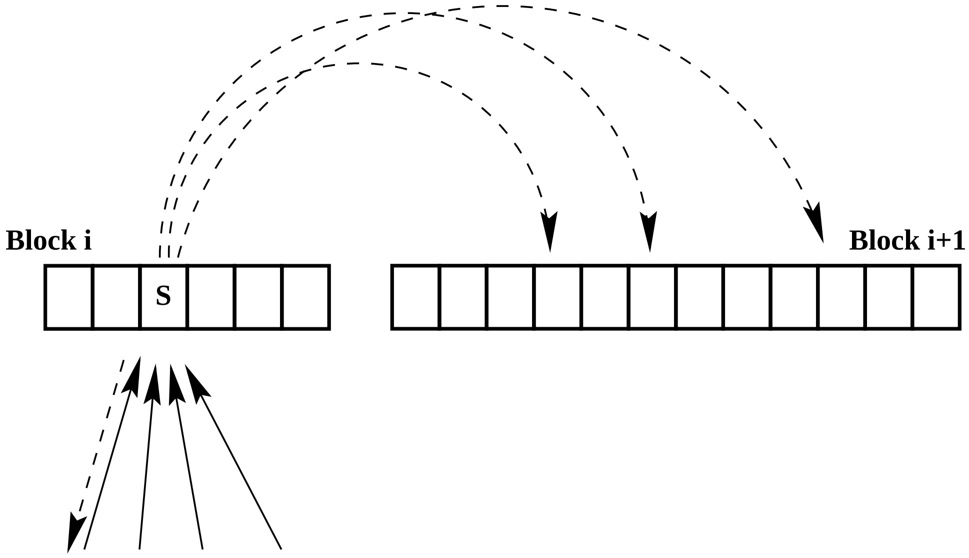


Figure 3.3: The response of a server in block i, $i > 0$. The unbroken arrows represent request messages from clients. The broken arrows represent responses from server S in block i. If S holds a copy of the object, it sends the following messages. If the number of requests to S exceeds a certain constant threshold, then S sends a constant number of copies of the object randomly chosen servers in block $i + 1$. This helps to enforce Invariant 1. Server S also sends a copy to an arbitrarily chosen constant-size subset of the requesting clients. This helps to enforce Invariant 2.

This is illustrated in Figure 3.2. (Recall that a fragment consists of a constant number of words and so the total number of words in all of these responses is $O(\log n)$.) Of course, each of these responses may or may not be received by the associated client due to the possibility of dynamic faults in the network. On the other hand, the server in block $i > 0$ may also receive as many as $O(\log n)$ requests for entire copies of objects, and since each object consists of $\Theta(\log n)$ words, only a constant number of these requests can be handled in a single step. In our protocol, the server selects a constant-size subset of the incoming requests for entire copies of objects, and responds only to this selected subset. (See Figure 3.3.)

Now suppose that the hypothesis of Invariant 2 holds, that is, the popularity of some object A is less than or equal to the number of server copies of the object. Since our mechanism for generating server copies fills in the blocks in ascending order of index, we can deduce that a block of servers of A with size within a constant fraction of the current popularity of A satisfies the following two conditions wvhp: (i) a constant fraction of the servers in the block contain a copy of A, and (ii) each client requesting a copy of A sends a constant number of messages to randomly chosen servers within the block. By a straightforward Chernoff-type argument [37], we can show that a constant fraction of the client requests for A are satisfied at the current step, establishing Invariant 2.

Cache management. Each node has a cache for holding extra object copies. When this cache becomes full, an LRU (least-recently-used) replacement policy is invoked to decide which object copy to abandon.

3.4 The Read-Only Protocol

In this section, we formally define our protocol for accessing read-only shared objects. With every object A we associate n server processes, which provide client processes access to A. Let the servers associated with A be labeled $S_i(A)$, $0 \leq i < n$. Let b equal $\log(n/c_1 \log n) + 1$, where c_1 is a constant that is specified later. We assume that $c_1 \log n$ and b are both integers. We partition the set of servers into *blocks* as follows. For each i in $[b]$, the ith block, denoted by $B_i(A)$, is the set $\{S_j(A) : (2^i - 1)c_1 \log n \leq j < (2^{i+1} - 1)c_1 \log n\}$. For each i in $[b]$, let $b_i = c_1 2^i \log n$ be the size of the ith block.

Each server associated with A is mapped to a physical node by means of a hash function h_A; the function h_A is chosen such that for any i in $[b]$, block $B_i(A)$ is mapped to a subset of $|B_i(A)|$ nodes chosen independently and uniformly at random. (Note that several servers associated with different objects may be mapped to the same node.) Using IDA [104], we encode A as a set of b_0 *fragments* such that any $b_0/4$ fragments suffice to decode A. For each i in $[b_0]$, $h_A(S_i(A))$ stores the ith fragment of A. For each integer j in $[1, b)$, and for each server $S \in B_j(A)$, $h_A(S)$ stores at most one replicated copy of the entire object. Let the cache at each node have the capacity to store the maximum number of object copies that may be received by the node in Γ rounds. Thus, the minimum cache capacity is $\Theta(\Gamma)$ objects. We assume that Γ is $\Omega(\log n)$.

We describe our access protocol in terms of the communication between the clients attempting to access a given object A and the servers associated with A. In order to simplify the presentation and analysis of the protocol, we assume that the clients send messages at even steps of the protocol and the servers send messages at odd steps of the protocol. The clients always send messages to servers; servers send messages to both clients and servers. For any nonnegative integer t, let round t denote the pair of steps $2t$ and $2t+1$ (steps are numbered from 0).

In our description of the protocol, we differentiate between several kinds of messages; these are listed in Table 3.1. In the priority-based model, any assignment of priorities that respects the following constraints can be used: (i) *frag-req* has a lower priority than each of *client-obj* and *client-frag*, and (ii) *obj-req* has a lower priority than each of *client-obj*, *client-frag*, *server-obj*, and *server-frag*, and (iii) each of *server-frag* and *server-obj* has a lower priority than each of *client-frag* and *client-obj*.

Message type	Source	Destination	Size	Contents
obj-req	client	server	$\Theta(1)$	request for object
frag-req	client	server	$\Theta(1)$	request for fragment
client-obj	server	client	$\Theta(\log n)$	copy of object
client-frag	server	client	$\Theta(1)$	copy of fragment
server-obj	server	server	$\Theta(\log n)$	copy of object
server-frag	server	server	$\Theta(1)$	copy of fragment

Table 3.1: Types of messages.

The protocol is defined in Figure 3.4, where we state the actions in round t of: (i) a client C attempting to access object A, and (ii) a server S associated with A. It is convenient to divide each step into two phases, one in which messages are sent, and the other in which messages are received. Thus, in Figure 3.4, Phase 0 (resp., Phase 2) is the "sending phase" for step $2t$ (resp., $2t+1$), while Phase 1 (resp., Phase 3) is the "receiving phase" for step $2t$ (resp., $2t+1$). In Figure 3.4, π_0, π_1, π_2, π_3, π_4, π_5, and π_6 denote positive integer constants.

The terms "send", "receive", and "attempt to send" are used in the protocol

Phase 0: *In step $2t$ clients send request messages.*

- Client. Attempt to send a *frag-req* message to each server in $B_0(A)$ and, for $0 \le i < b$, an *obj-req* message to a random server in $B_i(A)$.

(Remark: Note that each message is actually sent since the bound on the number of words that can be sent by a node is not exceeded.)

Phase 1: *Successfully transmitted Phase 0 messages are received by servers.*

- Server. Let $D(S,t)$ denote the set of clients that are the sources of *obj-req* and *frag-req* Phase 1 messages received by S.

Phase 2: *In step $2t+1$, servers holding a copy or fragment of object A respond to Phase 1 messages. Let $S \in B_i(A)$.*

- Server, $i = 0$. Attempt to send a *client-frag* message to $\min\{\pi_0 b_0, |D(S,t)|\}$ clients in $D(S,t)$, and if $D(S,t) \ge \pi_1 b_0$ then attempt to send a *server-frag* message to each server in $B_1(A)$.
- Server, $i > 0$. If $|D(S,t)| \ge \pi_2$ then attempt to send a *client-obj* message to $\min\{\pi_3, |D(S,t)|\}$ random clients in $D(S,t)$, and if $D(S,t) \ge \pi_4$ then attempt to send a *server-obj* message to π_5 random servers in $B_{i+1}(A)$.

(Remark: If the bound on the number of words a node can send in a step would be exceeded, an arbitrary subset of these messages are actually sent.)

Phase 3: *Successfully transmitted Phase 2 messages are received by clients and servers.*

- Client. If C receives a *client-obj* message or $c_2 \log n$ fragments, then the access attempt is successful. Otherwise, C attempts to access A in round $t+1$.
- Server, $i = 1$. If S receives at least $c_2 \log n$ fragments, then decode A and store it in the LRU cache; otherwise, discard the fragments received.
- Server, $i > 1$. If S receives at least π_6 *server-obj* messages, then store A in the LRU cache.

(Remarks: Note that C could receive more than one copy of A, and that S could receive a new copy of A even though S already has a copy. In a practical implementation: (i) C would stop transmission of all but one copy of A, (ii) a check would be added to ensure that a new copy is sent to S only if S does not already have a copy, and (iii) if fewer than $c_2 \log n$ fragments are received by a client or server, then these fragments would be cached and not discarded since sufficiently many additional fragments are likely to be received in the near future.)

Figure 3.4: The read-only protocol (object A, client C, server S, round t).

definition. When we say that a client/server mapped to node u *sends* a message x, we mean that u initiates the transmission of x. When we say that a client/server mapped to node u *receives* a message x, we mean that the transmission of x is successful and x is at destination u. When we say that a client/server mapped to node u *attempts to send* a message x, we mean that u sends x if x is in the subset of messages of total size at most $c_0 \log n$ that is selected for transmission from u.

3.5 Statement of the Results

In this section, we state our main performance bounds for the read-only protocol. Let $\mathcal{A}$ be a collection of m objects, labeled A_0 through A_{m-1}. For any round t, and any i in $[m]$, let $q_i(t)$, $r_i(t)$, and $s_i(t)$ denote the number of new requests generated, the number of requests remaining, and the number of requests attempted, respectively, for A_i at the start of round t. (For any nonnegative integer x, we use $[x]$ to denote the set $\{0, \ldots, x-1\}$.) Thus, for any round t, and any i in $[m]$, $s_i(t) = r_i(t) + q_i(t)$.

We measure the performance of our protocol in terms of *throughput*, *delay*, and *per-request communication*. The throughput of the protocol is the average number of access requests satisfied per round. The delay of an individual access request is the number of rounds taken to satisfy that request. The per-request communication is defined as the total number of words in all messages sent divided by the number of access requests satisfied. We say that round t is *nice* if: (i) there exists a real constant δ, $0 \leq \delta < 1$, such that $r_{i+1}(t) \leq \delta s_i(t)$ for all i in $[m]$, and (ii) the probability that an arbitrary access request is satisfied in round t is $\Omega(1)$. Thus, if round t is nice, for every object A, an expected constant fraction of the requests for A are satisfied in round t.

The first access model we consider is the *fixed* model, in which each new access request is independently drawn from a fixed probability distribution $\mathcal{D}$ on $\mathcal{A}$. The distribution $\mathcal{D}$ is specified by a vector $(p_0, \ldots, p_{m-1})$; for a random variable X drawn from $\mathcal{D}$, we have $\Pr[X = A_i] = p_i$. At the start of each round t, new requests drawn from $\mathcal{D}$ are generated and placed by an adversary on each of the nodes that do not currently have an outstanding request. The particular assignment of new requests to free clients can be arbitrarily determined by the adversary.

Theorem 3.1: *In the fixed model, there exists $t_0 = O(\log n)$ such that for any t, $t_0 \leq t \leq \text{poly}(n)$, round t is nice wvhp.*

It follows from Theorem 3.1 and the protocol definitions that in the fixed model, our protocol provides optimal throughput and optimal expected delay using optimal per-request communication.

Corollary 3.1.1: *In t rounds of the fixed model, where $\Omega(\log n) \leq t \leq O(\text{poly}(n))$, wvhp, the number of access requests satisfied is $\Omega(nt)$ using $O(\log n)$ per-request communication. Moreover, after $O(\log n)$ rounds, each access request is satisfied in expected $O(1)$ rounds.*

□

We note that in Corollary 3.1.1 the per-request communication is optimal since each object is of size $\Theta(\log n)$.

The bounds stated in Corollary 3.1.1 also hold for certain access patterns in which the popularity of an object may change considerably over time. In the *dynamic* model, an adversary assigns new requests to free clients subject to the following constraints for all i in $[m]$: (i) $q_i(0) = O(\log n)$, and (ii) for $t > 0$, $q_i(t) \leq \rho \max\{q_i(t') : \max\{0, t - \Gamma\} \leq t' < t\}$, where $\rho > 1$ is a real constant. We note that the dynamic model permits arbitrary decreases in the popularity of an object, and also permits large increases in certain cases.

Theorem 3.2: *In the dynamic model, for an appropriately chosen ρ it holds wvhp that each round is nice, the number of access requests satisfied per round is $\Omega(n)$, and the per-request communication is $O(\log n)$ words.*

The particular choice of the constant ρ in the above theorem depends on the rate of growth of b_i with i. As defined in Section 3.4, b_i grows as $2^i \log n$. We can establish the claim of Theorem 4.5 for arbitrarily larger values of ρ by choosing an appropriately large growth rate for b_i.

3.6 Analysis

Our analysis proceeds in two parts. In the first part, Sections 3.6.2 and 3.6.3, we establish certain properties of our protocol which hold under any access pattern model. In the second part, Sections 3.6.4 and 3.6.5, we restrict our attention to the fixed and dynamic models and prove Theorems 3.1 and 3.2, respectively. Before delving into the formal proofs, we give a brief overview of the analysis in Section 3.6.1.

To simplify the presentation, we assume that there is no contention among messages of distinct types. The message priorities defined in Section 3.4 can easily be used to remove this assumption.

3.6.1 Sketch of the Analysis

A significant portion of the analysis is devoted to establishing Invariants 1 and 2 under suitable assumptions. In Section 3.3, we presented an informal argument describing how the protocol maintains the invariants. The argument of Section 3.3, however, views each block in isolation and hence ignores the contention among messages that are destined to different blocks and, yet, to the same node. Such a contention is possible since all the hash functions share a common range, namely, the set of all nodes. We begin our analysis by proving in Section 3.6.2 that for an arbitrary set of access requests, the interaction among the different blocks is "small" enough that each block can be viewed in isolation.

We now give a brief sketch of the formal arguments developed in Section 3.6.2. The notion of a small interaction among different blocks is captured formally in the definition of a good round (see Definition 3.3 in Section 3.6.2.4). Loosely put, a round is good if for every block B only a small fraction of servers in B are inaccessible due to faults and/or contention with messages sent to other blocks.

Suppose that at the start of a particular round, we are given x_i requests for object A_i, where the x_i's are fixed integers such that $\sum_{i \in [m]} x_i$ is $O(n)$. Consider a block B of an object A. Let us first determine the number of the servers of B that are inaccessible due to faults. Since the hash function mapping the servers in B to the nodes of the network is random and independent of the node and link faults, it follows from a straightforward Chernoff-type argument that only a constant fraction of the servers in B are unreachable wvhp. It now remains to consider the servers that are swamped by messages destined for other blocks. Let U be the set of nodes u such that $O(\log n)$ words associated with messages sent to blocks other than B are destined to u. The total number of words sent in any round is $O(n \log n)$; therefore, $|U|$ is $\Omega(n)$. Since the hash function mapping B to the nodes is independent of the x_i's, it follows from a Chernoff-type argument that U contains a constant fraction of the nodes to which the servers in B are mapped. Hence, only a constant fraction of the servers in B are swamped with messages sent to other blocks. Thus, if we are given a fixed set of access requests at the start of any round t, then t is good wvhp.

After defining the notion of a good round, we formalize the informal discussion of Section 3.3 in Section 3.6.3 and show that the Invariants 1 and 2 hold wvhp whenever a round is good. The remainder of the analysis concerns the fixed and dynamic models of access patterns and is contained in Sections 3.6.4 and 3.6.5, respectively. While it is not too difficult to derive Theorems 3.1 and 3.2 from the invariants, the main challenge in the analysis of these models is to show that each round is good wvhp.

If the set of access requests are fixed for each round, then our informal discussion above implies that each round is good wvhp. However, in general, the vector $(s_0(t), \ldots, s_{m-1}(t))$ may depend on the particular actions of the protocol in the first t rounds; hence, $(s_0(t), \ldots, s_{m-1}(t))$ and the hash functions may be correlated among one another. (Recall that $s_i(t)$ denotes the number of outstanding requests for object A_i at the start of round t.) For example, if the choice of hash functions for objects A_0 and A_1 is such that a large number of servers in the 0th blocks for A_0 and A_1 map to the same subset of nodes, there will be a correlation between the number of requests for A_0 and A_1 at the start of each round. Although, intuitively, we do not expect such seemingly minor correlations to have a significant impact on the performance of the protocol, it is a technical challenge to establish this.

We overcome the problems associated with the aforementioned dependencies by obtaining tight estimates on the number of requests for each object at the start of any round. Our proof for each model is by an induction on the number of rounds. In the induction argument for round t: (i) we use the invariants of Section 3.6.3 to place a tight bound on the number of requests for each object at the start of round t such that the upper bound on the total number of requests is $O(n)$, and (ii) we invoke the results of Section 3.6.2 to show that round t is good, and hence establish the invariants for the next round.

3.6.2 Good Rounds

The goal of this section is to formalize the notion of a good round which we alluded to in Section 3.6.1. Sections 3.6.2.1, 3.6.2.2, and 3.6.2.3 develop some technical machinery and Section 3.6.2.4 gives the formal definition.

We begin by introducing some additional notation. Let an *α-message* refer to any message of type α. Let size(α) denote the number of words in an α-message. For any α and $i \in [b]$, let $N_\alpha(A, i, t)$ denote the set of servers in $B_i(A)$ that attempt any α-message in round t. (Here and in the rest of this section, we use the word "attempt" as a short form for "attempt to send".) Let $N'_\alpha(A, i, t)$ denote the set of servers S in $B_i(A)$ such that all of the α-messages attempted by S in round t are sent.

Let $M_\alpha(A, i, t)$ be the set of α-messages that are sent to $B_i(A)$ in round t. Let $M'_\alpha(A, i, t)$ denote the set of α-messages received by $B_i(A)$ in round t. Let $F(A, t)$ be the set of servers in $B_0(A)$ that send b_1 *server-frag* messages in round t. Let $G(A, t)$ denote the set of clients that send b_0 *frag-req* messages in round t.

If we assume a fault-free model with no upper bound on the number of words a node can send/receive in a single step, then it is easy to show that some of the sets defined above are related by well-studied balls-and-bins experiments. Unfortunately, in the presence of faults and contention, this is not true. However, we are able to establish similar relations using more complex balls-and-bins experiments which are defined in the following section.

3.6.2.1 Two Balls and Bins Experiments

Definition 3.1: *Let X denote a set of labels, let U denote a set of bins, let ε be a real in $[0,1)$, and let τ be a nonnegative integer. In a* **uniform** $(X, U, \varepsilon, \tau)$ **experiment**, *we are given a set $\{V_i : i \in X, V_i \subseteq U,$ and $|V_i| \leq \varepsilon|U|\}$, and the following steps are performed: (i) for each i in X, place a ball labeled i in each bin in $U \setminus V_i$, and (ii) for each bin that has more than τ balls, discard all but an arbitrary subset of at least τ balls. Let Y denote the set of remaining balls. We refer to the set of remaining balls as the outcome of the experiment.* □

Definition 3.2: *Let X denote a set of balls, and let U, ε, and τ be as defined in Definition 3.1. In a* **random** $(X, U, \varepsilon, \tau)$ **experiment**, *we are given a set $\{V_i : i \in X, V_i \subseteq U,$ and $|V_i| \leq \varepsilon|U|\}$, the following steps are performed: (i) throw the balls independently and uniformly at random into U, (ii) for each $i \in X$, if ball i lands in a bin in V_i, discard ball i, and (iii) for each bin that has more than τ balls, discard all but an arbitrary subset of at least τ balls. We refer to the set of remaining balls as the outcome of the experiment.* □

The following two sections contain analyses of the experiments introduced in Definitions 3.1 and 3.2, respectively.

3.6.2.2 Analysis of the Uniform Experiment

In this section, we analyze a uniform $(X, U, \varepsilon, \tau)$ experiment (see Definition 3.1). Let x and u denote $|X|$ and $|U|$, respectively. Throughout this section, we assume that u is at least $c_1 \log n$. Also, c_1 is assumed to be sufficiently large and ε to be sufficiently small. For any ball i in X and bin j in U, we say that i is *good* for j if j is not in V_i; otherwise, we say that i is *bad* for j.

Lemma 3.1: *The number of bins that receive at most $(1 - c\varepsilon)x$ balls is at most u/c.*

Proof: For each ball in X, the number of bad bins is at most εu. Thus, the total number of bad "ball-bin pairs" is at most εux. By an averaging argument, we obtain that the number of bins that are bad for at least $c\varepsilon x$ balls is at most u/c. ∎

Corollary 3.1.1: *The number of bins that receive at most $x/2$ balls is at most $u/10$.* ∎

3.6.2.3 Analysis of the Random Experiment

In this section, we analyze a random $(X, U, \varepsilon, \tau)$ experiment (see Definition 3.2). We are interested in bounds on random variables associated with the number of bins that receive at least (or at most) c balls for some positive integer c, where c is some positive integer. We refer to these variables as *threshold variables.* We are only concerned with threshold variables for which c is at most τ; hence, we can assume τ to be as large as $|X|$. Let x and u denote $|X|$ and $|U|$, respectively. Throughout this section, we assume that u is at least $c_1 \log n$. Also, c_1 is assumed to be sufficiently large and ε to be sufficiently small.

The theory of martingales provides a useful tool for analyzing the random experiment. Appendix B contains a brief discussion of martingales, including a large deviation bound that is used in the following lemma.

Lemma 3.2: *Let Z be a threshold variable in a random $(X, U, \varepsilon, \tau)$ experiment. For any $\lambda > 0$, we have:*

$$\Pr\left[|Z - E[Z]| > \lambda\sqrt{x}\right] < 2e^{-\lambda^2/2}.$$

Proof: The random experiment defines a probability distribution on the set of functions Ω from X to $U \cup \{\perp\}$, where $\perp$ is a special bin that contains the discarded balls. A random function g drawn from Ω satisfies the following. For any i in X, we have: (i) for any j in $U \setminus V_i$, $\Pr[g(i) = j] = 1/u$, (ii) for any j in V_i, $\Pr[g(i) = j] = 0$, and (iii) $\Pr[g(i) = \perp] = |V_i|/u$. Fix a gradation $\emptyset = B_0 \subset B_1 \subset \ldots \subset B_x = X$. Given any functional $L : \Omega \mapsto \mathbf{R}$, we define a martingale $Z_0, Z_1, \ldots, Z_x$ by setting

$$Z_i(h) = E[L(g) : g(b) = h(b) \text{ for all } b \text{ in } B_i].$$

Since the function associated with Z_i satisfies the Lipschitz condition defined in Theorem B.2, the desired claim follows from Theorem B.1. ∎

We extend the definition of threshold variables to a sequence of s random experiments, given by $(X_0, U_0, \varepsilon, \tau)$, ..., $(X_{s-1}, U_{s-1}, \varepsilon, \tau)$. A threshold variable Z for a sequence of s random experiments is the number of bins that receive at least (or at most) c balls in at least one of the s experiments, for some positive integer c.

Lemma 3.3: *Let Z be a threshold variable associated with a sequence of s random experiments. For any $\lambda > 0$, we have:*

$$\Pr\left[|Z - E[Z]| > \lambda\sqrt{\sum_{i\in[s]} X_i}\right] < 2e^{-\lambda^2/2}.$$

Proof: Similar to that of Lemma 3.2. □

In the remainder of the section, we use Lemmas 3.2 and 3.3 to obtain high probability bounds on certain threshold variables. Lemmas 3.4 and 3.5 consider a single random $(X, U, \varepsilon, \tau)$ experiment. As in Section 3.6.2.2, we say that bin j is *good* for ball i if j is not in V_i; otherwise, we say that j is *bad* for i.

Lemma 3.4: *Let c be a real number greater than 4. If x is at least $4cu$, then the number of bins that receive less than c balls is at most $u(1/e^c + 1/20 + \lambda)$ with probability at least $(1 - 2e^{-\lambda^2 u/(8c)})$.*

Proof: Let X' be an arbitrary $4cu$-size subset of X. Let U' be the set of bins j such that j is bad for at most $4c'\varepsilon cu$ balls in X'. By an averaging argument, we obtain that $|U'|$ is at least $(c' - 1)u/c'$.

Let Z denote the number of bins in U that receive less than c balls. Let i be a bin in U'. Let X'_i be the set of balls in X' that are good for i, and let x'_i denote $|X'_i|$. By the definition of U', x'_i is at least $(1 - c'\varepsilon)4cu$. The probability that i receives less than c balls is at most:

$$\begin{aligned}
& \sum_{0\le j<c} \binom{x'_i}{j}\left(1-\frac{1}{u}\right)^{x'_i-j}\frac{1}{u^j} \\
\le \quad & c\binom{x'_i}{c}\left(1-\frac{1}{u}\right)^{x'_i-c}\frac{i}{u^c}
\end{aligned}$$

$$
\begin{aligned}
&\le c(4e(1-c'\varepsilon))^c\left(1-\frac{1}{u}\right)^{(1-c'\varepsilon)4cu-c} \\
&\le c\left(\frac{4e(1-c'\varepsilon)}{e^{4(1-c'\varepsilon)-1/u}}\right)^c \\
&\le \frac{1}{e^c},
\end{aligned}
$$

for $c > 4$ and ε sufficiently smaller than $1/c'$.

We set c' to 20. Thus, $E[Z]$ is at most $(1/e^c + 1/20)u$. By Lemma 3.2, the probability that Z is at least $(1/e^c + 1/20 + \lambda)u$ is at most $2e^{-\lambda^2 u/(8c)}$. ∎

Corollary 3.4.1: *If c is sufficiently large, c_1 is sufficiently larger than c, and x is at least $4cu$, then the number of bins that receive less than c balls is at most $u/10$ wvhp.* ∎

Lemma 3.5: *If x is at most cu, then the number of bins that receive at least $2ec$ balls is at most $u(1/2^{2ec} + \lambda)$ with probability at least $1 - 2e^{-\lambda^2 u/(2c)}$.*

Proof: Let Z denote the number of bins that receive at least $2ec$ balls. For any bin i, the probability that i receives at least $2ec$ balls is at most $\binom{cu}{2ec}\frac{1}{u^{2ec}} \le \frac{1}{2^{2ec}}$. Thus, $E[Z]$ is at most $u/2^{2ec}$. By Lemma 3.2, the probability that Z exceeds $u(1/2^{2ec} + \lambda)$ is at most $2e^{-\lambda^2 u/(2c)}$. ∎

Corollary 3.5.1: *If c is sufficiently large, c_1 is sufficiently larger than c, and x is at most cu, then the number of bins that receive at least $2ec$ balls is at most $u/10$ wvhp.* ∎

In the following two lemmas, we analyze a sequence of random $(X, U, \varepsilon, \tau)$ experiments.

Lemma 3.6: *Let c be a positive integer constant. Consider a sequence of s random experiments, $(X_0, U_0, \varepsilon, \tau), \ldots, (X_{s-1}, U_{s-1}, \varepsilon, \tau)$, such that $c \le x_i = |X_i| \le u = |U_i|$ for all i in $[s]$, and $U_i \cap U_j$ is $\emptyset$ for $i \ne j$. The number of bins in $\cup_{i\in[s]} U_i$ that receive at least c balls is*

$$
O\left(\sum_{i\in[s]}(x_i^c/u^{c-1}) + \sqrt{\sum_{i\in[s]} x_i \log n}\right) \quad \textit{wvhp.}
$$

Proof: Let Z_i be the number of bins in U_i that receive at least c balls. Let Z equal $\sum_{i\in[s]} Z_i$. We first obtain an upper bound on $E[Z_i]$ for any i, and hence an upper bound on $E[Z]$.

Consider the ith experiment. The probability that j receives at least c balls is at most $\binom{x_i}{c}(1/u)^c = O((x_i/c)^c)$. Thus, $E[Z_i]$ is $O(x_i^c/u^{c-1})$ and $E[Z]$ is $O(\sum_{i\in[s]}(x_i^c/u^{c-1}))$. The desired claim follows from Lemma 3.3. □

Lemma 3.7: *Let ε_1 be a positive real constant in $(0,1]$, and let c be a positive integer constant. If s is a suitably large integer constant, then in any sequence of s random experiments $(X_0, U, \varepsilon, \tau), \ldots, (X_{s-1}, U, \varepsilon, \tau)$ satisfying $\varepsilon_1 u \le |X_i| \le u$ for all i in $[s]$, the number of bins in U that receive at least c balls in at least one of the s experiments is $9u/10$ wvhp.*

Proof: Let Z be the number of bins in U that receive at least c balls in at least one of the s experiments. We first obtain a lower bound on $E[Z]$.

Consider the ith experiment, namely the random $(X_i, U, \varepsilon, \tau)$ experiment. Let X_i' be an arbitrary $\varepsilon_1 u$-size subset of X_i. Let U_i' be the set of bins such that j is bad for at most $100\varepsilon\varepsilon_1 u$ balls in X_i'. By an averaging argument, we obtain that $|U_i'|$ is at least $99u/100$. Consider a bin j in U_i'. The probability that j receives at least c balls is at least:

$$\binom{(1-100\varepsilon\varepsilon_1)u}{c}\left(1-\frac{1}{u}\right)^{x_i-c}\frac{1}{u^c} = f(\varepsilon, \varepsilon_1, c),$$

where $f(\varepsilon, \varepsilon_1, c)$ is a constant in $[0,1]$, dependent on ε, ε_1, and c.

Let U' be the set of bins j such that j is in U_i' for at least $s/2$ different values of i. By an averaging argument, we obtain that $|U'|$ is at least $49u/50$. Consider any bin j in U'. The probability that j did not receive c balls in any of the s experiments is at most

$$(1 - f(\varepsilon, \varepsilon_1, c))^{s/2} \ge 19/20,$$

for s chosen a suitably large constant. Thus, $E[Z]$ is at least $19u/20$. By Lemma 3.3, it follows that Z is $9u/10$ wvhp. □

3.6.2.4 Definition of a Good Round

Using Definitions 3.1 and 3.2, we now define the notion of a good round.

Definition 3.3: *Round t is* **good** *if there exists a sufficiently small real ε such that, for every object A, the following conditions hold:*

1. *Given any α and any i in $[b]$, if $|N_\alpha(A,i,t)|$ is $\Omega(\log n)$, then $|N'_\alpha(A,i,t)|$ is $\Omega(|N_\alpha(A,i,t)|)$.*

2. *If α is frag-req, then $M'_\alpha(A,0,t)$ is the outcome of a uniform $(G(A,t), B_0(A), \varepsilon, \Theta(\log n))$ experiment.*

3. *If α is server-frag, then $M'_\alpha(A,1,t)$ is the outcome of a uniform $(F(A,t), B_1(A), \varepsilon, \Theta(\log n))$ experiment.*

4. *If α is in $\{$obj-req, server-obj$\}$, then for any i in $[b]$, $M'_\alpha(A,i,t)$ is the outcome of a random $(M_\alpha(A,i,t), B_i(A), \varepsilon, \Theta(\log n)/\operatorname{size}(\alpha))$ experiment.* □

Let $T_\alpha(t)$ denote the set of all α-messages that are attempted in round t. For any set of messages X, let $\|X\|$ denote the total number of words in X. The following lemma places an upper bound on $\|T_\alpha(t)\|$.

Lemma 3.8: *For any i in $[m]$, any round t, and any α, $\|T_\alpha(t)\|$ is $O(n \log n)$ wvhp.*

Proof: The messages attempted in step $2t$ are described in Phase 0 of Figure 3.4. (No messages are attempted in Phase 1.) Only clients attempt messages in Phase 0 and at most one client resides at any node. In Phase 0, each client attempts $O(\log n)$ *frag-req* messages and $O(\log n)$ *obj-req* messages. Thus, for α in $\{$*frag-req*, *obj-req*$\}$, $\|T_\alpha(t)\|$ is $O(\log n)$.

The messages attempted in step $2t+1$ are described in Phase 2 of Figure 3.4. (No messages are attempted in Phase 3.) We consider different cases based on α. Let x_i equal $s_i(t)$.

- Case $\alpha =$ *client-frag*: At most $O(\log n)$ α-messages are sent by servers in $B_0(A_i)$ to each client requesting A_i. Therefore, $\|T_\alpha(t)\|$ is $O(\sum_{i\in[m]} x_i \log n) = O(n \log n)$.

- Case α = *server-frag*: A server S in $B_0(A_i)$ attempts an α-message only if S receives at least $\pi_1 b_0$ *client-frag* messages. Since each *client-frag* message received by S is from a different client, it follows that if S attempts any α-message, then x_i is at least $\pi_1 b_0$. Since the number of α-messages attempted by S is at most $\pi_0 b_0$, the total number of α-messages attempted in step $2t+1$ by servers in $B_0(A_i)$ is $O(\log^2 n) = O(x_i \log n)$. Since the size of each *server-frag* message is $\Theta(1)$, $\|T_\alpha(t)\|$ is $O(n \log n)$.

- Case $\alpha \in \{$*client-obj*, *server-obj*$\}$: A server S attempts a *client-obj* (resp., *server-obj*) message in step $2t+1$ only if it receives at least π_2 (resp., π_4) *obj-req* messages in step $2t$. We show for α = *client-obj* that $\|T_\alpha(t)\|$ is $O(n \log n)$ wvhp. A similar proof holds for α = *server-obj*.

  We partition the set of objects into two disjoint subsets L and H. For each i in $[m]$, if $x_i \leq \log n$, then A_i is in L; otherwise, A_i is in H. Consider an object A_i in H. Let j be the largest integer such that $b_j \leq x_i$. Since each server attempts at most π_3 α-messages, it follows that the total number of α-messages attempted by servers in $B_1(A_i)$ through $B_j(A_i)$ is $O(x_i)$. We now place an upper bound on the number of α-messages attempted by servers in $B_k(A_i)$ for $k > j$. Since each *obj-req* message is destined to a random server, even if all of the *obj-req* messages are received, we obtain by Lemma 3.6 that the number of servers in $\cup_{k>j} B_k(A_i)$ that receive at least π_2 *obj-req* messages in step $2t$ is $O(x_i)$ wvhp. Thus, the total number of attempted *client-obj* messages associated with objects in H is $O(n)$ wvhp.

  Consider the set L. Since each *obj-req* message is destined to a random server, even if all of the *obj-req* messages are received, we obtain by Lemma 3.6 that the number of servers in $\cup_{A_i \in L} \cup_{k>0} B_k(A_i)$ that receive at least π_2 *obj-req* messages in step $2t$ is:

$$O\left(\sum_{k>0}\left(\sum_{A_i \in L} x_i/2^k + \sqrt{\sum_{A_i \in L} x_i \log n}\right)\right) = O(n) \text{ wvhp.}$$

Adding the bounds for the α-messages associated with L and H, we obtain that $\|T_\alpha(t)\|$ is $O(n \log n)$ wvhp.

∎

Lemma 3.9: *If there are nonnegative reals $x_0, \ldots, x_{m-1}$, independent of the hash functions, such that $s_i(t) \le x_i$ and $\sum_{i \in [m]} x_i = O(n)$, then round t is good wvhp.*

Proof: We fix indices $i \in [b]$ and $j \in [m]$. We prove the desired claim by establishing the four parts of Definition 3.3. Let U denote the set of up nodes (i.e., neither dead nor down) in round t. Note that $|U|$ is at least $(1 - (\phi_0 + \phi_1))n$.

1. Let α be any message-type that is attempted by a server. Since servers attempt messages in odd steps only, this part concerns step $2t + 1$ only. By Lemma 3.8, $\|T_\alpha(t)\|$ is $O(n \log n)$ wvhp. Let U' be the set of nodes u such that at most $(c_0 \log n)/2$ words are attempted from u in step $2t+1$. By an averaging argument, it follows that $|U'|$ is $\Omega(n)$, where the hidden constant can be made arbitrarily close to 1 by setting c_0 sufficiently large. By definition, all messages in $N_\alpha(A_j, i, t)$ are attempted by servers. Since $x_0, \ldots, x_{m-1}$ are independent of the hash functions, the mapping of $B_i(A_j)$ is independent of the source nodes associated with the messages in $T_\alpha(t) \setminus N_\alpha(A_j, i, t)$. It follows from bounds on the tail of the hypergeometric distribution [38], given in Theorem A.2, that a constant fraction of the messages in $N(A_j, i, t)$ are attempted by nodes in U'. By setting c_0 and c_1 sufficiently large, we obtain that $|N'_\alpha(A_j, i, t)|$ is $\Omega(|N_\alpha(A_j, i, t)|)$ wvhp.

2. Let α equal *frag-req*. We need to prove that $M'_\alpha(A_j, 0, t)$ is the outcome of a uniform $(G(A_j, t), B_0(A_j), \varepsilon, \Theta(\log n))$ experiment. In our proof, the clients in $G(A_j, t)$ correspond to the labels, and the servers in $B_0(A_j)$ correspond to bins. Step (i) of the experiment corresponds to the following: each client in $G(A_j, t)$ sends one α-message to each server in $B_0(A_j)$. We now establish step (ii) of the experiment.

Consider a client C in $G(A_j, t)$. Let v be the node associated with C. Let U_v be the set of up nodes u in U such that at most $(c_0 \log n)/2$ words in $T_\alpha(t) \setminus M_\alpha(A_j, i, t)$ are destined to u and u has a non-faulty link to v. Since $\|T_\alpha(t)\|$ is $O(n \log n)$, by an averaging argument, it follows that $|U_v|$ is $\Omega(n)$, where the hidden constant is arbitrarily close to 1 for c_0 sufficiently large and ϕ_0, ϕ_1, and ϕ_2 sufficiently small. Let W_v be $B_0(A_j) \cap U_v$. Since the mapping of servers in $B_0(A_j)$ is independent of the destination nodes associated with the messages in $T_\alpha(t) \setminus M_\alpha(A_j, i, t)$, it follows from Theorem A.2 that $|W_v|$ is at least cb_i wvhp, where c can be set arbitrarily close to 1 for appropriate values of c_0, ϕ_0, ϕ_1, and ϕ_2.

The correspondence to Definition 3.1 is established by substituting $G(A_j, t)$, $B_0(A_j)$, $(1 - c)$, $\Theta(\log n)$, and $B_0(A_j) \setminus W_v$ for X, U, ε, τ, and V_i, respectively.

3. Similar to Part 2.

4. Let α be in {*obj-req*, *server-obj*}. We need to prove that $M'_\alpha(A_j, i, t)$ is the outcome of a random $(M_\alpha(A_j, i, t), B_i(A_j), \varepsilon, \Theta(\log n)/\operatorname{size}(\alpha))$ experiment. In our proof, the messages in $M_\alpha(A_j, i, t)$ correspond to balls, and the servers in $B_i(A_j)$ correspond to bins. Step (i) of the experiment corresponds to the following: each α-message is sent to a server chosen independently and uniformly at random from $B_i(A_j)$. We now account for steps (ii) and (iii) of the experiment.

   Consider any message y in $M(A_j, i, t)$. Let U_y be the set of up nodes $u \in U$ such that at most $(c_0 \log n)/2$ words in $T_\alpha(t) \setminus M_\alpha(A_j, i, t)$ are destined to u and u has a non-faulty link to the source of message y. Since $\|T_\alpha(t)\|$ is $O(n \log n)$, by an averaging argument, it follows that $|U_y|$ is $\Omega(n)$, where the hidden constant is arbitrarily close to 1 for c_0 sufficiently large and ϕ_0, ϕ_1, and ϕ_2 sufficiently small. Let W_y be the set of servers in $B_i(A_j)$ that are mapped to nodes in U_y. Since the mapping of servers in $B_i(A_j)$ is independent of $T_\alpha(t) \setminus M_\alpha(A_j, i, t)$, it follows from Theorem A.2 that $|W_y|$ is at least cb_i wvhp, where c can be set arbitrarily close to 1 for appropriate values of c_0, ϕ_0, ϕ_1, and ϕ_2. For step (iii), it is enough to note that each server in W_y can receive at least $(c_0 \log n)/2$ words from $M'(A_j, i, t)$.

The correspondence to Definition 3.2 is established by substituting $M_\alpha(A_j, i, t)$, $B_i(A_j)$, $(1-c)$, $\Theta(\log n)/\operatorname{size}(\alpha)$, and $B_i(A_j) \setminus W_y$ for X, U, ε, τ, and V_i, respectively.

□

3.6.3 Invariants

For any server S associated with object A, let $a(S,t)$ be 1 if server S has a copy of A or a fragment of A at the start of round t, and 0 otherwise. With each i in $[b]$, we associate a state variable $s(A,i,t) \in \{\text{complete}, \text{incomplete}\}$ that is defined as follows: if the number of servers S in $B_i(A)$ such that $a(S,t) = 1$ is at least $9b_i/10$, then $s(A,i,t)$ is complete; otherwise, $s(A,i,t)$ is incomplete. Let $R(A,t)$ denote the set of clients that attempt to access A in round t. Let $R'(A,t)$ denote the set of clients C that receive at least $b_0/4$ distinct *client-frag* messages or at least one *client-obj* message in round t. We define predicates P_0 through P_3 as follows:

- $P_0(A,t)$: If $|R(A,t)|$ is at least $2\pi_1 b_0$, then for $t+1 \leq t' \leq t+\Gamma$, $s(A,1,t')$ is complete.
- $P_1(A,i,t)$: If $|R(A,t)|$ is at least $4\pi_4 b_i$ and $s(A,i,t)$ is complete, then for $t+1 \leq t' \leq t+\Gamma$, $s(A,i,t')$ is complete.
- $P_2(A,t)$: If $|R(A,t)|$ is at most $\pi_0 b_0$, then $R'(A,t) = R(A,t)$.
- $P_3(A,i,t)$: If $s(A,i,t)$ is complete then: (i) if $|R(A,t)|$ is at least $\pi_3 b_i/12$, then $|R'(A,t)|$ is at least $\pi_3 b_i/120$, and (ii) if $4\pi_2 b_i \leq R(A,t) = O(b_i)$, then for each $C \in R(A,t)$, we have $\Pr[C \in R'(A,t)] = \Omega(1)$.

The predicates $P_0(A,t)$ and $P_1(A,i,t)$ (resp., $P_2(A,t)$ and $P_3(A,i,t)$) formalize Invariant 1 (resp., Invariant 2) of Section 3.1. We assume that $\pi_3 \geq \max\{4\pi_0, 24\pi_1, 48\pi_2\}$.

Lemma 3.10: *If round t is good, then the following predicates hold for every object A wvhp: (i) $P_0(A,t)$, (ii) $\forall i > 0 : P_1(A,i,t)$, (iii) $P_2(A,t)$, and (iv) $\forall i > 0 : P_3(A,i,t)$.*

Proof: Since at most one client resides at any node and each client attempts at most $c_0 \log n$ words, each attempted message of type *client-frag* or *obj-req* is sent.

1. Proof of $P_0(A, t)$: Let α and β equal *frag-req* and *server-frag*, respectively. We are given that $R(A, t)$ is $2\pi_1 b_0$. Since round t is good, it follows from part (ii) of Definition 3.3 that $M'_\alpha(A, 0, t)$ is the outcome of a uniform $(R(A, t), B_0(A), \varepsilon, \Theta(\log n))$ experiment.

   Let X be the set of servers in $B_0(A)$ that receive at least $\pi_1 b_0$ α-messages in step $2t$. By Corollary 3.1.1, $|X|$ is at least $9b_0/10$. Each server in X attempts b_1 β-messages in step $2t + 1$ (see Phase 2 of Figure 3.4), i.e., $N_\beta(A, 0, t) = X$. Since step $2t + 1$ is good, it follows from part (i) of Definition 3.3 that $|N'_\beta(A, 0, t)|$ is at least $b_0/2$.

   By definition, $F(A, t)$ equals $N'_\beta(A, 0, t)$. Since round t is good, by part (iii) of Definition 3.3, $M'_\beta(A, 1, t)$ is an outcome of a uniform $(M_\beta(A, 1, t), B_1(A), \varepsilon, \Theta(\log n))$ experiment. Let Y be the set of servers in $B_1(A)$ that receive at least $b_0/4$ β-messages. By Corollary 3.1.1, $|Y|$ is at least $9b_1/10$. Therefore, $s(A, 1, t + 1)$ is complete, and the desired claim follows from the cache capacity assumption.

2. Proof of $P_1(A, i, t)$ for all $i > 0$: Let α and β equal *obj-req* and *server-obj*, respectively. We are given that $R(A, t)$ is at least $4\pi_4 b_i$. Hence, $|M_\alpha(A, i, t)|$ is at least $4\pi_4 b_i$. Since round t is good, it follows from part (i) of Definition 3.3 that $M'_\alpha(A, i, t)$ is the outcome of an $(M_\alpha(A, i, t), B_i(A), \varepsilon, \Theta(1))$ experiment.

   Let X denote the set of servers in $B_i(A)$ that receive at least π_4 α-messages. By Corollary 3.4.1, $|X|$ is at least $9b_i/10$. Since $s(A, i, t)$ is complete, at most $b_i/10$ servers in $B_i(A)$ do not have a copy of A. Therefore, at least $4b_i/5$ servers in X attempt π_5 β-messages to $B_{i+1}(A)$ in step $2t + 1$. Thus, $|N_\beta(A, i + 1, t)|$ is at least $4\pi_5 b_i/5$. Since round t is good, by part (ii) of Definition 3.3, the number of servers in $B_i(A)$ all of whose attempted β-messages are sent is at least $2b_i/5$. Therefore, $|M_\beta(A, i + 1, t)|$ is $2\pi_5 b_i/5$.

Since round t is good, it follows from part (i) of Definition 3.3 that $M'_\beta(A, i+1, t)$ is an outcome of an $(M_\beta(A, i+1, t), B_{i+1}(A), \varepsilon, \Theta(1))$ experiment. By Corollary 3.4.1, the number of servers in $B_{i+1}(A)$ that receive at least $\pi_5/20$ β-messages is $9b_{i+1}/10$ wvhp. Thus, $s(A, i+1, t+1)$ is complete wvhp, and the desired claim follows from the cache capacity assumption.

3. Proof of $P_2(A,t)$: Let α and β equal *frag-req* and *client-frag*, respectively. By the hypothesis of $P_2(A,t)$, we are given that $|R(A,t)|$ is at most $\pi_0 b_0$. Since round t is good, it follows from part (ii) of Definition 3.3 that $M'_\alpha(A, 0, t)$ is the outcome of a uniform $(M_\alpha(A, 0, t), B_0(A), \varepsilon, \Theta(\log n))$ experiment. Thus, for any client C, $(1-\varepsilon)b_0$ of the α-messages sent by C are received by $(1-\varepsilon)b_0$ different servers in $B_0(A)$.

   Since each server that receives an α-message attempts at least one β-message, it follows that $|N_\beta(A, 0, t)|$ is at least $(1-\varepsilon)b_0$. Since round t is good, by part (i) of Definition 3.3, all but $2b_0/5$ of the servers in $B_0(A)$ send all their attempted β-messages. All of the β-messages that are sent by servers in $B_0(A)$ are received by the clients. It follows that C receives β-messages from $(1-\varepsilon-2/5)b_0$ different servers in $B_0(A)$. Hence, the request by client C is satisfied in round t.

4. Proof of $P_3(A,i,t)$ for all $i > 0$: Let α and β equal *obj-req* and *client-obj*, respectively. By definition, $|M_\alpha(A,i,t)|$ is $|R(A,t)|$. Since round t is good, it follows from part (iv) of Definition 3.3 that $M'_\alpha(A,i,t)$ is the outcome of an $(M_\alpha(A,i,t), B_i(A), \varepsilon, \Theta(1))$ experiment. For part (i) of P_3, we are given that $\pi_3 b_i/12 \le |R(A,t)| \le \pi_3 b_i/6$. By Corollaries 3.4.1 and 3.5.1, at least $4b_i/5$ servers in $B_i(A)$ receive at least $\pi_3/48 \ge \pi_2$ and at most π_3 α-messages wvhp. Hence, $|N_\beta(A,i,t+1)|$ is $4b_i/5$ wvhp. Since round t is good, by part (i) of Definition 3.3, it follows that $|N'_\beta(A,i,t)|$ is $2b_i/5$. If a client C is sent at least one β-message, C receives at least one β-message. Therefore, $\pi_3 b_i/120 \ge |R(A,t)|/20$ of the clients receive a β-message wvhp, establishing part (i).

   The proof of part (ii) is similar. We are given that $4\pi_2 b_i/12 \le |R(A,t)| = O(b_i)$.

By Corollary 3.4.1 and Lemma 3.5, at least $4b_i/5$ servers in $B_i(A)$ receive at least π_2 and at most $O(1)$ α-messages. By part (i) of Definition 3.3, it follows that $|N'_\beta(A,i,t)|$ is $2b_i/5$. Thus, $\Omega(|R(A,t)|)$ of the clients receive a β-message. Since each client C selects a server at random and each server sends β-messages to a random subset of requesting clients, part (ii) of the claim follows.

$\square$

For any round t, we define $d_i(t)$ to be the largest index j such that $s(A_i,k,t)$ is complete for all k in $[j+1]$. Let $e_i(t)$ be the smallest index j such that $a(S,t)$ is 0 for every server S in $\cup_{k\geq j} B_k(A_i)$. In the following lemma, we use Lemma 3.10 to relate $d_i(t)$, $e_i(t)$, $r_i(t)$, and $s_i(t)$ when t is good.

Lemma 3.11: *Let t be a good round. For any i in $[m]$:*

1. *If $s_i(t)$ is at most $\pi_0 b_0$, then $r_i(t+1) = 0$ wvhp.*
2. *If $s_i(t)$ is at most $\pi_3 b_{d_i(t)}/6$, then $s_i(t) - r_i(t+1)$ is at least $s_i(t)/20$ wvhp.*
3. *If $s_i(t)$ is at least $4\pi_4 b_j$ and $d_i(t)$ is at least j, then $d_i(t+1)$ is at least $j+1$.*
4. *$s_i(t) - r_i(t+1)$ is at most $\pi_3 b_{e_i(t)}$.*

Proof: By definition, $s_i(t)$ equals $|R(A_i,t)|$ and $s_i(t) - r_i(t+1)$ equals $|R'(A_i,t)|$. Since round t is good, we invoke Lemma 3.10 to prove the claims.

1. The claim directly follows from Part (iii) of Lemma 3.10.
2. There exists an integer $j \leq d_i(t)$ such that $\pi_3 b_j/12 \leq s_i(t) \leq \pi_3 b_j/6$. Therefore, by Part (iv) of Lemma 3.10, $s_i(t) - r_i(t+1)$ is at least $s_i(k)/20$ wvhp.
3. If $s_i(t)$ is at least $4\pi_4 b_j$ and $d_i(t)$ is at least j, then for each k in $[j+1]$, $s_i(t)$ is at least $4\pi_4 b_k$. Therefore, Parts (i) and (ii) of Lemma 3.10 imply that $s(A_i,k,t+1)$ is complete for all j in $[j+2]$, establishing the desired claim.

4. Each server in $B_0(A)$ sends at most $\pi_0 b_0$ *client-frag* messages. Since each client requires $b_0/4$ fragments to reconstruct the object, the number of clients whose requests are satisfied by B_0 is at most $4\pi_0 b_0 \le \pi_3 b_0$. The number of requests satisfied by each server in a block $B_j(A_i)$ for $j > 0$ is at most π_3. Therefore $s_i(t) - r_i(t+1)$ is at most $\sum_{0 \le j < e_i(t)} \pi_3 b_j$, which is at most $\pi_3 b_{e_i(t)}$.

□

3.6.4 The Fixed Model

The fixed model is specified by a probability distribution $\mathcal{D} = (p_0, \ldots, p_{m-1})$, where each new request is for A_i with probability p_i. We use a number of positive real constants in our analysis. Constants a_0 through a_6 appear in the definitions and the statements of the lemmas and are required to satisfy the following inequalities.

$$
\begin{aligned}
a_0 &\le \pi_0 c_1 / a_3, \\
a_1 &\ge 9 a_3 \pi_3^2 / (a_2 a_4), \\
a_2 &\ge \max\{4\pi_4, 2\pi_1\}, \\
a_2 &\le \pi_3 / 12, \\
a_5 &= 1/(1 - 1/20 + a_3/(a_6 - a_3)), \text{ and} \\
a_6 &> 21 a_3.
\end{aligned}
$$

The above inequalities are satisfied if the constants are selected as follows. Constants a_3 and a_4 are first selected according to the desired failure probability of the protocol. Then, a_1, a_2, and a_6 are selected so that $\pi_4, \pi_1 \ll a_2 \ll \pi_3 \ll a_1$ and $a_3 \ll a_6$. Finally, a_0 is chosen appropriately. (The inequalities associated with other constants that arise in the proofs are specified whenever such constants are introduced.)

We partition the set $\mathcal{A}$ into $O(\log n)$ subsets as follows:

$$
\mathcal{A}_j = \begin{cases} \{A_i : np_i \le a_0 \log n\}, \text{ if } j = 0, \\ \{A_i : a_0 a_1^{j-1} \log n < np_i \le a_0 a_1^j \log n\}, \text{ otherwise.} \end{cases}
$$

Let $\mathcal{A}_{\geq i}$ denote the set $\cup_{j \geq i} \mathcal{A}_j$. We define $\mathcal{A}_{\leq i}$, $\mathcal{A}_{>i}$, and $\mathcal{A}_{<i}$ similarly. For each i in $[m]$, object A_i is said to be *good* in round t if $s_i(t) \leq a_2 b_{d_i(t)}$; otherwise, A_i is said to be *bad* in round t. Thus, if an object A_i is good in round t, then at the start of round t, the number of requests for A_i is at most a constant factor times the number of copies of A_i.

Let $B(m,p)$ be the random variable denoting the number of successes in m independent Bernoulli trials with success probability p. Let a_3 and a_4 be real constants such that for $p \geq a_0 \log n/n$, $a_4 np \leq B(n,p) \leq a_3 np$ wvhp; a_3 and a_4 may be obtained from standard Chernoff bounds [37] given in Theorem A.1.

Lemma 3.12: *Let rounds* 0 *through* $r-1$ *be good. If object* A_i *is bad in rounds* 0 *through* r*, then wvhp we have* $d_i(j) = j$ *and* $e_i(j) = j+1$ *for* $0 \leq j < r$.

Proof: The proof is by induction on j. The induction basis is trivial. For the induction hypothesis, we assume that $d_i(j) = j$ and $e_i(j) = j+1$ for all j in $[k]$, where k is in $[r-1]$. Since A_i is bad in round $k-1$, we have $s_i(k-1) > a_2 b_{d_i(k-1)}$. Since round $k-1$ is good, by Part 3 of Lemma 3.11, $d_i(k)$ is k wvhp. Since $e_i(k-1)+1 \leq e_i(k) < d_i(k)$ and $e_i(k-1) = k$, $e_i(k)$ is $k+1$ wvhp. ∎

Lemma 3.13: *Let rounds* 0 *through* $r-1$ *be good. If* A_i *is not in* $\mathcal{A}_0$ *and* A_i *is bad in rounds* 0 *through* r*, then* $s_i(r) \geq a_2 a_4 np_i/(3\pi_3)$ *wvhp.*

Proof: By Lemma 3.12, $d_i(j)$ is j for all j in $[r+1]$. By Part 4 of Lemma 3.11, it follows that $s_i(j) - r_i(j+1)$ is at most $2\pi_3 b_j$. Thus, we have wvhp:

$$\begin{aligned}
s_i(r) &= s_i(0) + \sum_{0 \leq j \leq r} q_i(j) - \sum_{0 \leq j < r} (s_i(j) - r_i(j+1)) \\
&\geq s_i(0) - \sum_{0 \leq j < r} 2\pi_3 b_j \\
&\geq s_i(0) - 2\pi_3 b_r \\
&\geq a_4 np_i - 2\pi_3 b_r \\
&\geq a_4 np_i - 2\pi_3 s_i(r)/a_2
\end{aligned}$$

$$\geq a_4 np_i/(1+2\pi_3/a_2)$$
$$\geq a_2 a_4 np_i/(3\pi_3).$$

(The first equation follows from the definition of s_i. The fourth equation follows from a Chernoff bound. The fifth equation holds since A_i is bad in round r. The last equation holds since $a_2 \leq \pi_3$.) ∎

Lemma 3.14 places an upper bound on the number of requests for object A_i at the start of a round subsequent to the first round in which A_i is good.

Lemma 3.14: *Let rounds* 0 *through* $r-1$ *be good, where* r *is at most* Γ. *Assume that* A_i *does not belong to* $\mathcal{A}_0$, *and let* $j < r$ *be the smallest integer such that* A_i *is good in round* j. *There exist constants* $a_5 > 1$ *and* a_6 *such that, for* $j \leq k < r$, *we have wvhp:*

$$s_i(k) \leq \max\left\{\frac{s_i(j)}{a_5^{k-j}}, a_6 np_i\right\}. \tag{3.1}$$

Proof: By a Chernoff bound, $q_i(k+1)$ is at most $a_3 np_i$ wvhp. If $s_i(k) \leq (a_6 - a_3)np_i$, then $s_i(k+1) \leq s_i(k) + q_i(k+1) \leq a_6 np_i$ wvhp, thus establishing the claim. For the remainder of the proof, we assume that:

$$s_i(k) > (a_6 - a_3)np_i. \tag{3.2}$$

We consider two cases depending on whether $s_i(j) \geq a_6 np_i$.

- Case $s_i(j) \geq a_6 np_i$: We show by induction on k that Equation 3.1 holds wvhp. The induction basis is trivially true. Let Equation 3.1 be true for rounds j through k, where $j \leq k < r-1$. Since j is the first round in which A_i is good, Lemma 3.12 implies that $d_i(j)$ is j. Therefore, $s_i(j) \leq a_2 b_j$ wvhp.

  By the induction hypothesis, $s_i(k) \leq s_i(j)$. Moreover, since r is at most Γ, $d_i(k)$ is at least j. Therefore, $s_i(k) \leq a_2 b_j \leq a_2 b_{d_i(k)}$. Since a_2 is at most $\pi_3/6$, Part 2 of Lemma 3.11 implies wvhp:

$$s_i(k) - r_i(k+1) \geq s_i(k)/20. \tag{3.3}$$

Therefore we have wvhp:

$$\begin{aligned}
s_i(k+1) &= r_i(k+1) + q_i(k+1) \\
&= s_i(k) - (s_i(k) - r_i(k+1)) + q_i(k+1) \\
&\leq s_i(k)(1 - 1/20) + a_3 s_i(k)/(a_6 - a_3) \\
&\leq s_i(k)(1 - 1/20 + a_3/(a_6 - a_3)) \\
&\leq s_i(k)/a_5 \\
&\leq \max\left\{\frac{s_i(j)}{a_5^{k-j}}, a_6 n p_i\right\}.
\end{aligned}$$

(The second equation follows from the definition of s_i. The third equation follows from Equations 3.2 and 3.3. The fifth equation follows from the choice of the constants: $a_6 > 21a_3$ and $1/a_5 = (1 - 1/20 + a_3/(a_6 - a_3))$.)

- Case $s_i(j) < a_6 n p_i$: We show that in this case $s_i(k) \leq a_6 n p_i$ wvhp. The proof is by induction on k. The induction basis is trivial. Let the claim be true for rounds j through k where $j \leq k < r$. In the induction step, we need to show that $s_i(k+1) \leq a_6 n p_i$.

  Let ℓ be the last round in which $s_i(\ell) \leq a_2 b_{d_i(\ell)}$. (Such an ℓ exists since $s_i(j) \leq a_2 b_{d_i(j)}$.) By Part 3 of Lemma 3.11 and the definition of ℓ, $d_i(k) \geq d_i(\ell) + (k - \ell)$ wvhp. By a Chernoff bound, $s_i(\ell)$ is at least $(a_6 - (k-\ell)a_3)np_i$ wvhp. Therefore, $b_{d_i(\ell)}$ is at least $(a_6 - (k-\ell)a_3)np_i/a_2$ wvhp. Moreover, since ℓ is at most Γ, $d_i(\ell) \geq j$. Therefore, we have wvhp:

$$\begin{aligned}
b_{d_i(\ell)} &\geq b_j \\
&\geq \frac{2s_i(j-1)}{2a_2 + \pi_3} \\
&\geq \frac{2a_2 a_4 n p_i}{9\pi_3^2}.
\end{aligned}$$

  (The second equation holds since A_i is good in round j. The third equation follows from Lemma 3.13.) Therefore, we have wvhp:

$$b_{d_i(\ell)} \geq \max\left\{(a_6 - (k-\ell)a_3), \frac{2a_2^2 a_4}{9\pi_3^2}\right\}\frac{np_i}{a_2}. \tag{3.4}$$

We select a_6 and a new constant a_7 such that $2a_3a_7 \leq a_6 \leq 2^{a_7+2}a_2a_4/(9\pi_3^2)$. If $k-\ell$ is at most a_7, then $b_{d_i(k)}$ is at least $a_6np_i/2a_2$. By the induction hypothesis, $s_i(k)$ is at most a_6np_i. Therefore, $s_i(k) \leq 2a_2b_{d_i(k)}$. If $k-\ell$ is at least a_7, then $b_{d_i(k)}$ is at least $2^{a_7+1}a_2a_4np_i/(9\pi_3^2)$. By the choice of a_6 and a_7, we obtain that $s_i(k) \leq 2a_2b_{d_i(k)}$ wvhp.

By Part 2 of Lemma 3.11, we have wvhp: $s_i(k) - r_i(k+1)$ is at least $s_i(k)/20$. Therefore, $s_i(k+1)$ is at most $19s_i(k)/20 + a_3np_i$, which is at most a_6np_i by the choice of the constants.

∎

Lemma 3.15 relates the number of requests, in round r, for any two objects that are bad in rounds 0 through $r-1$.

Lemma 3.15: *Let rounds 0 through $r-1$ be good. Let i_1 and i_2 be integers in $[m]$ such that A_{i_1} and A_{i_2} are not in $\mathcal{A}_0$ and A_{i_2} is bad in rounds 0 through r. Then, wvhp we have:*

$$s_{i_1}(r) \leq \frac{3a_3\pi_3p_{i_1}}{a_2a_4p_{i_2}}s_{i_2}(r).$$

Proof: Consider any round j, $0 \leq j < r$. We are given that A_{i_2} does not become good in any of the r rounds. We invoke Lemma 3.12 and obtain that $d_{i_2}(j) = j$ wvhp. Therefore, $s_{i_2}(j) > a_2b_j$ wvhp. By Part 4 of Lemma 3.11 $(s_{i_2}(j) - r_{i_2}(j+1))$ is at most $2\pi_3b_j$ wvhp. Let q denote the number of new requests generated in rounds 0 through r. Since $q \geq n$ and $A_{i_1}, A_{i_2} \notin \mathcal{A}_0$ we have wvhp: $\sum_{0\leq j\leq r} q_{i_1}(j)$ is at least $a_4p_{i_1}q$ and $\sum_{0\leq j\leq r} q_{i_2}(j)$ is at most $a_3p_{i_2}q$. We thus have:

$$\begin{aligned} s_{i_1}(r) &\leq a_3p_{i_1}q, && (3.5)\\ s_{i_2}(j) &\geq a_4p_{i_2}q - \sum_{0\leq j<r} 2\pi_3b_j. && (3.6)\end{aligned}$$

From Equations 3.5 and 3.6, we obtain wvhp:

$$s_{i_1}(r) \leq a_3p_{i_1}q$$

$$
\begin{aligned}
&\leq \frac{a_3 p_{i_1}}{a_4 p_{i_2}}(s_{i_2}(r) + 2\pi_3 b_r) \\
&\leq \frac{3a_3\pi_3 p_{i_1}}{a_2 a_4 p_{i_2}} s_{i_2}(r).
\end{aligned}
$$

(The first equation follows from Equation 3.5. The second equation follows from Equation 3.6. The last equation holds since A_{i_2} is bad in round r and $a_2 \leq \pi_3$.) ∎

Lemma 3.16 shows that the number of rounds taken for an object to become good grows almost logarithmically with its probability of access. Lemma 3.17 states that each object in $\mathcal{A}_0$ is good in all rounds.

Lemma 3.16: *Let rounds* 0 *through* $r-1$ *be good. Let* i_1 *and* i_2 *be in* $[m]$. *Let* j *in* $[r]$ *be the smallest integer such that* A_{i_1} *is good in round* j. *If* $p_{i_1} \geq a_1^k p_{i_2}$, *where* k *is a positive integer, then there exists* $j' \leq j-k+1$ *such that* A_{i_2} *is good in round* j'. *If* $p_{i_1} \geq p_{i_2}/a_1$, *then there exists* $j' \leq j + O(1)$ *such that* A_{i_2} *is good in round* j'.

Proof: We first consider the case in which $p_{i_1} \geq a_1^k p_{i_2}$ for some positive integer k. Since A_{i_1} is bad in rounds 0 through $j-1$, $d_{i_1}(j) = j$ wvhp by Lemma 3.12. Since A_{i_1} is good in round j, we have $s_{i_1}(j) \leq a_2 b_j$ wvhp, and we obtain an upper bound on $s_{i_1}(j-k+1)$ as follows:

$$
\begin{aligned}
s_{i_1}(j-k+1) &= s_{i_1}(j) + \left(\sum_{0 \leq \ell < k} (s_{i_1}(j-\ell) - r(j-\ell+1)) - q(j-\ell+1) \right) \\
&\leq s_{i_1}(j) + \left(\sum_{0 \leq \ell < k} (s_{i_1}(j-\ell) - r(j-\ell+1)) \right) \\
&\leq s_{i_1}(j) + 2\pi_3 b_j \\
&\leq 3\pi_3 b_j. \qquad (3.7)
\end{aligned}
$$

(The first equality follows from the definition of s_{i_1}. The third equation follows from Part 1 of Lemma 3.11. The last equation holds since A_{i_1} is good in round j and $a_2 \leq \pi_3$.)

If A_{i_2} is good in some round $j' < j-k+1$, then the claim holds. Otherwise, by Lemma 3.15, it holds wvhp that $s_{i_2}(j-k+1) \leq \frac{3a_3\pi_3}{a_2 a_4 a_1^k} s_{i_1}(j-k+1)$. Hence, A_{i_2} is good in round $j-k+1$ wvhp because:

$$\begin{aligned}
s_{i_2}(j-k+1) &\leq \frac{9a_3(\pi_3)^2}{a_1^k a_2 a_4} b_j \\
&\leq a_2 b_{j-k+1}.
\end{aligned}$$

(The first equation follows from Equation 3.7. The second equation follows from the choice of constants: $a_1 \geq 9a_3\pi_3^2/(a_2a_4) \geq 2$.)

We now consider the case in which $p_{i_1} \geq p_{i_2}/a_1$. Let a_8 be an integer constant satisfying:

$$2^{a_8} \geq 3a_2^2(a_2 + 3a_8\pi_3)(a_2 + \pi_3)a_1a_3\pi_3/(a_2a_4).$$

(Thus, a_8 is a sufficiently large integer constant.) If A_{i_2} is good in some round $j' < j+a_8$, then the claim holds. Otherwise, A_{i_2} is good in round $j + a_8$ wvhp because:

$$\begin{aligned}
s_{i_2}(j+a_8) &\leq s_{i_2}(j-1) + a_4a_8np_{i_2} \\
&\leq s_{i_2}(j-1)(1 + 3\pi_3a_8/a_2) \\
&\leq \frac{3(1+3\pi_3a_8/a_2)a_1a_3\pi_3}{a_2a_4} s_{i_1}(j-1) \\
&\leq \frac{3(1+3\pi_3a_8/a_2)a_1a_3\pi_3}{a_2a_4} (s_{i_1}(j) + \pi_3 b_j) \\
&\leq \frac{3(1+3\pi_3a_8/a_2)a_1a_3\pi_3(a_2+\pi_3)}{a_2a_4} b_j \\
&\leq a_2 b_{j+a_8}.
\end{aligned}$$

(The first equation follows from the definition of s_{i_2}. The second equation follows from Lemma 3.13. The third equation follows from Lemma 3.15. The fourth equation follows from the definition of s_{i_1}. The fifth equation holds since A_{i_1} is good in round j. The last equation follows from the choice of a_8.) ∎

Lemma 3.17: *Let rounds* 0 *through* $r-1$ *be good. For any nonnegative integer* i *such that* $0 \leq i < m$ *and* $A_i \in \mathcal{A}_0$, *we have* $r_i(r) = 0$.

Proof: We will prove by induction on j that for $0 \leq j \leq r$, $r_i(j)$ is zero. The base case is trivial. Let the claim hold for j. Consider round $j+1 \leq r$. Since $r_i(j)$ is zero,

$s_i(j)$ equals $q_i(j)$, which is at most $\pi_0 b_0$ wvhp. Since round j is good, by Part 1 of Lemma 3.11, it follows that $r_i(j+1)$ equals zero wvhp. ∎

In Definition 3.4, we define $O(\log n)$ m-vectors of the form $(s_0^*(j), \ldots, s_{m-1}^*(j))$. These vectors are used in Lemma 3.9 to place an upper bound on the number of requests for each object at the start of a given round.

Definition 3.4: *For nonnegative integers i and j, $0 \le i < m$, we define:*

$$s_i^*(j) = \begin{cases} B(n, p_i) & \text{if } A_i \in \mathcal{A}_0, \\ \frac{np_i}{a_5^{j-\ell} \sum_{A_k \in \mathcal{A}_{\ge \ell}} p_k} + np_i & \text{if } A_i \in A_\ell, 0 < \ell \le j \\ \frac{np_i}{\sum_{A_k \in \mathcal{A}_{>j}} p_k} + np_i & \text{otherwise.} \end{cases}$$

∎

Lemma 3.18: *For all $j > 0$, $\sum_{0 \le i < m} s_i^*(j)$ is $O(n)$ wvhp.*

Proof: We rewrite $\sum_{0 \le i < m} s_i^*(j)$ as follows:

$$\sum_{0 \le i < m} s_i^*(j) = \sum_{i : A_i \in \mathcal{A}_0} s_i^*(j) + \sum_{i : A_i \in \mathcal{A}_{\le j} \cap \mathcal{A}_{>0}} s_i^*(j) + \sum_{i : A_i \in \mathcal{A}_{>j}} s_i^*(j).$$

We establish the lemma by obtaining upper bounds on the three terms in the right-hand side of the above equation. The first sum is at most $a_4 n$ wvhp. The second sum is bounded as follows:

$$\begin{aligned} \sum_{i : A_i \in \mathcal{A}_{\le j} \cap \mathcal{A}_{>0}} s_i^*(j) &= \left(\sum_{0 < \ell \le j} \sum_{i : A_i \in A_\ell} \frac{np_i}{a_5^{j-\ell} \sum_{A_k \in \mathcal{A}_{\ge \ell}} p_k} \right) + \sum_{i : A_i \in \mathcal{A}_{\le j}} np_i \\ &\le \left(\sum_{0 < \ell \le j} \sum_{i : A_i \in A_\ell} \frac{np_i}{a_5^{j-\ell} \sum_{A_k \in A_\ell} p_k} \right) + \sum_{i : A_i \in \mathcal{A}_{\le j}} np_i \\ &= \sum_{0 < \ell \le j} \frac{n}{a_5^{j-\ell}} + \sum_{i : A_i \in \mathcal{A}_{\le j}} np_i \\ &\le \frac{n}{1 - a_5} + \sum_{i : A_i \in \mathcal{A}_{\le j}} np_i. \end{aligned} \tag{3.8}$$

Similarly, we bound the third sum as follows:

$$\begin{aligned}
\sum_{i:A_i \in \mathcal{A}_{>j}} s_i^*(j) &= \left(\sum_{i:A_i \in A_{\geq j}} \frac{np_i}{\sum_{A_k \in \mathcal{A}_{\geq j}} p_k} \right) + \sum_{i:A_i \in \mathcal{A}_{>j}} np_i \\
&= n + \sum_{i:A_i \in \mathcal{A}_{>j}} np_i. \qquad (3.9)
\end{aligned}$$

It follows from the bounds on the three sums that $\sum_{0 \leq i < m} s_j^*$ is $O(n)$. □

Lemma 3.19 shows that each of the first $O(\log n)$ rounds is good wvhp.

Lemma 3.19: *Let rounds* 0 *through* $r-1$ *be good, where* r *is at most* Γ. *Wvhp, round* r *is good.*

Proof: We show that there exists an integer h such that for all i in $[m]$, $s_i(r)$ is $O(s_i^*(h))$ wvhp. We divide $\mathcal{A}$ into three groups $\mathcal{A}_0$, S, and U. Let S be the set $\{A_i \in \mathcal{A} \setminus \mathcal{A}_0 :$ there exists $j \leq r$ such that A_i is good in round $j\}$. Let U be the set $\mathcal{A} \setminus (S \cup \mathcal{A}_0)$.

We first consider any object A_i in $\mathcal{A}_0$. By Lemma 3.17, we have $s_i(r) = q_i(r) \leq B(n, p_i)$.

Let h be the largest index such that $\mathcal{A}_h \cap S$ is nonempty. If S is empty then we set h to 0. By Lemma 3.16 and the definition of $\mathcal{A}_i$, it follows wvhp that for i in $[h]$, every object in $\mathcal{A}_i$ is good in some round $j' \leq r - h + i + 1$.

Consider any object $A_j \in S \cap A_i$ where $0 \leq i \leq h$. Let r_j be the smallest round in which A_j is good. By Lemma 3.16 and the definition of $\mathcal{A}_i$, it holds wvhp that every object $A_k \in \mathcal{A}_{\geq i}$ is bad in some round $r' = r_j - O(1)$. By Lemma 3.15, it holds that for $A_k \in \mathcal{A}_{\geq i}$, we have $s_k(r') = \Omega(p_k s_j(r')/p_j)$ wvhp. Therefore we have wvhp:

$$s_j(r_j - 1) = O\left(\frac{np_j}{\sum_{A_k \in \mathcal{A}_{\geq i}} p_k} + np_j \right),$$

and hence,

$$\begin{aligned}
s_j(r) &= O\left(\frac{np_j}{a_5^{r-r_j} \sum_{A_k \in \mathcal{A}_{\geq i}} p_k} + np_j \right) \\
&= O\left(\frac{np_j}{a_5^{h-i-1} \sum_{A_k \in \mathcal{A}_{\geq i}} p_k} + np_j \right) \\
&= O(s_j^*(h)).
\end{aligned}$$

(The first equation follows from Lemma 3.15. The second equation follows from the earlier claim that $r_j \le r - h + i + 1$. The last equation follows from the definition of $s_j^*(h)$.)

We now consider the objects in U. Let h' be the smallest index such that $\mathcal{A}_{h'} \cap U \neq \emptyset$. If U is empty, then let h' equal h. By Lemma 3.16, h' is at least $h-1$ wvhp. By Lemma 3.16, it holds wvhp that every object in $\mathcal{A}_{\ge h'}$ is bad in some round $r' = r - O(1)$. Consider any object $A_j \in U \cap A_i$, $i \ge h'$. By Lemma 3.15, for all objects $A_k \in \mathcal{A}_{\ge h'}$, it holds wvhp that $s_k(r') = \Omega(p_k s_j(r')/p_j)$. Therefore we have wvhp:

$$\begin{aligned} s_j(r) &= O\left(\frac{np_j}{\sum_{A_k \in \mathcal{A}_{\ge h'}} p_k} + np_j\right) \\ &= O(s_j^*(h)). \end{aligned}$$

(The first equation holds since $\sum_{\ell \in [m]} s_\ell(t)$ is at most n for any t. The second equation holds since $\mathcal{A}_{\ge h'} \supseteq \mathcal{A}_{\ge h}$.)

We have shown that for each j in $[m]$, $s_j(r)$ is $O(s_j^*(h))$ wvhp. By Lemmas 3.9 and 3.18, it follows that round r is good wvhp. □

We are now ready to establish tight bounds on $s_i(t)$ for each object A_i and round t. These bounds, which are proved in Lemma 3.20, imply that after $\Theta(\log n)$ rounds the number of copies of any object is at least a constant fraction of the number of requests for the object.

Let t_0 equal $b + \log_{a_5} n$. We assume that Γ is at least $t_0 + \Theta(1)$, where the hidden constant is specified in the following proof.

Lemma 3.20: *For any i in $[m]$ and for any $t \ge t_0$, we have wvhp:*

1. *If A_i is in $\mathcal{A}_0$, $s_i(t)$ is $B(\Theta(n), p_i)$.*
2. *If A_i is not in $\mathcal{A}_0$, $s_i(t)$ is $\Theta(np_i)$, and $b_{d_i(t)}$ is $\Omega(np_i)$.*
3. *Round t is good.*

Proof: By Lemma 3.19, rounds 0 through t_0 are good wvhp. For any i in $[m]$, if A_i is bad in rounds 0 through $b-2$, then by Lemma 3.12, $d_i(b-1)$ is $b-1$. Since

$a_2c_12^{b-1}\log n > n \geq s_i(b-1)$, A_i is good in round $b-1$. We have thus shown that for each i in $[m]$, A_i is good in some round j in $[b]$.

Part 1 follows directly from Part 2. We establish Parts 2 and 3 by showing that for any A_i not in $\mathcal{A}_0$, and $t \geq t_0$: (i) $s_i(t)$ is at most $a_{10}np_i$, (ii) round t is good, (iii) if $t > t_0$, $s_i(t)$ is at least $a_{12}np_i$, and (iv) $b_{d_i(t)}$ is at least $a_{11}np_i$. Constants a_{10}, a_{11}, and a_{12} are specified below.

The proof of the above four claims is by induction on t. For the induction basis, let t equal t_0. By Lemma 3.14, $s_i(t)$ is at most $a_6np_i \leq a_{10}np_i$ wvhp, thus establishing claim (i) (we set $a_{10} \geq a_6$). Claim (ii) follows from claim (i) and Lemma 3.9. Claim (iii) holds vacuously. Since Γ is $\Omega(\log n)$, Lemma 3.13 implies that $b_{d_i(t_0)}$ is at least $a_2a_4np_i/(9\pi_3^2) \geq a_{11}np_i$, thus establishing claim (iv) (we set $a_{11} \leq a_2a_4/(9\pi_3^2)$). This completes the induction basis.

For the induction step, we assume that claims (i), (ii), (iii), and (iv) hold for rounds t_0 through t. We first establish claim (i) for round $t+1$. If $s_i(t)$ is at most $(a_{10}-a_3)np_i$, then $s_i(t+1) \leq s_i(t) + q_i(t+1) \leq a_{10}np_i$ wvhp, and the desired claim holds.

We now consider the case in which $s_i(t)$ is at least $(a_{10}-a_3)np_i$. Let $\ell \geq t_0$ be the last round in which $s_i(\ell) \leq 9a_6\pi_3^2b_{d_i(\ell)}/(a_2a_4)$. (Such an ℓ exists as t_0 satisfies the inequality.) Since $9a_6\pi_3^2/(a_2a_4) \geq 4\pi_4$, Part 3 of Lemma 3.11 and the definition of ℓ imply that $d_i(t)$ is at least $d_i(\ell) + t - \ell$. By a Chernoff bound, $s_i(\ell)$ is at least $(a_{10} - (t-\ell)a_3)np_i$ wvhp. Therefore, $b_{d_i(\ell)}$ is at least $2a_2a_4(a_{10} - (t-\ell)a_3)np_i/(9a_6\pi_3^2)$ wvhp. Moreover, by the induction hypothesis, $b_{d_i(\ell)}$ is at least $a_{11}np_i$. Thus,

$$b_{d_i(\ell)} \geq \max\{\frac{2a_2a_4(a_{10}-(t-\ell)a_3)}{9a_6\pi_3^2}, a_{11}\}np_i. \tag{3.10}$$

We choose a_{13} and a_{10} such that $2a_3a_{13} \leq a_{10} \leq a_2a_{11}2^{a_{13}}$. If $t-\ell$ is at most a_{13}, then $b_{d_i(t)}$ is at least $a_2a_3a_4a_{10}np_i/(9\pi_3^2)$. By the induction hypothesis, $s_i(t)$ is at most $a_{10}np_i$. Therefore, $s_i(t)$ is at most $18\pi_3^2b_{d_i(t)}/(a_2a_4)$. By Part (iii) of Lemma 3.10, we have wvhp: $s_i(t) - r_i(t+1)$ is at least $a_2a_4s_i(t)/(2160a_6\pi_3)$ wvhp. If $t-\ell$ is at least a_{13}, then $b_{d_i(t)}$ is at least $2^{a_{13}}a_{11}np_i$ wvhp. By the choice of constants, we obtain that $s_i(t)$

is at most $a_2 b_{d_i(t)}$ wvhp. By Part 2 of Lemma 3.11, $s_i(t) - r_i(t+1)$ is at least $s_i(t)/20$. Thus, in either case, since $s_i(t)$ is at least $(a_{10} - a_3)np_i$, if a_{10} is chosen sufficiently larger than π_3, $s_i(t+1)$ is at most $a_{10}np_i$.

Claim (ii) follows from claim (i) and Lemma 3.9. We now prove claim (iii). Since $s_i(t)$ is at most $a_{10}np_i$ and $b_{d_i(t)}$ is at least $a_{11}np_i$, by Part (iii) of Lemma 3.10, $s_i(t) - r_i(t+1)$ is at least $a_{11}\pi_3 s_i(t)/(120a_{10})$ wvhp. Therefore, the total number of new requests is at least $a_{11}\pi_3 n/(120a_{10})$. By a Chernoff bound, the number of new requests for each A_i in $\mathcal{A}_{>0}$ is at least $a_{12}np_i$ wvhp for a suitable choice of a_{12}.

We now prove claim (iv). We need to show that $s(A_i, j, t+1)$ is complete for all j such that b_j is at most $2a_{11}np_i$. Fix an index j such that b_j is at most $2a_{11}np_i$. By the induction hypothesis, $s_i(t)$ is at least $a_{12}np_i$, and $b_{d_i(t)}$ is at least $a_{11}np_i$. Part (ii) of Lemma 3.10 is applicable only if a_{12} were at least $4a_{11}\pi_4$. Such a choice of a_{11} and a_{12} is not always possible. Therefore, instead of considering new copies made in $B_j(A_i)$ in round t only (as is done in the Part (ii) of Lemma 3.10), we consider copies made in rounds $t - a_9$ through t, where a_9 is a sufficiently large constant. The proof of claim (iv) is as follows. If $t \leq t_0 + a_9$, then since the cache is assumed to hold copies created in rounds 0 through $t_0 + a_9$, as in the induction basis, it follows that $b_{d_i(t)}$ is at least $a_{11}np_i$. If $t > t_0 + a_9$, the induction hypothesis implies that for $t - a_9 \leq t' \leq t$, $s_i(t')$ is at least $a_{12}np_i$. By Lemma 3.7, if a_9 is chosen sufficiently large, the number of new copies created in $B_j(A_i)$ in rounds $t - a_9$ through t is at least $9b_j/10$. Thus, $b_{d_i(t+1)}$ is at least $a_{11}np_i$. $\square$

Proof of Theorem 3.1: It follows from Lemmas 3.20 and 3.17, and Part (iv) of Lemma 3.10 that each round is nice. Since each round is nice, the number of access requests satisfied in each round is $\Omega(n)$, thus establishing the bound on the number of access requests. The bound on per-request communication follows from the protocol definition. $\square$

3.6.5 The Dynamic Model

The proof for the dynamic model follows easily from the properties of the protocol derived in Sections 3.6.2 and 3.6.3. Recall that in the dynamic model, an adversary assigns new requests to free clients subject to the following constraints for all i in $[m]$: (i) $q_i(0) = O(\log n)$, and (ii) for $t > 0$, $q_i(t) \leq (21/20) \max\{q_i(t') : \max\{0, t - \Gamma\} \leq t' < t\}$.

Lemma 3.21: *For all i in $[m]$ and $0 \leq t \leq \text{poly}(n)$, we have wvhp:*

1. *$s_i(t)$ is at most $\sum_{0 \leq j \leq t} (19/20)^{(t-j)} q_i(j)$,*

2. *Round t is good, and*

3. *$s_i(t)$ is at most $8\pi_1 b_{d_i(t)}$.*

Proof: Fix an i in $[m]$. We first show that Part 1 implies Part 2. Let x_i denote $\sum_{0 \leq j \leq t} (19/20)^{t-j} q_i(j)$. For all i and t', $q_i(t')$ is chosen independent of the hash functions. Moreover, $\sum_{0 \leq i < m} q_i(t')$ is at most n for all t'. Therefore, $\sum_{0 \leq i < m} x_i$ is $O(n)$. It thus follows from Lemma 3.9 that if Part 1 holds, then round t is good wvhp.

The proof of the lemma is by induction on t. For the base case, we let t equal 0. Since $s_i(0)$ equals $q_i(0)$, Part 1 holds, thus also implying Part 2. Since $q_i(0)$ is $O(\log n)$ and $d_i(t)$ is at least 0, Part 3 holds.

For the induction step, we assume that the statement of the lemma holds for all rounds before round t. Therefore, $s_i(t)$ is at most $8\pi_1 b_{d_i(t)}$. By Part 2 of the induction hypothesis, round t is good. Since $8\pi_1$ is at most $\pi_3/6$, it follows from Part 2 of Lemma 3.11 that $r_i(t+1)$ is at most $19 s_i(t)/20$. We thus have:

$$
\begin{aligned}
s_i(t+1) &\leq q_i(t+1) + 19 s_i(t)/20 \\
&\leq q_i(t+1) + (19/20) \sum_{0 \leq j \leq t} (19/20)^{t-j} q_i(j) \\
&\leq \sum_{0 \leq j \leq t+1} (19/20)^{t+1-j} q_i(j),
\end{aligned}
$$

which establishes Part 1 of the induction step. (The second equation follows from Part 1 of the induction hypothesis.) Part 2 of the induction step is implied by Part 1.

For Part 3 of the induction step, we first show that for any t' in $[\max\{0, t-\Gamma\}, t]$, $s_i(t')$ is at most $4\pi_1 b_{d_i(t+1)}$. By Part 1 of the induction hypothesis, $s_i(t')$ is at most $8\pi_1 b_{d_i(t')}$. Let j be the largest integer such that $s_i(t')$ is at least $4\pi_1 b_j$. It follows from the definition of j that $d_i(t')$ is at least j. By Part 2 of the induction hypothesis, round t is good. Hence, by Part 3 of Lemma 3.11, $d_i(t+1)$ is at least $j+1$. Therefore, $s_i(t') < 4\pi_1 b_{d_i(t+1)}$.

After round t, we have:

$$\begin{aligned}
s_i(t+1) &\leq (19/20)s_i(t) + q_i(t+1) \\
&\leq (19/20)s_i(t) + (21/20)\max\{q_i(t') : \max\{0, t-\Theta(\Gamma)\} \leq t' < t\} \\
&< (19/20)4\pi_1 b_{d_i(t+1)} + (21/20)4\pi_1 b_{d_i(t+1)} \\
&\leq 8\pi_1 b_{d_i(t+1)},
\end{aligned}$$

thus completing the proof of the induction step. □

Proof of Theorem 3.2: It follows from Part (iv) of Lemma 3.10 and Parts 2 and 3 of Lemma 3.21 that each round is nice. Since each round is nice, the number of access requests satisfied in each round is $\Omega(n)$, thus establishing the bound on the number of access requests. The bound on per-request communication follows from the protocol definition. □

3.7 Write Operations

Thus far, we have focused our attention on read-only objects. In this section, we describe our algorithm for handling write operations. We consider two different approaches: the *write-and-update* and the *invalidate-and-write* protocols.

At a given time step, any number of clients may attempt to simultaneously initiate a write operation on some object A. Each client communicates with servers in block $B_0(A)$ only, where the primary copy of A is stored. The first part of each protocol consists of a simple three-round randomized leader election procedure to select one of these clients to actually write the object A. We introduce four new types of control messages: *write-req*, *write-may*, *write-try*, and *write-ok*. In the first round, each writer attempts a *write-req* message to each server in $B_0(A)$. For a server S in $B_0(A)$, let $Q(S)$ denote the set of clients whose *write-req* messages are received by S. If $Q(S)$ is non-empty, S sends a *write-may* control message to an arbitrary client in $Q(S)$. In the second round, each client that receives at least one *write-may* message attempts a *write-try* control message to each server in $B_0(A)$. Let $T(S)$ denote the set of clients whose *write-try* messages are received by S. Server S selects the client C in $T(S)$ with the largest id and sends a *write-ok* message to C. In the third and final round, the unique client that receives more than $b_0/2$ *write-ok* messages, writes A by sending the fragments of the new version of object A to $B_0(A)$. A time-stamp is sent along with each of these fragments so that future clients reading the fragments can differentiate old fragments from new ones.

The two protocols differ in the second part. In the write-and-update protocol, after the fragments are sent to block $B_0(A)$, updates are propagated to servers in higher-numbered blocks that hold copies of A, by the same method as is used to propagate copies. The write is assumed to "complete" before these updates are propagated. As a result, it is possible that a client reads an old version of an object. We use the following validation scheme to ensure that each client receives a version that is at most $O(\log n)$ steps out of date. A steady stream of validation time-stamps is created by servers in $B_0(A)$ and propagated to higher-numbered blocks that hold copies of A. Each server S that has a copy of A maintains a variable $b(S)$ that denotes the last validation time-stamp received by S. A server S satisfies a request in round t only if $b(S)$ is at least $t - O(\log n)$, thus ensuring that the version sent to a client is at most $O(\log n)$ steps old. Since the per-request communication due to the validation time-stamps is asymptotically smaller than that required by the rest of the protocol, the results of Section 3.5 hold as stated.

In the invalidate-and-write protocol, we maintain, for each object, a fault-tolerant distributed list of all servers and clients holding a copy of the object. When a write operation is performed, before updating the primary copy, the servers in block 0 participate in an invalidation scheme in which each client/server on the list is sent one or more invalidation messages wvhp. The main advantage of this extension is that clients can make use of locally-cached copies of objects since they are informed once such a copy becomes out of date. The main disadvantage is that it is not possible to guarantee in the worst case that these invalidation messages are all sent quickly (e.g., within $O(\log n)$ steps). The difficulty is that the lists can grow very long over time, and if a large number of write operations are performed over a short period on a set of objects with long associated client/server lists, then it is simply not possible to send all of the invalidation messages quickly.

3.8 Concluding Remarks

In order to achieve fault-tolerance and space-efficiency, our protocol uses Rabin's IDA technique to encode each object as a set of fragments such that only a constant fraction of the fragments are needed to reconstruct an object. One shortcoming of IDA is that it does not tolerate errors in the fragments. Suppose, for example, that a client reading an object receives a large number of fragments, each of which is noisy (i.e., contains arbitrarily many errors) with some constant probability $\varepsilon > 0$. Unless the noisy fragments can be easily identified as such, the client cannot efficiently reconstruct the object using IDA. In such a noisy setting, it would be worthwhile to consider variants of our protocol based on the Berlekamp-Welch decoder [28] (see also [110, Appendix A]), which tolerates noise in a constant fraction of the fragments.

We would like to extend our protocols to other interesting models of distributed computation that incorporate asynchrony or locality information. We conjecture that, with suitably modified definitions and appropriate technical assumptions, the performance bounds of the present chapter can be extended to apply to models allowing limited forms of asynchrony (e.g., bounded asynchrony). To address the issue of locality, it would be interesting to consider a variant of our protocol in which the number of copies of an object that are created in any region of the network is proportional to its popularity within the region, and where regions are identified on the basis of some hierarchical decomposition of the network. Recent work on locality issues includes algorithms for allocating files on arbitrary networks [19, 26], protocols for accessing nearby objects on restricted classes of networks [74, 103, 117], and caching schemes for the Internet [67].

The most important technical problem left open in this chapter is to extend our analysis to more general models of access patterns. For example, we would like to study a model in which an adversary places an arbitrary sequence of requests at each node at the start of the computation. We conjecture that our protocol is provably efficient under the preceding model as well. We anticipate that the framework developed in Sections 3.6.2 and 3.6.3 will be useful in generalizing our results to other models.

Part II

Sharing Processors

Chapter 4

Static Load Balancing on Arbitrary Networks

4.1 Introduction

We begin our discussion about processor sharing in a distributed system by considering the problem of static load balancing. We model the distributed system by an undirected graph $G = (V, E)$ in which V is the set of nodes and E is the set of edges. Each node has an initial collection of tokens and no tokens are created or destroyed while the tokens are being balanced. We assume that in one unit of time, at most one token can be transmitted across an edge of the network in each direction. Figure 4.1 illustrates an instance of the static load balancing problem.

In this chapter, we analyze the performance of simple local load balancing algorithms for both single-port and multi-port models of computation. In the *single-port* model, a node may send and/or receive at most one token per unit time. In the *multi-port* model, a node may send and/or receive a token across all of its edges (there may be as many as d) per unit time.

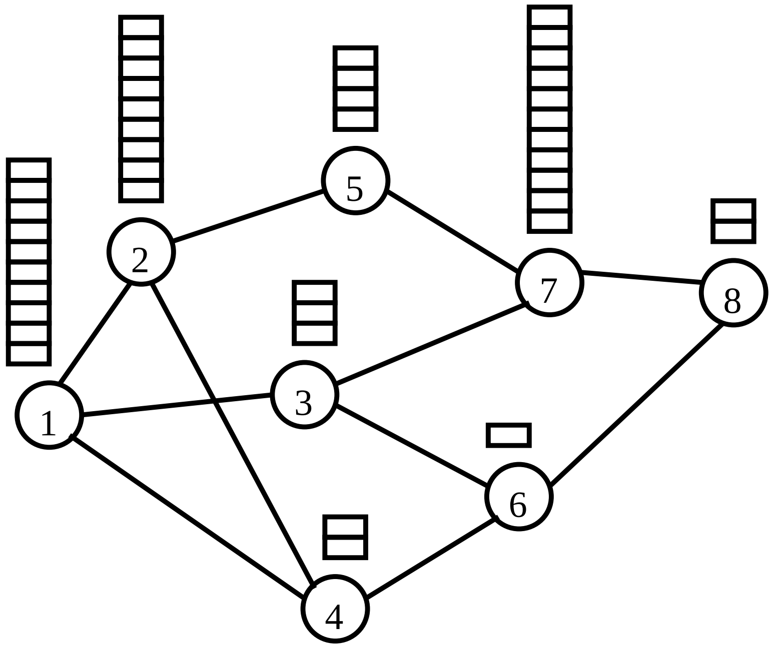


Figure 4.1: An instance of the static load balancing problem. Each small rectangle represents a token. For example, node 5 has 4 tokens.

4.1.1 The Single-Port and Multi-Port Algorithms

The single-port and multi-port algorithms are both based on the *local balancing* approach, in which each node exchanges load with a subset of its neighbors at each step. In the single-port algorithm, a matching is randomly chosen at each step. First, each (undirected) edge in the network is independently selected to be a *candidate* with probability $1/(4d)$. Then each candidate edge (u, v) for which there is another candidate edge (u, x) or (y, v) is removed from the set of candidates. The remaining candidates form a matching M in the network. For each edge (u, v) in M, if u and v have the same number of tokens, then no tokens are sent across (u, v). Otherwise, a token is sent from the node with more tokens to the node with fewer. Figure 4.2 illustrates one step of the single-port algorithm on the example given in Figure 4.1. This algorithm was first analyzed in [54]. (The single-port algorithm is sometimes referred to as the dimension-exchange method in the load balancing literature [119].) The multi-port algorithm is

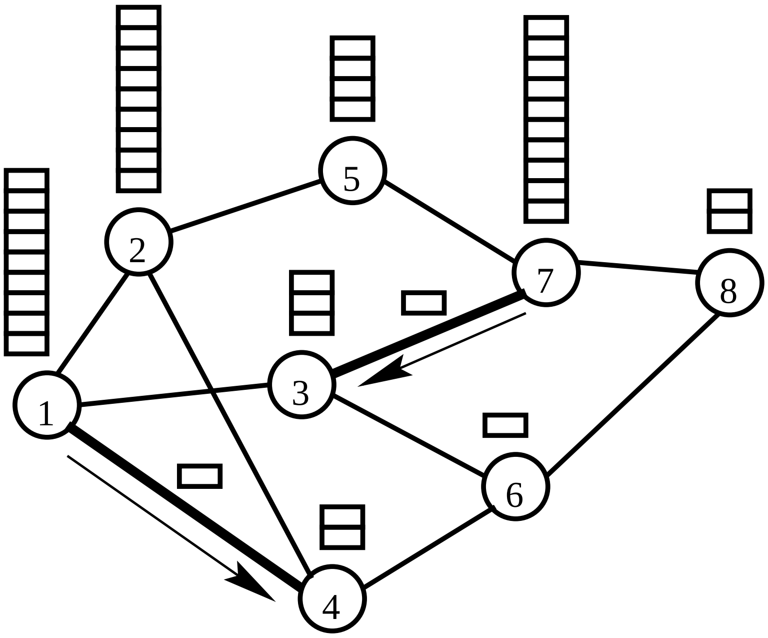


Figure 4.2: A step of the single-port algorithm on the example given in Figure 4.1. The bold edges are the edges that are chosen in the random matching.

simpler and deterministic. At each step, a token is sent from node u to node v across edge (u, v) if at the beginning of the step node u contains at least $2d + 1$ more tokens than node v. Figure 4.3 illustrates one step of the multi-port algorithm on the example given in Figure 4.1. This algorithm was first analyzed in [5]. (The multi-port algorithm is sometimes referred to as the diffusion method in the load balancing literature [119].) We characterize the performance of a load balancing algorithm by the time that it takes to balance the tokens, and by the final imbalance that it achieves. The *imbalance* is defined to be the maximum difference between the number of tokens at any node and the average number of tokens in the network. We say that an algorithm *balances to within t tokens* if the final imbalance is at most t.

4.1.2 Overview of the Results

We analyze the single-port and multi-port algorithms in terms of the initial imbalance, which we denote Δ, the number of nodes in the network, n, the maximum degree d of

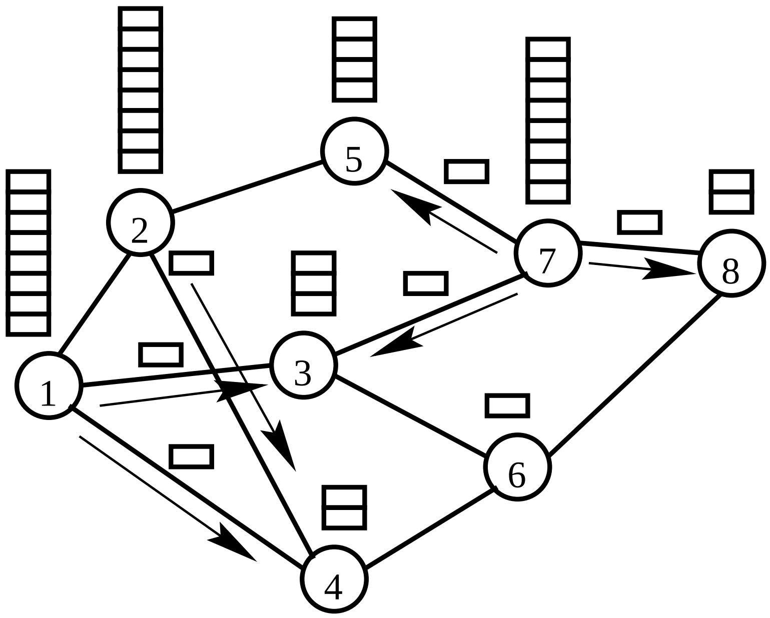


Figure 4.3: A step of the multi-port algorithm on the example given in Figure 4.1.

the network, and the node and edge expansions of the network. We define the *node expansion* μ of a network G to be the largest value such that every set S of $n/2$ or fewer nodes in G has at least $\mu|S|$ neighbors outside of S. We define the *edge expansion* α of a network G to be the largest value such that for every set S of $n/2$ or fewer nodes in G, there are at least $\alpha|S|$ edges in G with one endpoint in S and the other not in S. Consider the example in Figure 4.1. It is easy to see that $\Delta = 42/8$, $n = 8$, and $d = 3$. To determine the node and edge expansions, note that each set S of at most 4 nodes has at least $3|S|/4$ neighbors (resp., $3|S|/4$ edges) outside of S (resp., coming out of S). Moreover, the set $\{3, 6, 7, 8\}$ has 3 neighbors and 3 edges outside of S. Thus, μ and α are both $3/4$.

Our main results in this chapter are as follows:

- In Section 4.3.1, we show that the single-port algorithm balances any network to within $O(d \log n/\alpha)$ tokens in $O(d\Delta/\alpha)$ steps wvhp[1]. This bound is tight in the

[1]Recall that the term wvhp, which is defined in Section 1.3, means with probability $1 - n^{-c}$, where c is a constant that can be set arbitrarily large by appropriately adjusting other constants defined within the relevant context.

sense that for many values of n, d, α, and Δ, there is an n-node network with maximum degree d, edge expansion α, and an initial placement of tokens with imbalance Δ, where the time (for any algorithm) to balance to within even $\Delta/2$ tokens is at least $\Omega(d\Delta/\alpha)$. We also establish a bound on the single-port algorithm in terms of node expansion.

- As in the single-port case, we analyze the multi-port algorithm in terms of both edge expansion and node expansion. We show that the multi-port algorithm balances any network to within $O(d^2 \log n/\alpha)$ tokens in $O(\Delta/\alpha)$ steps. This bound is tight in the sense that for any network with edge expansion α, and any value Δ, there exists an initial distribution of tokens with imbalance Δ such that the time to reduce the imbalance to even $\Delta/2$ is $\Omega(\Delta/\alpha)$. Our analysis of the multi-port algorithm is contained in Section 4.3.2.

- Thus far we have described a network model in which the nodes are synchronized by a global clock (i.e., a *synchronous* network), and in which the edges are assumed not to fail. In Section 4.4, we consider dynamic and asynchronous networks, in which edges may fail and recover dynamically. We show that a minor variant of the multi-port algorithm achieves the same bounds as for the synchronous case, if the network satisfies the following constraint: at each time step, the set of live edges has edge expansion α, or node expansion μ.

- Finally, in Section 4.5, we study a centralized version of the static load balancing problem, in which every node has knowledge of the global state of the network. We prove that any network can be balanced by a centralized algorithm to within three tokens in at most $2\lceil(1+\mu)\Delta/\mu\rceil$ steps in the single-port model. Moreover, there exists a network and an initial token distribution for which any single-port algorithm takes more than $\lceil(1+\mu)\Delta/\mu\rceil$ steps to balance the network to within one token. Similarly, for the multi-port model, we show that that any network can be balanced to within $d+1$ tokens by a centralized algorithm in at most $2\lceil\Delta/\alpha\rceil$ steps, while there exists an initial token distribution such that any algorithm will take at least $\lceil \Delta/\alpha \rceil$ steps to balance the network to within one token.

4.1.3 Related Work

Load balancing has been studied extensively because it arises in a wide variety of settings including adaptive mesh partitioning [68, 118], fine-grain functional programming [59], job scheduling in operating systems [48, 83], and distributed tree searching [76, 86]. A number of models have been proposed for studying load balancing problems. These models can be classified on the basis of three characteristics: (i) centralized control (e.g., [93, 111]) versus distributed control (e.g., [30, Chapter 7] [42]), (ii) shared memory communication such as the PRAM model (e.g., [27]), uniform communication (e.g., [111]), or fixed-connection network communication (e.g., [5, 42]), and (iii) unbounded edge capacity (e.g., [30, Chapter 7] [109]) versus bounded edge capacity (e.g., [5, 101]) (the capacity of an edge is the maximum number of tokens it can transmit per step). In the discussion that follows, we restrict our attention to models of computation with the same basic characteristics as the model considered in this chapter, namely: distributed control, fixed-connection network communication, and bounded edge capacity.

Local algorithms restricted to particular networks have been studied on counting networks [15, 78], hypercubes [72, 101], and meshes [68, 93]. Another class of networks on which load balancing has been studied is the class of expanders. Peleg and Upfal [97] pioneered this study by identifying certain small-degree expanders as being suitable for load balancing. Their work was extended in [34, 69, 98]. These algorithms either use strong expanders to approximately balance the network, or the AKS sorting network [6] to perfectly balance the network. Thus, they do not work on networks of arbitrary topology. Also, these algorithms work by transferring load along fixed paths in the network and, therefore, cannot cope with the changes in the network topology. In contrast, our local algorithm works on any dynamic network that remains connected.

On arbitrary topologies, load balancing algorithms that use bounded edge capacity were first proposed and analyzed in [5] for the multi-port variant and in [54] for the single-port variant. The associated upper bounds are suboptimal by factors of $\Omega(\log(n\Delta))$ and $\Omega(\sqrt{n})$, respectively. We improve these results for both single-port and multi-port variants.

As remarked earlier, our multi-port results (and those in [5]) hold even for dynamic or asynchronous networks. In general, work on dynamic and asynchronous networks has been limited. In work related to load balancing, for instance, an end-to-end communication problem, namely one in which messages are routed from a single source to a single destination, has been studied in [4, 24] on dynamic networks. Our scenario is substantially more involved since we are required to move load between several sources and destinations simultaneously. Another result on dynamic networks is the recent analysis of a local algorithm for the approximate multicommodity flow problem [22, 23]. While their result has several applications including the end-to-end communication problem mentioned above, it does not seem to extend to load balancing. Our result on load balancing uses similar techniques; however, our algorithm and analysis are simpler and we obtain worst-case optimal bounds for our problem.

The convergence of local load balancing algorithms is related to that of random walks on Markov chains. Indeed the convergence bounds in both cases depend on the expansion properties of the underlying network and they are established using potential function arguments. There are however two important differences. First, the analysis of the rapid convergence of random walks [73, 95] relies on averaging arbitrary probabilities across any edge. This corresponds to sending an arbitrary (possibly nonintegral) load along an edge, which is forbidden in our model. In this sense, the analysis in [42] (and all references in the unbounded capacity model) are similar to the random walk analysis. Second, our argument uses an exponential potential function. The analyses in [42, 73, 95], in contrast, use quadratic potential functions. Our potential function and our amortized analysis appear to be necessary since a number of previous attempts using quadratic potential functions yielded suboptimal results [5, 54] for local load balancing.

4.2 Preliminaries

For any network $G = (V, E)$ with n nodes and edge expansion α, we denote the number of tokens at $v \in V$ by $w(v)$. We denote the average number of tokens by ρ, i.e., $\rho = (\sum_{v \in V} w(v))/n$. For simplicity, throughout this chapter we assume that ρ is an integer. We assign a unique *rank* from $[1, w(v)]$ to every token at v. The *height* of a token is its rank minus ρ. The height of a node is the maximum among the heights of all its tokens.

Consider a partition of V given by $\{S_i\}$, where the index i is any integer (positive, negative, or zero) and S_i may be empty for any i. Let $S_{>j}$ be $\cup_{i>j} S_i$. Similarly we define $S_{\geq j}$, $S_{<j}$, and $S_{\leq j}$. We define index i to be *good* if $|S_i| \leq \alpha |S_{>i}|/2d$. An index that is not good is called a *bad* index. Thus, index i is good if there are at least $\alpha|S_{>i}|/2$ edges from nodes in $S_{>i}$ to nodes in $S_{<i}$. To observe this, note that the number of edges out of $S_{>i}$ is at least $\alpha|S_{>i}|$. On the other hand, the number of edges coming out of S_i is at most $d|S_i|$ which is at most $\alpha|S_{>i}|/2$ if i is good. Therefore, at least $\alpha|S_{>i}|/2$ edges go from nodes in $S_{>i}$ to nodes in $S_{<i}$.

For any bad index i, it follows from the equality $|S_i| = |S_{>i-1}| - |S_{>i}|$ that $|S_{>i}| < |S_{>i-1}|/(1 + \alpha/(2d))$. Consider the reduction in $|S_{>i}|$ as i increases. For each bad index, there is a reduction by a factor of $1/(1 + \alpha/(2d))$. Hence, there can be at most $\lceil \log_{(1+\alpha/(2d))} n \rceil$ bad indices because $(1 + \alpha/(2d))^{\log_{(1+\alpha/(2d))} n} \geq n$. It follows that at least half of the indices in $[1, 2\lceil \log_{(1+\alpha/(2d))} n \rceil]$ are good.

4.3 Analysis for Static Synchronous Networks

4.3.1 The Single-Port model

In this section, we analyze the single-port load balancing algorithm that is described in Section 4.1.2.

Theorem 4.1: *For an arbitrary network G with n nodes, maximum degree d, edge expansion α, and initial imbalance Δ, the single-port algorithm balances G to within $O((d \log n)/\alpha)$ tokens in $O((d\Delta)/\alpha)$ steps, wvhp.*

For the sake of analysis, before every step we partition the set of nodes according to how many tokens they contain. For every integer i, we denote the set of nodes having $\rho+i$ tokens as S_i. Consider the first T steps of the algorithm, with T to be specified later. It holds that either $|S_{>0}| \leq n/2$ at the start of at least half the steps, or $|S_{\leq 0}| \leq n/2$ at the start of at least half the steps. Without loss of generality, assume the former is true. Thus, every subset of nodes in $S_{>0}$ expands, and we will use this expansion property to show that the number of nodes that have at least $\rho + 2\log_{(1+\alpha/(2d))} n$ tokens rapidly goes to zero.

Recall that at least half of the indices in $[1, 2\lceil \log_{(1+\alpha/(2d))} n \rceil]$ are good in any time step. Therefore, there exists an index j in $[1, 2\lceil \log_{(1+\alpha/(2d))} n \rceil]$ that is good in at least half of those time steps in which $|S_{>0}| \leq n/2$. Hence j is good in at least $T/4$ steps.

With every token at height x we associate a potential of $\phi(x)$, where $\phi : N \rightarrow R$ is defined as follows:

$$\phi(x) = \begin{cases} 0 & \text{if } x \leq j, \\ (1+\nu)^x & \text{otherwise,} \end{cases} \tag{4.1}$$

where $\nu = \alpha/(cd)$, and $c > 1$ is a real constant to be specified later. The potential of the network is the sum of the potentials of all tokens in the network. While transmitting a token, every node sends its token with maximum height. Similarly, any token arriving at a node with height h is assigned height $h + 1$. It follows from the definition of the potential function, and the fact that the height of a token never increases, that the potential of the network never increases. In the following, we show that during any step when j is good, the expected decrease in the potential of the network is at least an $\varepsilon\nu^2$ fraction of the potential before the step, where $\varepsilon > 0$ is a real constant to be specified later.

Before proving Theorem 4.1, we present an informal outline of the proof. For simplicity, let us assume that G is a constant-degree expander, i.e., $d = O(1)$ and $\mu = \Omega(1)$. Consider the scenario in which all of the indices greater than j are bad. In this situation, for indices greater than j, the size of the set $S_{\geq i}$ decreases exponentially with increasing i, and hence the number of tokens with height i decreases exponentially with increasing i. If the rate of growth of $\phi(x)$ with increasing x is smaller than the rate of decrease of $|S_{\geq i}|$ with increasing i, then the total potential due to tokens at height i "dominates" the total potential due to tokens at height greater than i. In such a case the potential of $S_{>j}$ is essentially a constant times the potential of tokens at height $j+1$. In addition, if the potential of tokens at height at most j is zero, then in every step when j is good, there is a constant fraction potential drop, because a constant fraction of the nodes in $S_{>j}$ send tokens to $S_{<j}$ in such a step. The exponential function we have defined in Equation (4.1) satisfies the properties described above for c sufficiently large.

In general, the indices greater than j may form any sequence of good and bad indices, provided that the upper bound on the number of bad indices is respected. We consider the indices greater than j in reverse order and show by an amortized analysis that for each index i we can "view" all indices greater than or equal to i as bad. If i is bad, then this view is trivially preserved; otherwise, the number of edges from $S_{>i}$ to $S_{<i}$ is at least $\alpha|S_{>i}|/2$ and hence there is a significant potential drop across the cut $(S_{\leq i}, S_{>i})$. This drop can be used to rearrange the potential of $S_{>i}$ in order to maintain the view that all indices greater than i are bad. We then invoke the argument for the case in which all indices greater than j are bad, and complete the proof.

Consider step t of the algorithm. Let Φ_t denote the potential of the network after step $t > 0$. Let M_i be the set of tokens that are sent from a node in $S_{>i}$ to a node in $S_{<i}$. Note that a token may appear in several different sets M_i. Let $m_i = |M_i|$. We say that a token p has an *i-drop* of $\phi(i+1) - \phi(i)$ if p moves from a node in $S_{>i}$ to a node in $S_{<i}$. Thus, the potential drop due to a token moving on an edge from node $u \in S_i$ to node $v \in S_{i'}$, $i > i' + 1$, can be expressed as the sum of k-drops for $i' < k < i$. In Lemma 4.1, we use this notion of i-drops to relate the total potential drop in step t, Ψ, to the m_i's.

Lemma 4.1:

$$\Psi = \left(\sum_{i>j} m_i \nu(1+\nu)^i\right) + m_j(1+\nu)^{j+1}.$$

Proof: Let M be the set of tokens that are moved from a node in $S_{>j}$. (Note that tokens that start from and end at nodes in $S_{>j}$ also belong to M.) For any token p, let $a(p)$ (resp., $b(p)$) be the height of p after (resp., before) step t.

$$\begin{aligned}
\Psi &= \sum_{p \in M} (\phi(b(p)) - \phi(a(p))) \\
&= \sum_{p \in M} \sum_{a(p) \le i < b(p)} (\phi(i+1) - \phi(i)) \\
&= \sum_{i \ge j} \sum_{p \in M_i} (\phi(i+1) - \phi(i)) \\
&= \left(\sum_{i > j} \sum_{p \in M_i} (\phi(i+1) - \phi(i)) \right) + \sum_{p \in M_j} (\phi(j+1) - \phi(j)) \\
&= \left(\sum_{i > j} \sum_{p \in M_i} \nu(1+\nu)^i \right) + \sum_{p \in M_j} (1+\nu)^{j+1} \\
&= \left(\sum_{i > j} m_i \nu(1+\nu)^i \right) + m_j(1+\nu)^{j+1}.
\end{aligned}$$

(The second equation holds since the sum of $\phi(i+1) - \phi(i)$ over i telescopes. For the third equation, we interchange the order of summation and use the fact that $\phi(i)$ is zero for all $i \le j$. The fourth equation is obtained by separating the case $i \ge j$ into two cases $i > j$ and $i = j$. For deriving the fifth equation, we use: (i) for all $i > j$, $\phi(i+1) - \phi(i) = \nu(1+\nu)^i$, and (ii) $\phi(j) = 0$. The last equation follows from the definition of m_i.) □

We now describe the amortized analysis, which we alluded to earlier in this section, that we use to prove Theorem 4.1. We associate a charge of $\varepsilon\nu^2\phi(h)$ with each token at height h. We show that we can pay for all of the charges using the expected potential drop $E[\Psi]$, which implies a lower bound on $E[\Psi]$. We consider the indices in $[j+1, \ell]$ in reverse order, where ℓ is the maximum token height. For every i in $[j, \ell]$, we maintain a "debt" term, given by Γ_i below, which is the difference between the charges due to tokens at height greater than i and the sum of i'-drops for $i' > i$. We will place an upper bound on $E[\Gamma_i]$ that lets us view all of the indices in $[i+1, \ell]$ as bad indices. In other words, we upper bound $E[\Gamma_i]$ by $\varepsilon\nu|S_{\geq i}|(1+\nu)^i$. It follows from this upper bound and the informal argument outlined earlier in this section that the expected total debt can be paid for by the expected drop across index j.

Formally, for any $i > j$, we define

$$\begin{aligned}\Psi_i &= \sum_{k \geq i} m_k \nu (1+\nu)^k, \text{ and} \\ \Gamma_i &= (\varepsilon\nu^2)\left(\sum_{p: b(p) \geq i} (1+\nu)^{b(p)}\right) - \Psi_i.\end{aligned}$$

We also define

$$\Gamma = (\varepsilon\nu^2)\left(\sum_{p: b(p) > j} (1+\nu)^{b(p)}\right) - \Psi.$$

Note that $\Phi_{t-1} = \sum_{p: b(p) > j} (1+\nu)^{b(p)}$ is the total potential of $S_{>j}$ prior to step t.

In order to prove the upper bound on $E[\Gamma_i]$, we place a lower bound on $E[m_i]$ that is obtained from the following lemma of [54].

Lemma 4.2 ([54]): *For any edge $e \in E$, the probability that e is selected in the matching is at least $1/(8d)$.* □

Lemma 4.3: *There exists a real constant $\varepsilon > 0$ such that for all $i > j$, we have $E[\Gamma_i] \leq (\varepsilon\nu)|S_{\geq i}|(1+\nu)^i$.*

Proof: The proof is by reverse induction on i. If $i > \ell$, then the claim holds trivially since Γ_i and $|S_{\geq i}|$ are both equal to zero. (Recall that ℓ denotes the maximum token height.) Therefore, for the base case we consider $i = \ell$. Since $m_\ell = 0$, we have $\Psi_\ell = 0$. Thus, $\Gamma_\ell = (\varepsilon\nu^2)|S_\ell|(1+\nu)^\ell \leq (\varepsilon\nu)|S_{\geq \ell}|(1+\nu)^\ell$, since $\nu = \alpha/(cd) \leq 1/c \leq 1$ by our choice of c.

For the induction step we consider two cases, depending on whether i is good or bad. We begin with the case when i is good. By the definition of a good index, we have $|S_i| \leq \alpha|S_{>i}|/2d$. Since each node has at most d adjacent edges, there are at most $\alpha|S_{>i}|/2$ edges adjacent to nodes in S_i. Therefore, there are at most $\alpha|S_{>i}|/2$ edges from $S_{>i}$ to S_i. By the expansion property of the graph, $S_{<i}$ has at least $\alpha|S_{<i}|$ edges to nodes in $S_{\geq i}$, so there are at least $\alpha|S_{>i}|/2$ edges from $S_{>i}$ to $S_{<i}$. By Lemma 4.2, we have $E[m_i] \geq \alpha|S_{>i}|/(16d)$.

We are now ready to place a bound on $E[\Gamma_i]$. By definition, Γ_i can be calculated by subtracting the sum of i-drops from Γ_{i+1} and adding the charges due to tokens at height i. Therefore, we have:

$$
\begin{aligned}
E[\Gamma_i] &= E[\Gamma_{i+1}] + (\varepsilon\nu^2)|S_{\geq i}|(1+\nu)^i - E[m_i]\nu(1+\nu)^i \\
&\leq E[\Gamma_{i+1}] + (\varepsilon\nu^2)|S_{\geq i}|(1+\nu)^i - c\nu^2|S_{>i}|(1+\nu)^i/16 \\
&\leq E[\Gamma_{i+1}] - (\nu^2)|S_{\geq i}|(1+\nu)^i(f(c,\alpha,d) - \varepsilon) \\
&\leq (\varepsilon\nu)|S_{>i}|(1+\nu)^{i+1} - (\nu^2)|S_{\geq i}|(1+\nu)^i(f(c,\alpha,d) - \varepsilon) \\
&\leq (\varepsilon\nu)|S_{\geq i}|(1+\nu)^i((1+\nu) - \nu(f(c,\alpha,d) - \varepsilon)/\varepsilon),
\end{aligned}
$$

where $f(c,\alpha,d) = c/(16(1+\alpha/(2d)))$. (The first equation holds since the number of tokens p such that $b(p) = i$ is $|S_{\geq i}|$. The second equation follows from the lower bound on $E[m_i]$. The third equation holds since $|S_{>i}| \geq |S_{\geq i}|/(1+\alpha/(2d))$ whenever i is a good index. The fourth equation follows from the induction hypothesis. The last equation follows from the inequality $|S_{>i}| \leq |S_{\geq i}|$.)

The second case is when i is bad. Thus $|S_i| > \alpha|S_{>i}|/(2d)$. We now place an upper bound on $E[\Gamma_i]$ as follows.

$$
\begin{aligned}
E[\Gamma_i] &\leq E[\Gamma_{i+1}] + (\varepsilon\nu^2)|S_{\geq i}|(1+\nu)^i \\
&\leq (\varepsilon\nu)|S_{>i}|(1+\nu)^{i+1} + (\varepsilon\nu^2)|S_{\geq i}|(1+\nu)^i \\
&\leq (\varepsilon\nu)|S_{\geq i}|(1+\nu)^i((1+\nu)/(1+c\nu/2) + \nu).
\end{aligned}
$$

(The first equation holds since the number of tokens p such that $b(p) = i$ is $|S_{\geq i}|$. The second equation follows from the induction hypothesis. The third equation holds since $|S_{\geq i}| > (1+\alpha/(2d))|S_{>i}|$ whenever i is a bad index.)

We now complete the induction step by determining values for c and ε such that the following equations hold.

$$((1+\nu) - \nu(f(c,\alpha,d) - \varepsilon)/\varepsilon) \leq 1, \text{ and} \tag{4.2}$$

$$(1+\nu)/(1+c\nu/2)+\nu \leq 1 \tag{4.3}$$

We set c to be any constant greater than or equal to $(\alpha/d)+4$ (e.g., $c=5$). For this choice of c, $\nu = \alpha/(cd) \leq (c-4)/c$, and hence $2\nu + c\nu^2/2 \leq c\nu/2$. Therefore, we have:

$$\begin{aligned}
(1+\nu)/(1+c\nu/2)+\nu &= (1+2\nu+c\nu^2/2)/(1+c\nu/2) \\
&\leq (1+c\nu/2)/(1+c\nu/2) \\
&= 1.
\end{aligned}$$

Thus, Equation (4.3) is satisfied. Since $\alpha \leq d$, we find that $f(c,\alpha,d) \geq c/24$. We now set $\varepsilon = c/48$ to establish Equation (4.2). (For example, $c=5$ and $\varepsilon = 5/48$.) $\square$

We are now in a position to bound $E[\Gamma]$ on those steps in which j is good. By applying Lemma 4.3 with $i=j+1$, we obtain that $E[\Gamma_{j+1}] \leq (\varepsilon\nu)|S_{\geq j+1}|(1+\nu)^{j+1}$. If j is good, then by the definitions of Γ, Γ_{j+1}, and Ψ, we have:

$$\begin{aligned}
E[\Gamma] &= E[\Gamma_{j+1}] - E[m_j](1+\nu)^{j+1} \\
&\leq E[\Gamma_{j+1}] - \alpha|S_{>j}|(1+\nu)^{j+1}/(16d) \\
&\leq (\varepsilon\nu)|S_{>j}|(1+\nu)^{j+1} - \alpha|S_{>j}|(1+\nu)^{j+1}/(16d) \\
&= \nu|S_{>j}|(1+\nu)^{j+1}(\varepsilon - c/16) \\
&\leq 0,
\end{aligned}$$

(The second equation follows from the inequality $E[m_j] \geq \alpha|S_{>j}|/16d$ which holds whenever j is good. The third equation follows from the upper bound on $E[\Gamma_{j+1}]$. The fifth equation holds since $c/16 \geq \varepsilon$.)

We now derive a lower bound on the expected drop in the potential of the network during a sequence of T steps. By the definitions of Ψ and Γ, we have $\Phi_t = \Phi_{t-1} - \Psi$ and $\Gamma = \varepsilon\nu^2\Phi_{t-1} - \Psi$. If j is good during step t, we have $E[\Gamma] \leq 0$, and therefore, $E[\Phi_t] \leq \Phi_{t-1}(1-\varepsilon\nu^2)$, where the expectation is taken over the random matching selected in step t. Since j is good in at least $T/4$ steps, we obtain that $E[\Phi_{t+T}] \leq \Phi_t(1-\varepsilon\nu^2)^{T/4}$, where the expectation is over all the random matchings in the T steps. By setting

$T = \lceil (4 \ln 4)/(\varepsilon \nu^2) \rceil$, we obtain $E[\Phi_{t+T}] \le \Phi_t/4$. By Markov's inequality, the probability that $\Phi_{t+T} \ge \Phi_t/2$ is at most $1/2$. Therefore, using standard Chernoff bounds [37], we can show that in $T' = 8aT\lceil(\log \Phi_0 + \log n)\rceil$ steps, $\Phi_{T'} > 1$ with probability at most $O(1/(\Phi_0)^a + 1/n^a)$ for any constant $a > 0$.

If Δ is at most $2 \log_{(1+\alpha/(2d))} n$, then the claim of the theorem holds trivially. Accordingly, we assume that Δ is greater than $2 \log_{(1+\alpha/(2d))} n$ in what follows. Since Φ_0 is at least $(1+\nu)^{\Delta}$, Φ_0 is at least $n^{2/c}$. Therefore, $1/(\Phi_0)^a$ is inverse-polynomial in n. Since $\Phi_0 \le n(1+\nu)^{\Delta+1}/\nu$, we have $\log \Phi_0 \le (\Delta+1)(\nu) + \log n - \log \nu$. Therefore, for $T' = O(\Delta d/\alpha + d^2 \log n/\alpha^2)$, we have $\Phi_{T'} < 1$ wvhp which implies that after T' steps, $|S_{>2\log_{(1+\alpha/(2d))} n}| = 0$ wvhp.

To establish an upper bound on the imbalance in the number of tokens below the average, we use an averaging argument to show that after T' steps $|S_{<-2\log_{(1+\alpha/(2d))} n}| \le n/2$ wvhp, and then repeat the above arguments the potential redefined appropriately. This proves Theorem 4.1.

4.3.2 The Multi-Port Model

In this section, we analyze the deterministic multi-port algorithm described in Section 4.1.2.

Theorem 4.2: *For an arbitrary network G with n nodes, maximum degree d, edge expansion α, and initial imbalance Δ, the multi-port algorithm balances G to within $O((d^2 \log n)/\alpha)$ tokens in $O(\Delta/\alpha)$ steps.*

The proof of Theorem 4.2 is similar to that of Theorem 4.1. We assign a potential to every token, where the potential is exponential in the height of the token. We then show by means of an amortized analysis that a suitable rearrangement of the potential reduces every instance of the problem to a special instance that we understand well.

For the sake of analysis, before every step we partition the set of nodes according to how many tokens they contain. For every integer i, we denote the set of nodes having between $\rho - d + 2id$ and $\rho + d - 1 + 2id$ tokens as S_i. (Recall that ρ is the average number of tokens per node.) Consider the first T steps of the algorithm, with T to be specified later. Without loss of generality, we assume that $|S_{>0}| \leq n/2$ holds in at least half of these steps. As shown in Section 4.2, there exists an index j in $[1, 2\lceil\log_{(1+\alpha/(2d))} n\rceil]$ that is good in at least half of those steps in which $|S_{>0}| \leq n/2$. Hence in T steps of the algorithm, j is good in at least $T/4$ steps.

With every token at height h we associate a potential of $\phi(h)$, where $\phi : N \to R$ is defined as follows:

$$\phi(x) = \begin{cases} 0 & \text{if } x \leq 2jd, \\ (1+\nu)^x & \text{otherwise,} \end{cases}$$

where $\nu = \alpha/(cd^2)$, and $c > 0$ is a constant to be specified later. The potential of the network is the sum of the potentials of all tokens in the network.

While transmitting some number, say m, of tokens in a particular step, a node sends the m highest-ranked tokens. Similarly, if m tokens arrive at a node during a step, they are assigned the m highest ranks within the node. Thus, tokens that do not move retain their ranks after the step. We now describe what specific ranks we assign to tokens that move during any step t. Let u be a node in $S_{<i}$ with height h at the start of step t. Let A (resp., B) be the set of tokens that u receives from nodes in $S_{>i}$ (resp., $S_{\leq i}$). We assign new ranks to tokens in A and B such that the rank of every token in A is less than that of every token in B. Let C be the set of tokens in A that attain height at most $h + (d/2)$ after the step. Since $|A| \leq d$, by the choice of our ranking, we have $|C| \geq |A|/2$. We call C the set of *primary* tokens. We also note that for any node v with height h all tokens leaving v during a step are at height at least $h - d + 1$ prior to the step.

It follows from the definition of the potential function and the fact that the height of a token never increases that the network potential never increases. In the following we show that whenever j is good the potential of $S_{>j}$ decreases by a factor of $\varepsilon\nu^2 d^2$, where $\varepsilon > 0$ is a real constant to be specified later. (For the sake of simplicity, we assume that d is even. If d is odd, we can replace d by $d+1$ in our argument without affecting the bounds by more than constant factors.)

For any token p, let $a(p)$ (resp., $b(p)$) be the index i such that S_i contains p after (resp., before) the step. (Note that the indexing is done prior to the step.) Let M_i be the set of primary tokens received by nodes in $S_{<i}$. Let $m_i = |M_i|$. Note that m_i is at least half the number of edges connecting nodes in $S_{<i}$ and nodes in $S_{>i}$. This is because a token is sent along every one of the edges connecting $S_{<i}$ and $S_{>i}$ and at least half the tokens received by any node in $S_{<i}$ from nodes in $S_{>i}$ are primary tokens. Lemma 4.4 establishes the relationship between the total potential drop Ψ in step t and the m_i's.

Lemma 4.4:

$$\Psi \geq \left(\frac{1}{2}\sum_{i>j} m_i \nu d(1+\nu)^{(2i-1)d}\right) + m_j(1+\nu)^{2jd+1}.$$

Proof: Let M be the set of primary tokens that are moved from nodes in $S_{>j}$. (Note that primary tokens that start from a node in $S_{>j}$ and end at a node in $S_{>j}$ are in M.) Let p be a token in M. By the definition of a primary token, the height of p prior to the step is at least $2b(p)d - 2d + 1$ and the height after the step is at most $2a(p)d + 3d/2$. Moreover, p belongs to M_i for all i such that $a(p) < i < b(p)$.

$$\begin{aligned}
\Psi &\geq \sum_{p\in M}[\phi(2b(p)d - 2d + 1) - \phi(2a(p)d + 3d/2)] \\
&\geq \sum_{p\in M}\sum_{a(p)<i<b(p)}[\phi(2(i+1)d - 2d + 1) - \phi(2(i-1)d + 3d/2)] \\
&= \sum_{i\geq j}\sum_{p\in M_i}[\phi(2(i+1)d - 2d + 1) - \phi(2(i-1)d + 3d/2)] \\
&= \sum_{i>j}\sum_{p\in M_i}[\phi(2(i+1)d - 2d + 1) - \phi(2(i-1)d + 3d/2)] \\
&\quad + \sum_{p\in M_j}[\phi(2(j+1)d - 2d + 1) - \phi(2(j-1)d + 3d/2)] \\
&\geq \left(\frac{1}{2}\sum_{i>j}\sum_{p\in M_i}\nu d(1+\nu)^{2id-d}\right) + \sum_{p\in M_j}(1+\nu)^{2jd+1} \\
&\geq \left(\frac{1}{2}\sum_{i>j} m_i\nu d(1+\nu)^{2id-d}\right) + m_j(1+\nu)^{2jd+1}.
\end{aligned}$$

(The first equation follows from the lower bound (resp., upper bound) on the height of a token p in M before (resp., after) the step. For the second equation, note that $2id - 2d + 1 \leq 2(i-1)d + 3d/2$. Therefore, $\phi(2id - 2d + 1) \leq \phi(2(i-1)d + 3d/2)$. The second equation now follows since the sum telescopes. The third equation is obtained by interchanging the sums and noting that $\phi(x)$ is 0 for $x \leq 2jd$. The fourth equation is obtained by partitioning M into the subsets $M \setminus M_j$ and M_j. The fifth equation is derived using the following calculations: (i) $\phi(2id+1) - \phi(2id-d/2) \geq ((1+\nu)^{d/2} - 1)(1+\nu)^{2id-d/2} \geq \nu d(1+\nu)^{2id-d}/2$, (ii) $\phi(2jd+1) = (1+\nu)^{2jd+1}$, and (iii) $\phi(2jd - d/2) = 0$. The last equation follows from the definition of m_i.) $\square$

We establish Theorem 4.2 by means of an amortized analysis similar to the one used in Section 4.3.1. We associate a charge of $\varepsilon\nu^2 d^2 \phi(h)$ with every token at height h. We show that we can pay for all of the charges using the potential drop Ψ and thus place a lower bound on Ψ. We consider the sets S_i in reverse order and maintain a "debt" term Γ_i for each i. Informally, Γ_i indicates the difference between the total charges due to tokens at height at least $2id - d$ and the current upper bound on the potential drop. Our amortized analysis terminates by showing that the total debt Γ is at most zero.

We now formally define Γ_i and Γ. For any token p, let $h(p)$ denote the height of p prior to the step. Thus $2b(p)d - d \leq h(p) \leq 2b(p)d + d - 1$. For $i > j$ and for a suitable constant $\varepsilon > 0$ to be specified later, we define

$$\begin{aligned} \Psi_i &= \frac{1}{2} \sum_{k \geq i} m_k \nu d (1+\nu)^{2kd-d}, \text{ and} \\ \Gamma_i &= (\varepsilon \nu^2 d^2) \left(\sum_{p:h(p) \geq 2id-d} (1+\nu)^{h(p)} \right) - \Psi_i. \end{aligned}$$

We also define

$$\Gamma = (\varepsilon \nu^2 d^2) \left(\sum_{p:h(p) > 2jd} (1+\nu)^{h(p)} \right) - \Psi.$$

For any step t', let $\Phi_{t'}$ denote the total potential after step t'. The total potential after step $t-1$, Φ_{t-1}, equals $\sum_{p:h(p)>2jd} (1+\nu)^{h(p)}$.

Lemma 4.5: *There exists a real constant $\delta > 0$ such that for all $i > j$, we have*

$$\Gamma_i \leq (\delta \nu d^2)|S_{\geq i}|(1+\nu)^{2id-d}.$$

Proof: The proof is by reverse induction on i. Let ℓ be the maximum token height. Consider first the case when $i > \lfloor(\ell+d)/2d\rfloor$. Since $2id - d > \ell$, there is no token with height at least $2id - d$. Hence $\Gamma_i \leq 0$ and $|S_{\geq i}| = 0$. Thus, the desired claim holds. We now consider $i = \lfloor(\ell+d)/2d\rfloor$. Since $\Psi_i = 0$, we have

$$\begin{aligned}
\Gamma_i &\leq (2\varepsilon\nu^2 d^3)|S_{\geq i}|(1+\nu)^{\ell} \\
&\leq (2\varepsilon\nu^2 d^3)|S_{\geq i}|(1+\nu)^{2d(i+1)-d} \\
&\leq (\delta\nu d^2)|S_{\geq i}|(1+\nu)^{2id-d}.
\end{aligned}$$

(The first equation holds because: (i) each node in S_i has at most $2d$ tokens with height at least $2id - d$, and (ii) $h(p) \leq \ell$ for each token p. The second equation follows from the fact that $\ell < 2(i+1)d - d$. The third equation is obtained by choosing δ and ε such that $\delta > 2\varepsilon\nu d(1+\nu)^{2d}$. Note that for c sufficiently large, $(1+\nu)^{2d}$ can be set to an arbitrarily small constant.)

For the induction step we consider two cases. If i is good, then $|S_i| \leq \alpha|S_{>i}|/(2d)$ and $m_i \geq \alpha|S_{>i}|/4$. Therefore, we have

$$\begin{aligned}
\Gamma_i &\leq \Gamma_{i+1} + (2\varepsilon\nu^2 d^3)|S_{\geq i}|(1+\nu)^{2id+d-1} - m_i\nu d(1+\nu)^{2id-d}/2 \\
&\leq \Gamma_{i+1} + (2\varepsilon\nu^2 d^3)|S_{\geq i}|(1+\nu)^{2id+d-1} - c\nu^2 d^3|S_{>i}|(1+\nu)^{2id-d}/8 \\
&\leq \Gamma_{i+1} - (\nu^2 d^3)|S_{\geq i}|(1+\nu)^{2id-d}(f(c,\alpha,d) - 2\varepsilon(1+\nu)^{2d}) \\
&\leq (\delta\nu d^2)|S_{>i}|(1+\nu)^{2(i+1)d-d} - (\nu^2 d^3)|S_{\geq i}|(1+\nu)^{2id-d}(f(c,\alpha,d) - 4\varepsilon) \\
&\leq (\delta\nu d^2)|S_{\geq i}|(1+\nu)^{2id-d}((1+\nu)^{2d} - \nu d(f(c,\alpha,d) - 4\varepsilon)/\delta),
\end{aligned}$$

where $f(c,\alpha,d) = c/(8(1+\alpha/(2d)))$. (The first equation holds because: (i) each node in S_i has at most $2d$ tokens with height at least $2id - d$, and (ii) $h(p) \leq 2id + d - 1$ for each token p that contributes to Γ_i and not to Γ_{i+1}. The third equation follows from the inequality $|S_{>i}| \geq |S_{\geq i}|/(1+\alpha/(2d))$. The fourth equation follows from the induction hypothesis and the inequality $(1+\nu)^{2d} \leq 2$ for c sufficiently large. The last equation is derived using straightforward algebra.)

The second case is when i is bad. Thus $|S_i| > \alpha|S_{>i}|/(2d)$. We have

$$\begin{aligned}
\Gamma_i &\leq \Gamma_{i+1} + (2\varepsilon\nu^2 d^3)|S_{\geq i}|(1+\nu)^{2id+d-1} \\
&\leq (\delta\nu d^2)|S_{>i}|(1+\nu)^{2(i+1)d-d} + 2\varepsilon\nu^2 d^3|S_{\geq i}|(1+\nu)^{2id+d-1} \\
&\leq (\delta\nu d^2)|S_{\geq i}|(1+\nu)^{2id-d}((1+\nu)^{2d}/(1+\alpha/(2d)) + 2\varepsilon\nu d(1+\nu)^{2d}/\delta).
\end{aligned}$$

We now set c, δ, and ε such that $c > 4$, $c/12 - 4\varepsilon \geq 4\delta$, and $c/4 - 2\varepsilon/\delta \geq 4$. (One set of choices is $c = 50$, $\delta = 1$, and $\varepsilon = 1/24$.) Since $\alpha \leq d$, we have $f(c,\alpha,d) \geq c/12$. Since $c > 4$, we have $2\nu d < 1/2$, and hence $(1+\nu)^{2d} \leq 1 + \sum_{i>0}(2\nu d)^i = 1 + 2\nu d/(1-2\nu d) \leq 1 + 4\nu d$. Thus,

$$((1+\nu)^{2d} - \nu d(f(c,\alpha,d) - 4\varepsilon)/\delta) \leq 1 + 4\nu d - 4\nu d \leq 1.$$

Since $\alpha/(2d) \leq 1/2$, we have $1/(1+\alpha/(2d)) \leq 1 - \alpha/2d + (\alpha/2d)^2 \leq 1 - \alpha/(2d) + \alpha/(4d) = 1 - \alpha/(4d)$, and hence,

$$\begin{aligned}
(1+\nu)^{2d}/(1+\alpha/(2d)) + 2\varepsilon\nu d(1+\nu)^{2d}/\delta &= (1+\nu)^{2d}(1/(1+\alpha/(2d)) + 2\varepsilon\nu d/\delta) \\
&\leq (1+\nu)^{2d}(1 - \alpha/4d + 2\varepsilon\nu d/\delta) \\
&= (1+\nu)^{2d}(1 - c\nu d/4 + 2\varepsilon\nu d/\delta) \\
&\leq (1+4\nu d)(1-4\nu d) < 1.
\end{aligned}$$

(The second equation follows from the upper bound on $1/(1+\alpha/(2d))$. The fourth equation follows from the upper bound of $(1+4\nu d)$ on $(1+\nu)^{2d}$.)

Thus, in both cases, $\Gamma_i \leq (\delta\nu d^2)|S_{\geq i}|(1+\nu)^{2id-d}$. This completes the induction step. $\square$

Corollary 4.5.1: *If j is good on step t then we have $\Psi \geq \varepsilon\nu^2 d^2 \Phi_{t-1}$.*

Proof: Applying Lemma 4.5 with $i = j+1$, it follows that $\Gamma_{j+1} \leq (\delta\nu d^2)|S_{\geq j+1}|(1+\nu)^{2(j+1)d-d}$. If j is good then $|S_{\geq j}| \leq (1+\alpha/(2d))|S_{>j}| \leq 3|S_{>j}|/2$ and $m_j \geq \alpha|S_{>j}|/2$.

Therefore,

$$\begin{aligned}
\Gamma &\leq \Gamma_{j+1} + \varepsilon\nu^2 d^3 |S_{\geq j}|(1+\nu)^{2jd+d-1} - \alpha|S_{>j}|(1+\nu)^{2jd+1}/2 \\
&\leq (\delta\nu d^2)|S_{>j}|(1+\nu)^{2(j+1)d-d} + (3\varepsilon\nu^2 d^3)|S_{>j}|(1+\nu)^{2jd+d-1}/2 - \frac{\alpha}{2}|S_{>j}|(1+\nu)^{2jd+1} \\
&\leq (\nu d^2)|S_{>j}|(1+\nu)^{2(j+1)d-d}(\delta + 3\varepsilon\alpha/(2cd) - c/4) \\
&\leq 0,
\end{aligned}$$

for c, δ, and ε chosen above. (In the first equation, the term $\varepsilon\nu^2 d^3 |S_{\geq j}|(1+\nu)^{2jd+d-1}$ is an upper bound on the contribution to Γ_j by tokens in $S_{\geq j}$ since: (i) tokens with height at least $2jd + d$ contribute to Γ_{j+1}, and (ii) each node in $S_{\geq j}$ has $d - 2 \leq d$ tokens with height in the interval $[2jd + 1, 2jd + d - 1]$. Also, the third term in the first equation is the second term in the right-hand side of the inequality of Lemma 4.4. In the second equation, we use the upper bounds on Γ_{j+1} and $|S_{\geq j}|$. The third equation follows from the choice of c, δ, and ε, and the observation that for $c > 4$, we have $(1+\nu)^d \leq (1+\alpha/(cd^2))^d \leq (1+1/(cd))^d < (1+1/(4d))^d \leq e^{1/4} \leq 2$.)

By the definitions of Γ and Ψ, we have $\Phi_t \leq \Phi_{t-1} - \Psi$ and $\Gamma = \varepsilon\nu^2 d^2 \Phi_{t-1} - \Psi$. If j is good during step t, then $\Gamma \leq 0$, and the desired claim follows. ∎

By Corollary 4.5.1, if j is good during step t then we have

$$\Phi_t \leq \Phi_{t-1}(1 - \varepsilon\nu^2 d^2).$$

After $T = \lceil 4 \ln \Phi_0 / (\varepsilon\nu^2 d^2) \rceil$ steps, we have $\Phi_T \leq \Phi_0(1-\varepsilon\nu^2 d^2)^{T/4} < 1$. Since the height of each node is at most Δ initially, $\Phi_0 \leq n \sum_{2jd<i\leq\Delta}(1+\nu)^i \leq n(1+\nu)^{\Delta+1}/\nu$, $\ln \Phi_0 = O(\Delta\nu + \log n)$. Substituting $\alpha/(cd^2)$ for ν, we obtain that within $O(\Delta/\alpha + d^2 \ln n/\alpha^2)$ steps, $|S_{>2\log_{(1+\alpha/(2d))} n}| \leq |S_{>j}| = 0$.

We use an averaging argument to show that after T steps, $|S_{<-2\log_{(1+\alpha/(2d))} n}| \leq n/2$. By redefining the potential function and repeating the above analysis in the other direction, we obtain that in another T steps $|S_{<-4\log_{(1+\alpha/(2d))} n}| = 0$. This completes the proof of Theorem 4.2.

4.3.3 Results in Terms of Node Expansion

The proofs of Theorems 4.1 and 4.2 can be easily modified to analyze the algorithm in terms of the node expansion μ of the graph instead of the edge expansion α. Recall that μ and α are related by the following inequalities: $\alpha/d \leq \mu \leq \alpha$. The primary modifications that need to be done to obtain bounds in terms of node expansion are to change the definition of a good index and to set ν appropriately. We call index i good if $|S_i| \leq \mu|S_{>i}|/2$. We set $\nu = \mu/c$ (resp., $\nu = \mu/(cd)$) for the single-port model (resp., multi-port model).

By an argument similar to the one used in Section 4.2, we obtain that the number of bad indices is at most $\lceil \log_{(1+\mu)} n \rceil$. (In fact, the argument in Section 4.2 uses α/d as a lower bound on μ.) This bound on the number of bad indices leads to an upper bound of $O((\log n)/\mu)$ (resp., $O(d(\log n)/\mu)$) on the final imbalance obtained by the single-port algorithm (resp., multi-port algorithm). For a bound on the number of steps, note that while deriving a bound on the potential drop in Sections 4.3.1 and 4.3.2, we use the edge expansion α to obtain a lower bound on the number of tokens leaving sets $S_{>i}$. Since the best lower bound on α in terms of node expansion is μ, our time bounds here are obtained by substituting μ for α in the time bounds of Theorems 4.1 and 4.2, respectively. We thus obtain Theorems 4.3 and 4.4. Finally, Corollary 4.3.1 (resp., Corollary 4.4.1) follows from Theorems 4.1 and 4.3 (resp., Theorems 4.2 and 4.4).

Theorem 4.3: *For an arbitrary network G with n nodes, maximum degree d, node expansion μ, and initial imbalance Δ, the single-port algorithm balances to within $O((\log n)/\mu)$ tokens in $O(d\Delta/\mu)$ steps wvhp.*

Corollary 4.3.1: *If $\Delta \geq (d \log n)/\mu$, the single-port algorithm balances to within $O(\log n/\mu)$ tokens in $O((d\Delta)/\alpha)$ steps wvhp. If $\Delta < (d \log n)/\mu$, the single-port algorithm balances to within $O(\log n/\mu)$ tokens in $O((d\Delta)/\mu)$ steps wvhp.*

Theorem 4.4: *For an arbitrary network G with n nodes, maximum degree d, node expansion μ, and initial imbalance Δ, the multi-port algorithm load balances to within*

$O((d \log n)/\mu)$ *tokens in* $O(\Delta/\mu)$ *steps.*

Corollary 4.4.1: *If* $\Delta \geq (d^2 \log n)/\mu$, *the multi-port algorithm balances to within* $O((d \log n)/\mu)$ *tokens in* $O(\Delta/\alpha)$ *steps. If* $\Delta < (d^2 \log n)/\mu$, *the multi-port algorithm balances to within* $O((d \log n)/\mu)$ *tokens in* $O(\Delta/\mu)$ *steps.*

4.4 Extension to Dynamic and Asynchronous Networks

In this section, we extend our results of Section 4.3.2 for the multi-port model to dynamic and asynchronous networks. We first prove that a variant of the local multi-port algorithm is optimal on dynamic synchronous networks in the same sense as for static synchronous networks. We then use a result of [5] that relates the dynamic synchronous and asynchronous models to extend our results to asynchronous networks.

In the dynamic synchronous model, the edges of the network may fail or succeed dynamically. An edge $e \in E$ is *live* during step t if e can transmit a message in each direction during step t. We assume that at each step each node knows which of its adjacent edges are live. The local load balancing algorithm for static synchronous networks can be modified to work on dynamic synchronous networks. The algorithm presented here is essentially the same as in [5].

Since edges may fail dynamically, a node u may have no knowledge of the height of a neighboring node v and hence may be unable to decide whether to send a token to v. In our algorithm, which we call **DS**, every node u maintains an estimate $e^u(v)$ of the number of tokens at v for every neighbor v of u. (The value of $e^u(v)$ at the start of the algorithm is arbitrary.) In every step of the algorithm, each node u performs the following operations:

(1) For each live neighbor v of u, if $w(u) - e^u(v) > 12d$, u sends a message consisting of $w(u)$ and a token; otherwise, u sends a message consisting only of $w(u)$. Next, $w(u)$ is decreased by the number of tokens sent.

(2) For each message received from a live neighbor v, $e^u(v)$ is updated according to the message and if the message contains a token, $w(u)$ is increased by one.

Unlike the algorithm for static networks, the above algorithm may (temporarily) worsen the imbalance since a node may have an old estimate of the height of one of its neighbors. Two anomalies may occur while executing **DS**: (i) a token sent by u to v may gain height as it is possible for $w(u) - e^u(v)$ to be greater than $12d$ even if $w(u)$ is at most $w(v)$, and (ii) node u may not send a token to v as it is possible for $w(u) - e^u(v)$ to be at most $12d$ even if $w(u) - w(v)$ is much larger than $12d$. Consequently, the analysis for dynamic networks is more difficult than for static networks. We employ a more complicated amortized analysis to account for the above anomalies.

For every integer i, let S_i denote the set of nodes that have at least $\rho - 12d + 24id$ and at most $\rho + 12d - 1 + 24id$ tokens. Consider T steps of **DS**. We assume without loss of generality that $|S_{>0}| \leq n/2$ at the start of at least $T/2$ steps. As shown in Section 4.2, there exists an index j in $[1, 2\lceil \log_{(1+\alpha/(2d))} n \rceil]$ that is good in at least half of those steps in which $|S_{>0}| \leq n/2$. (Recall that index i is good if $|S_i| \leq \alpha |S_{>i}|/2d$.) If index j is good at the start of step t, we call t a *good* step. For any token p, let $h_t(p)$ denote the height of p after step t, $t > 0$. For convenience, we denote the height of p at the start of **DS** by $h_0(p)$. Similarly, for $t \geq 0$, we define $h_t(u)$ for every node u and $e^u_t(v)$ for every edge (u, v).

With every token at height h, we associate a potential of $\phi(h)$, where $\phi : N \to R$ is defined as follows:

$$\phi(x) = \begin{cases} 0 & \text{if } x \leq 24jd - 11d, \\ (1+\nu)^x & \text{otherwise,} \end{cases}$$

where $\nu = \alpha/(cd^2)$ and $c > 0$ is a constant to be specified later. Let Φ_t denote the total potential of the network after step t. Let Ψ_t denote the potential drop during step t.

We analyze **DS** by means of an amortized analysis over the steps of the algorithm. Let E_t be the set $\{(u, v) : (u, v)$ is live during step t, $u \in S_{>j}$ and $h_{t-1}(u) - h_{t-1}(v) \geq 24d\}$. For every step t, we assign an amortized potential drop of

$$\hat{\Psi}_t = \frac{1}{50} \sum_{\substack{(u,v) \in E_t \\ h_{t-1}(u) > h_{t-1}(v)}} (\phi(h_{t-1}(u) - d) - \phi(h_{t-1}(v) + d)).$$

The definition of $\hat{\Psi}_t$ is analogous to the amount of potential drop that we use in step t in the argument of Section 4.3.2 for the static case. By modifying that argument slightly and choosing appropriate values for the constants c and ε, we show the following lemma.

Lemma 4.6: *If the live edges of G have an edge expansion of α during every step of* **DS**, *then for every good step t we have $\hat{\Psi}_t \geq \varepsilon\nu^2 d^2 \Phi_{t-1}$, where ε is an appropriately chosen constant.*

Proof Sketch: Let M_i denote the set of live edges between nodes in $S_{<i}$ and nodes in $S_{>i}$. Let $m_i = |M_i|$. For any node u, let $g(u)$ represent the group to which u belongs prior to step t. We now place a lower bound on $\hat{\Psi}_t$ which is analogous to that on Ψ in Lemma 4.4 of Section 4.3.2. By the definition of $\hat{\Psi}_t$, we have

$$
\begin{aligned}
\hat{\Psi}_t &= \frac{1}{50} \sum_{\substack{(u,v)\in E_t \\ h_{t-1}(u)>h_{t-1}(v)}} (\phi(h_{t-1}(u)-d) - \phi(h_{t-1}(v)+d)) \\
&\geq \frac{1}{50} \sum_{\substack{(u,v)\in E_t \\ h_{t-1}(u)>h_{t-1}(v)}} \sum_{g(v)<i<g(u)} (\phi(24(i+1)d-13d) - \phi(24(i-1)d+13d)) \\
&= \frac{1}{50} \sum_{i\geq j} \sum_{\substack{(u,v)\in M_i \\ h_{t-1}(u)>h_{t-1}(v)}} (\phi(24(i+1)d-13d) - \phi(24(i-1)d+13d)) \\
&= \frac{1}{50} \sum_{i>j} \sum_{\substack{(u,v)\in M_i \\ h_{t-1}(u)>h_{t-1}(v)}} (\phi(24(i+1)d-13d) - \phi(24(i-1)d+13d)) \\
&\quad + \frac{1}{50} \sum_{\substack{(u,v)\in M_j \\ h_{t-1}(u)>h_{t-1}(v)}} \phi(24(j+1)d-13d) \\
&\geq \frac{22}{50} \sum_{i>j} \sum_{\substack{(u,v)\in M_i \\ h_{t-1}(u)>h_{t-1}(v)}} \nu d(1+\nu)^{24id-11d} + \frac{1}{50} \sum_{\substack{(u,v)\in M_j \\ h_{t-1}(u)>h_{t-1}(v)}} (1+\nu)^{24jd+11d} \\
&\geq \frac{22}{50} \sum_{i>j} m_i \nu d(1+\nu)^{24id-11d} + \frac{1}{50} m_j (1+\nu)^{24jd+11d}.
\end{aligned}
$$

(For the second equation, note that $24id - 13d \leq 24(i-1)d + 13d$. Therefore, $\phi(24id - 13d) \leq \phi(24(i-1)d + 13d)$. The second equation now follows since the sum telescopes. The third equation is obtained by interchanging the sums and noting that $\phi(x)$ is zero for $x \leq 24jd - 11d$. The fourth equation is obtained by partitioning the set M into subsets

$M \setminus M_j$ and M_j. The fifth equation uses the following calculations: (i) $\phi(24id+11d) - \phi(24id-11d) \geq ((1+\nu)^{22d}-1)(1+\nu)^{24id-11d} \geq 22d(1+\nu)^{24id-11d}$, (ii) $\phi(24jd+11d) = (1+\nu)^{24jd+11d}$, and (iii) $\phi(24jd-11d) = 0$. The last equation follows from the definition of m_i.)

We next establish claims similar to Lemma 4.5 and Corollary 4.5.1 of Section 4.3.2 by modifying the constants in the proofs. We thus have $\hat{\Psi}_t \geq \varepsilon\nu^2 d^2 \Phi_{t-1}$ for an appropriately chosen constant ε. □

The following lemma relates the amortized potential drops to the actual potential drops.

Lemma 4.7: *For any initial load distribution and any step $t' > 0$, we have*

$$\sum_{t \leq t'} \Psi_t \geq \left(\sum_{t \leq t'} \hat{\Psi}_t \right) - 2\Phi_0 - n^2\phi(24jd). \tag{4.4}$$

In order to prove Lemma 4.7, we need to address two issues that arise in the dynamic setting: (i) potential gains, i.e., when a token gains height, and (ii) the lack of a otential drop across edges that join nodes differing by at least $24d$ tokens. We show that for any of the above events to occur, "many" tokens should have lost height in previous steps. We use a part of this prior potential drop to account for (i) and (ii). At a high level, our proof follows the lines of Lemma 3 of [5]. Since the potential functions involved are different, however, the two proofs differ considerably in the details. We have included a complete proof of Lemma 4.7 in Appendix D.

The main result follows from Lemmas 4.6 and 4.7. We first show that within $O(1/(\varepsilon\nu^2 d^2))$ steps, there is a step when the actual potential of the network either decreases by a factor of 2 or falls below a threshold value.

Lemma 4.8: *Let t be any integer such that at least $7/(\varepsilon\nu^2 d^2)$ of the first t steps are good. There exists $t' \leq t$ such that $\Phi_{t'} \leq \max\{\Phi_0/2, n^2\phi(24jd)\}$.*

Proof: If $\Phi_0 \leq n^2\phi(24jd)$, then the claim is proved for $t = 0$. For the remainder of the proof, we assume that $\Phi_0 \geq n^2\phi(24jd)$. If $\Phi_{t'} \leq \Phi_0/2$ for any $t' < t$, the claim is proved. Otherwise, for all $t' < t$, we have $\Phi_{t'} > \Phi_0/2$. In this case, we obtain

$$\begin{aligned}
\Phi_t &= \Phi_0 - \sum_{t'<t} \Psi_{t'} \\
&\leq 3\Phi_0 + n^2\phi(24jd) - \sum_{t'<t} \hat{\Psi}_{t'} \\
&\leq 4\Phi_0 - \sum_{\substack{t'<t \\ t'\text{good}}} (\varepsilon\nu^2 d^2)\Phi_{t'} \\
&\leq \Phi_0/2.
\end{aligned}$$

(To obtain the second equation, we invoke Lemma 4.7. For the third equation, we invoke Lemma 4.6 and use the inequalities $\Phi_0 \geq n^2\phi(24jd)$, and $\hat{\Psi}_{t'} \geq 0$ for every t'. The last equation holds since at least $7/(\varepsilon\nu^2 d^2)$ of the t steps are good and $\Phi_{t'} > \Phi_0/2$ for every $t' < t$.) □

Theorem 4.5: *For an arbitrary network G with n nodes, degree d, and initial imbalance Δ, if the live edges at every step t of G have edge expansion α, then the dynamic synchronous multi-port algorithm load balances to within $O(d^2(\log n)/\alpha)$ tokens in $O(\Delta/\alpha)$ steps.*

Proof: We first place an upper bound on the number t of steps such that the height of each node at the end of step t is $O(d^2(\log n)/\alpha^2)$. If Δ is at most $d^2(\log n)/\alpha^2$, then a trivial upper bound is 0.

We now consider the case when Δ is at least $d^2(\log n)/\alpha^2$. By repeatedly invoking Lemma 4.8, we obtain that within $T = \lceil(7/(\varepsilon\nu^2 d^2))\rceil\lceil\log\Phi_0\rceil$ good steps, there exists a step after which the potential of the network is at most $n^2\phi(24jd)$. (Note that the requirement that Lemma 4.7 hold for arbitrary initial values of the estimates, the $e^u(v)$'s, is crucial here.) Since at least $T/4$ of the first T steps are good, there exists $t \leq 4\lceil(7/(\varepsilon\nu^2 d^2))\rceil\lceil\log\Phi_0\rceil$ such that $\Phi_t \leq n^2\phi(24jd)$. Since $\Phi_0 \leq n(1+\nu)^{(\Delta+1)}/\nu$, we have $\log\Phi_0 \leq \log n + (\Delta+1)\log(1+\nu) - \log\nu$. Since $\nu = \alpha/(cd^2)$ and $\log(1+\nu) < \nu$, we have $t = O((\Delta/\alpha) + d^2(\log n)/\alpha^2) = O(\Delta/\alpha)$.

Let h be the maximum height of any node after step t. We thus have

$$\begin{aligned}\phi(h) &\leq \Phi_t \\ &\leq n^2(1+\nu)^{24jd}.\end{aligned}$$

Therefore, if $\phi(h) > 0$, then $h \leq \log_{(1+\nu)}(n^2(1+\nu)^{24jd})$. If $\phi(h) = 0$, then $h \leq 24jd - 11d$. In either case,

$$\begin{aligned}h &\leq 24jd + (2\log n)/\log(1+\nu) \\ &\leq 24jd + (4\log n)/\nu \\ &= O((d^2 \log n)/\alpha).\end{aligned}$$

(The right-hand side of the first equation is an expansion of $\log_{(1+\nu)}(n^2(1+\nu)^{24jd})$. The second equation holds since $\log(1+\nu) < \nu/2$ for c appropriately large. The final equation follows from the relations $\nu = \alpha/(cd^2)$ and $j = O((d \log n)/\alpha)$.)

Thus, at the end of step t, no node has more than $a = \rho + h$ tokens. We now prove by contradiction that for every step after step t, no node has more than $a + d$ tokens. Let t' be the first step after step t such that there exists some node u with more than $a + d$ tokens. Of the $d+1$ highest tokens received by u after step t, at least 2 tokens (say p and q) were last sent by the same neighbor v of u. Without loss of generality, we assume that p arrived at u before q. Let t_1 be the step when p was last sent by v to u. Therefore, we have $e^v_{t_1}(u) \geq h_{t_1}(p) - d \geq a - d$. Hence q can be sent to u only when v has height at least $a + 11d$, which contradicts our choice of t'.

We have shown that after $O(\Delta/\alpha)$ steps, no node ever has more than $\rho + O((d^2 \log n)/\alpha)$ tokens. An easy averaging argument shows that there exists $k = O((d \log n)/\alpha)$ such that after every step $t' \geq t$, $|S_{<-k}| \leq n/2$. By defining an appropriate potential function for tokens with heights below the average and repeating the analysis done for $S_{>j}$, we show that in another $O(\Delta/\alpha)$ steps, all nodes have more than $\rho - O(d^2(\log n)/\alpha)$ tokens. $\square$

As suggested in [5], a simple variant of **DS** can be defined for asynchronous networks.

As shown in [5], the analysis for the dynamic synchronous case can be used for asynchronous networks to yield the same time bounds. Hence, the multi-port local load balancing algorithm balances to within $O(d^2 \log n/\alpha)$ tokens in time $O(\Delta/\alpha)$ on asynchronous networks.

4.5 Tight Bounds on Centralized Load Balancing

In this section, we analyze the load balancing problem in the centralized setting for both single-port and multi-port models. We derive nearly tight bounds on the minimum number of steps required to balance on arbitrary networks in terms of the node and edge expansion of the networks. We assume that the network is synchronous.

We first consider the network $G = (V, E)$ under the single-port model. For any subset X of V, let $\overline{X}$ denote $V \setminus X$, $m(X)$ denote the number of edges in a maximum matching between X and $\overline{X}$, $A(X)$ denote the set $\{v \in \overline{X} : \exists x \in X \text{ such that } (x, v) \in E\}$, and $B(X)$ denote the set $\{x \in X : \exists y \in A(X) \text{ such that } (x, y) \in E\}$. For subsets X and Y of V, let $M(X, Y)$ denote the set of edges with one endpoint in X and the other in Y.

Lemma 4.9: *For any network $G = (V, E)$ with node expansion μ and any subset X of V, we have $m(X) \le \mu \min\{|X|, |\overline{X}|\}/(1 + \mu)$. Moreover, for any subset X of V, $m(X \cup A(X)) \le |A(X)|$.*

Proof: Without loss of generality, assume that $|X| \le |\overline{X}|$. Consider the bipartite graph $H = (B(X), A(X), M(X, \overline{X}))$. A maximum matching in H is equal to a maximum flow in the graph $I = (B(X) \cup A(X) \cup \{s, t\}, M(X, \overline{X}) \cup \{(s, x) : x \in B(X)\} \cup \{(x, t) : x \in A(X)\})$ from source s to sink t, where all of the edges of I have unit capacity. We will show that every cut C of I separating s and t is of cardinality at least $\mu|X|/(1 + \mu)$. Consider any cut $C = (S, T)$ with $s \in S$ and $t \in T$. The set of edges in C is $M(S, T)$. Let $Y = T \cap B(X)$ and $Z = T \cap A(X)$. The capacity of C, given by $|M(S, T)|$, can be lower bounded as follows.

$$|M(S,T)| \quad = \quad |Y| + |M(Y, A(X) \setminus Z)| + |M(B(X) \setminus Y, Z)| + |A(X) \setminus Z|$$

$$
\begin{aligned}
&\geq |Y| + |M(B(X) \setminus Y, Z)| + |A(X) \setminus Z| \\
&\geq |A(X \setminus Y)| \\
&\geq \mu|X \setminus Y| \\
&= \mu(|X| - |Y|) \\
&\geq \mu|X|/(1+\mu).
\end{aligned}
$$

(For the third equation, see Figure 4.4. Three subsets of nodes contribute to the set $A(X \setminus Y)$: (i) the set of nodes in Y that have an edge to a node in $X \setminus Y$, (ii) the set of nodes in Z that have an edge to a node in $X \setminus Y$, and (iii) the set of nodes in $A(X) \setminus Z$ that have an edge to a node in $X \setminus Y$. The size of the three sets is bounded by $|Y|$, $|M(B(X) \setminus Y, Z)|$, and $|A(X) \setminus Z|$, respectively. The fourth equation follows from the definition of $A(X \setminus Y)$. The fifth equation holds since Y is a subset of X. The last equation holds since $|Y| \leq |M(S,T)|$.) For the second part of the lemma, we note that

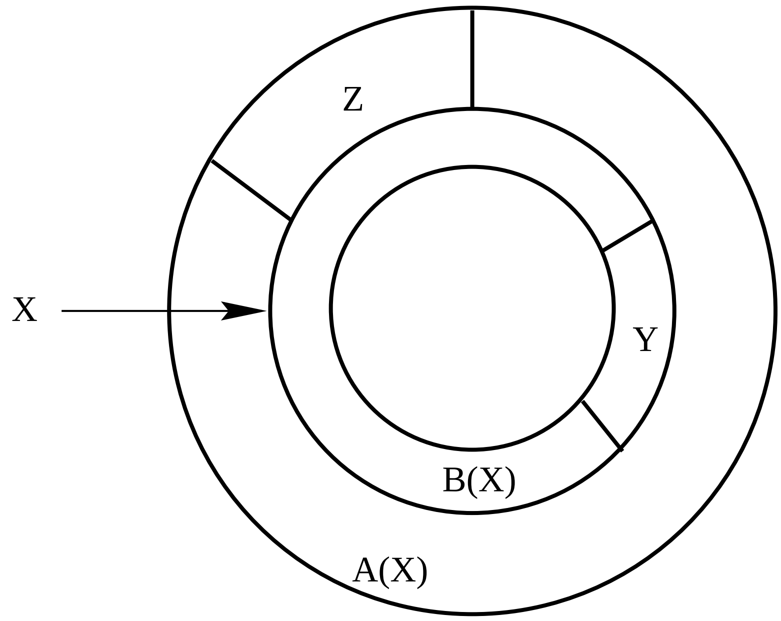


Figure 4.4: The sets X, Y, Z, $A(X)$, and $B(X)$ in the proof of Lemma 4.9.

since all of the neighbors of X are in $A(X)$, any node in $X \cup A(X)$ that connects to some node outside of $X \cup A(X)$ is in $A(X)$. Therefore, $m(X \cup A(X)) \leq |A(X)|$. $\square$

Theorem 1 of [93] obtains tight bounds on the centralized complexity of load balancing in terms of the function m. We restate the theorem using our notation and terminology. Before stating the theorem, we need one additional notation. For any subset X of nodes of any network, let $I(X)$ denote the number of tokens held by nodes in X in the initial distribution.

Theorem 4.6 ([93]): *Consider a network $G = (V, E)$ in the single-port model. The network G can be balanced in at most $\max_{\emptyset \not\subseteq X \not\subseteq V} \lceil (I(X) - \rho|X|)/m(X) \rceil$ steps so that every node has at most $\lceil \rho \rceil + 1$ tokens. Moreover, any algorithm takes at least $\max_{\emptyset \not\subseteq X \not\subseteq V} \lceil (I(X) - \rho|X|)/m(X) \rceil$ steps to balance the network so that every node has at most $\lceil \rho \rceil$ tokens.* □

Theorem 4.6 and Lemma 4.9 imply the following result.

Lemma 4.10: *Assume the single-port model. Any network G with node expansion μ and initial imbalance Δ can be balanced in at most $\lceil \Delta(1+\mu)/\mu \rceil$ steps so that every node has at most $\lceil \rho \rceil + 1$ tokens. Moreover, there exists a network G and an initial load distribution with imbalance Δ such that any algorithm takes at least $\lceil \Delta(1+\mu)/\mu \rceil$ steps to balance G such that every node has at most $\lceil \rho \rceil$ tokens.*

Proof: If $I(X)$ is the the total number of tokens belonging to nodes in X in the initial distribution distribution, then we have: $-\Delta|X| \leq I(X) - \rho|X| \leq \Delta|X|$ for all X. Moreover, $|I(X) - \rho|X|| = |I(\overline{X}) - \rho|\overline{X}||$. Therefore, for all X, $|I(X) - \rho|X|| = \Delta \min\{|X|, |\overline{X}|\}$. By Lemma 4.9, $m(X)$ is at least $\mu \min\{|X|, |\overline{X}|\}/(1+\mu)$ for all X. Thus, the first claim of Theorem 4.6 establishes the first claim of the desired lemma.

For the second claim of the lemma, given any μ, we construct the following network $G = (V, E)$ with node expansion μ. The node set V is partitioned into 3 sets X, Y, and Z such that: (i) $|Y| = \mu|X|$, and (ii) $|Z| = |X|(1+\mu)^2/(1-\mu)$. Let n and x denote $|V|$ and $|X|$, respectively. Thus, n equals $x(1+\mu+(1+\mu)^2/(1-\mu)) = 2x(1+\mu)/(1-\mu)$. The edge set E is the union of the sets $X \times X$, $X \times Y$, $Y \times Y$, $Y \times Z$, $Z \times Z$.

We now show that the node expansion of G is μ. Consider any nonempty subset U of V of size at most $n/2$ and let X', Y', and Z' denote $U \cap X$, $U \cap Y$, and $U \cap Z$, respectively. Let $N(U)$ denote the number of neighbors of U that lie outside of U. We need to show that $N(U)$ is at least $\mu|U|$.

We consider two cases: (i) Y' and Z' are both empty, and (ii) Y' is nonempty or Z' is nonempty. In the first case, $U = X'$. Therefore, $N(U) \geq |Y| = \mu x \geq \mu|U|$. In the second case, we have:

$$\begin{aligned}
N(U) &\geq |Z| - |Z'| \\
&\geq |Z| - |U| \\
&\geq x((1+\mu)^2/(1-\mu) - (1+\mu)/(1-\mu)) \\
&= x\mu(1+\mu)/(1-\mu) \\
&\geq \mu|U|.
\end{aligned}$$

(The second equation holds since Z' is a subset of U. For the third equation, note that $|U| \leq n/2 = x(1+\mu)/(1-\mu)$. The last equation follows from the upper bound of $x(1+\mu)/(1-\mu)$ on $|U|$.)

We now apply the second claim of Lemma 4.9 to the subset X. Since $A(X) = Y$, $m(X \cup Y) = \mu x = \mu|X \cup Y|/(1+\mu)$. Given any Δ, consider the initial token distribution in which each node in $X \cup Y$ has $\rho + \Delta$ tokens, and each node in Z has $\rho - \Delta(1-\mu)/(1+\mu)$ tokens, where ρ is any integer that is at least $\Delta(1-\mu)/(1+\mu)$. (Note that the average number of tokens is ρ.) By applying the second claim of Theorem 4.6, we obtain that the number of steps to balance G so that each node has at most ρ tokens is at least $(I(X \cup Y) - \rho|X \cup Y|)/m(X \cup Y) \geq \Delta|X \cup Y|/m(X \cup Y) \geq \Delta(1+\mu)/\mu$. Since the number of steps is an integer, the desired claim follows. $\square$

By using the techniques of [93], we can modify the proof of Lemma 4.10 to show that any network G with node expansion μ and initial imbalance Δ can be globally balanced to within 3 tokens in at most $2\lceil \Delta(1+\mu)/\mu \rceil$ steps. The extra factor of 2 is required because even after balancing the network so that each node has at most $\lceil \rho \rceil + 1$ tokens, there may exist a node with considerably fewer than ρ tokens. It takes an additional $\lceil \Delta(1+\mu)/\mu \rceil$ steps to bring the network to a state in which the global imbalance is at most 3.

Lemma 4.10 implies that the time bound achieved by the single-port algorithm (see Theorems 4.1 and 4.3) is not optimal for all networks. An example of a network for which the single-port algorithm is not optimal is the hypercube whose maximum degree is $\log n$, edge expansion is 1, and node expansion is $\Theta(1/\sqrt{\log n})$. The local algorithm balances in $\Omega(\Delta \log n)$ time, while there exists an $O(\Delta\sqrt{\log n} + \log^2 n)$ time load balancing algorithm for the hypercube [101] which is optimal for Δ sufficiently large. For the class of constant-degree networks, however, the time taken by the single-port algorithm to reduce the global imbalance to $O(\log n/\mu)$ (see Theorem 4.3) is within a constant factor of the time taken by any algorithm to completely balance the network (see Lemma 4.10).

The proofs of Theorem 1 of [93] and Lemma 4.10 can be modified to establish the following result for the multi-port model.

Lemma 4.11: *Assume the multi-port model. Any network G with edge expansion α and initial imbalance Δ can be balanced in at most $\lceil \Delta/\alpha \rceil$ steps so that every node has at most $\lceil \rho \rceil + d$ tokens. Moreover, for every network G, there exists an initial load distribution with imbalance Δ such that any algorithm takes at least $\lceil \Delta/\alpha \rceil$ steps to balance G so that every node has at most $\lceil \rho \rceil$ tokens.*

Proof Sketch: We prove that there exists a centralized algorithm that balances to within d tokens in at most $T = \max_{\emptyset \subset X \subset V} \left\lceil \frac{|I(X)-\rho|X||}{|M(X,\overline{X})|} \right\rceil$ steps. For all $X \subseteq V$, we have (i) $|I(X) - \rho|X|| \leq \Delta \min\{|X|, |\overline{X}|\}$ (see proof of Lemma 4.10), and (ii) $|M(X, \overline{X})| \geq \alpha \min\{|X|, |\overline{X}|\}$. It follows from (i) and (ii) that $T \leq \lceil \Delta/\alpha \rceil$.

We modify the proofs of Theorem 1 and Lemma 4 of [93] (where the single-port model was assumed) to establish the desired claims for the multi-port model. We transform the load balancing problem on G to a network flow problem on a directed graph $H = (V', E')$ which is constructed as follows. Let V_i be $\{\langle v, i\rangle : v \in V\}$, $0 \leq i \leq T$. Let E_i be $\{(\langle u, i\rangle, \langle v, i+1\rangle) : (u, v) \in E \text{ or } u = v\}$, $0 \leq i < T$. We set V' to $\{s\} \cup \bigcup_{0 \leq i \leq T} V_i \cup \{t\}$, and E' to $\{(s, \langle v, 0\rangle) : v \in V\} \cup \bigcup_{0 \leq i < T} E_i \cup \{(\langle v, T\rangle, t) : v \in V\}$.

For any v in V, the capacity of the edge $(s, \langle v, 0\rangle)$ is $w(v)$. For any (u, v) in E, the capacity of any edge $(\langle u, i\rangle, \langle v, i+1\rangle)$, $0 \le i < T$, is 1. For any v in V, the capacity of any edge $(\langle v, i\rangle, \langle v, i+1\rangle)$, $0 \le i < T$, is ∞. For any v in V, the capacity of the edge $(\langle v, T\rangle, t)$ is $\lceil \rho \rceil + d$.

We show that the value of the maximum integral flow in H is equal to the total number of tokens N in V, from which it follows that there exists a centralized algorithm that balances to within d tokens in T steps. Consider any cut $C = (S, T)$ of H separating $s \in S$ and $t \in T$. Let $S_i = S \cap V_i$ and $D(S_i) = \{v \in V : \langle v, i\rangle \in S_i\}$. If $S_0 = \emptyset$, or $S_T = V_T$, or there is an edge of infinite capacity, then the capacity of C is at least N. Otherwise, the number of edges from V_i to V_{i+1} that belong to the cut is at least $|M(D(S_i), \overline{D(S_i)})| - d(|S_{i+1}| - |S_i|)$. Moreover, since there is no edge with infinite capacity in C, $D(S_i)$ is a subset of $D(S_{i+1})$. Thus the capacity of C is at least

$$\begin{aligned}
& I(D(V_0) \setminus D(S_0)) + (\sum_{i=0}^{T-1} \Big(|M(D(S_i), \overline{D(S_i)})| - d(|S_{i+1}| - |S_i|)\Big) + (\lceil \rho \rceil + d)|S_T| \\
\ge\ & I(D(V_0) \setminus D(S_0)) + (\sum_{i=0}^{T-1} ((I(D(S_i)) - \rho|S_i|)/T - d(|S_{i+1}| - |S_i|)) + (\lceil \rho \rceil + d)|S_T| \\
\ge\ & I(D(V_0) \setminus D(S_0)) + (\sum_{i=0}^{T-1} ((I(D(S_0)) - \rho|S_T|)/T - d(|S_T| - |S_0|)) + (\lceil \rho \rceil + d)|S_T| \\
\ge\ & I(D(V_0) \setminus D(S_0)) + I(D(S_0)) - \rho|S_T| + d|S_0| + \lceil \rho \rceil |S_T| \\
\ge\ & N.
\end{aligned}$$

(In the first equation: (i) $I(D(V_0) \setminus D(S_0))$ is the capacity of the edges from s to V_0 that belong to the cut, (ii) $|M(D(S_i), \overline{D(S_i)})| - d(|S_{i+1}| - |S_i|)$ is the capacity of the edges from V_i to V_{i+1} that belong to the cut, and (iii) $(\lceil \rho \rceil + d)|S_T|$ is the capacity of the edges from S_T to t that belong to the cut. The second equation follows from the definition of T and the equality $|D(S_i)| = |S_i|$. For the third equation, note that $D(S_0)$ is a subset of $D(S_i)$ for all i and $|S_T| \ge |S_i|$ for all i. The fourth equation is obtained since the sum of $|S_{i+1}| - |S_i|$ telescopes. The final equation is obtained since $I(D(V_0)) = N$.) Since the capacity of the cut $(\{s\}, V' \setminus \{s\})$ equals N, the maximum flow in H is N.

To prove the second part of the lemma, given any network G with a partition

(V_1, V_2) of its nodes such that $|V_1| \le n/2$ and $|M(V_1, V_2)| = \alpha|V_1|$, we define an initial load distribution with average ρ in which each node in V_1 has $\rho + \Delta$ tokens and each node in V_2 has $\rho - \Delta|V_1|/|V_2|$ tokens. The desired claim holds since at least $\Delta|V_1|$ tokens need to leave the set V_1. □

Lemma 4.11 implies that the local multi-port algorithm is asymptotically optimal for *all* networks. As in the single-port case, we can modify the above proof to obtain upper bounds on the centralized complexity of globally balancing a network. We can show that any network G with edge expansion α and initial imbalance Δ can be globally balanced to within $d + 1$ tokens in at most $2\lceil \Delta/\alpha \rceil$ steps.

4.6 Concluding Remarks

In this chapter, we have shown that the local balancing approach brings any network to a state of small global imbalance in time that is asymptotically optimal in the worst-case. Two natural questions come to mind. Can we improve the guarantee on the quality of balance achieved by the algorithms? Is the local balancing approach optimal, not just in the worst-case, but for all distributions?

Quality of balance. It is easy to see that there exist distributions for which the global imbalance guaranteed by the multi-port and single-port balancing algorithms cannot be better than the network diameter. However, even if a global imbalance that is within a constant factor of the diameter is reached, the network may not be *locally balanced* to within a small number (say $O(1)$) of tokens. We say that a network is locally balanced to within t tokens if the maximum difference between the number of tokens at any two neighboring nodes is at most t. Both the single-port and multi-port algorithms will eventually locally balance the network, the single-port algorithm to within one token, and the multi-port algorithm to within $2d$ tokens. However, even after reducing the global imbalance to a small value, the time for either of these algorithms to reach a locally balanced state can be quite large. For example, it is shown in [53] that after reaching a state that is globally balanced to within $O((d \log n)/\mu)$ tokens, the multi-port algorithm may take another $\Omega(n^{1/2})$ steps to reach a state that is locally balanced to within $2d$ tokens. (A similar result for the single-port algorithm is also contained in [53].)

Performance ratio for all distributions. It is open whether the time taken by the local balancing approach is asymptotically optimal for all distributions on all networks. We believe that an improved analysis will require substantially new techniques that consider the particular topology of the given network in greater detail.

In the following chapter, we settle both of the questions raised above in the affirmative for the special case of ring networks.

Chapter 5

Static Load Balancing on Rings

5.1 Introduction

While the results of Chapter 4 establish that the local balancing approach is optimal in the worst-case, it is not the case that the upper bounds shown are optimal for all initial distributions. In fact, for the class of ring networks, the upper bound may be suboptimal by an $\Omega(n)$ factor. To see this, note that the edge expansion of an n-node ring is $\Theta(n)$. Therefore, an application of Theorem 4.2 to the special case of a ring network yields that if the initial imbalance is Δ, then the multi-port algorithm defined in Section 4.1.1 balances in $O(n\Delta)$ steps. While there exists a distribution with imbalance Δ for which any algorithm takes $\Omega(n\Delta)$ steps to balance, it is easy to construct distributions with imbalance Δ that can be balanced in $O(\Delta)$ steps.

In this chapter, we show that a simple variant of the multi-port algorithm of Section 4.1.1 converges to a *completely* balanced distribution in near-optimal time for *every* initial distribution on both *synchronous* and *asynchronous* rings. We are not aware of any other load balancing algorithm that has been shown to achieve such universal near-optimality with respect to a non-trivial family of networks (e.g., rings). All previous optimality results known for load balancing are worst-case results.

5.1.1 Overview of the Results

Let R be a ring network with the set $[n] = \{0, 1, \ldots, n-1\}$ of nodes and the set $\{(i, (i+1) \bmod n\}$ of edges. The local balancing algorithm, which we denote by $\mathcal{A}$, is defined as follows. In each step, for all i in $[n]$, node i sends a token to node $(i+1) \bmod n$ if and only if i has more tokens than $(i+1) \bmod n$. (See Section 5.2 for a message-passing implementation of $\mathcal{A}$.) We note that there is a single direction, say clockwise, in which all the token movements in $\mathcal{A}$ take place. We refer to algorithms that move tokens in the clockwise direction as *unidirectional* algorithms.

We first consider a synchronous model of computation in which: (i) in each step of the network, all of the nodes simultaneously perform one step of their computations, and (ii) each message sent during a step is delivered prior to the start of the subsequent step. We show that:

- The number of steps taken by $\mathcal{A}$ to balance any distribution b on a synchronous ring is at most $4\mathrm{OPT}(b) + n$, where $\mathrm{OPT}(b)$ is the time taken by an optimal centralized algorithm to balance b. The proof is given in Section 5.4.

We note that the optimal centralized algorithm need not be a unidirectional algorithm; that is, $\mathrm{OPT}(b)$ is the time taken to balance b by the best algorithm among *all* algorithms that send and/or receive at most one token along each of its incident edges in each step. In fact, if $\mathrm{OPT}(b)$ was instead defined as the time taken by an optimal centralized *unidirectional* algorithm to balance b, then the factor of 4 in the stated bound could be replaced by 2.

Our next result concerns an asynchronous model of computation, in which local computations may be performed at arbitrary speeds and messages may be delayed arbitrarily, subject to the constraint that each message is eventually delivered and each computation is eventually performed [79]. In order to measure the time complexity in the asynchronous model, we define a *round* to be a minimal sequence of steps in which each component of the ring (i.e., each node or edge) is scheduled at least once. The time complexity of an algorithm is then defined as the maximum number of rounds taken among all possible schedulings of the components. (See Section 5.5 for a formal description of the asynchronous model.)

The above notion of time is based on the model proposed in [13] for shared memory systems. An analogous model for message-passing systems was studied in [16]. Moreover, our model is equivalent to that proposed in [87], where the time complexity of an algorithm is defined to be the longest amount of elapsed real time from the start to the completion of the algorithm, assuming that the time delay between two steps of the same network component is at most unity [13]. (The model proposed in [87] has been subsequently used in the study of several distributed computing problems [18, 20].)

We generalize our result for the synchronous model to the asynchronous model at the expense of a factor of 2 in the time complexity. In particular, we show that:

- The number of rounds taken by $\mathcal{A}$ to balance any distribution b on an asynchronous ring is at most $8\mathrm{OPT}(b) + 2n$. The proof is given in Section 5.5.

We remark that if $\mathrm{OPT}(b)$ were instead defined as the time taken by an optimal centralized *unidirectional* algorithm for b, then the factor of 8 in the stated bound could be replaced by 4. We also show that in both the synchronous and asynchronous models, for every initial token distribution, the message complexity of $\mathcal{A}$ is asymptotically optimal among all unidirectional algorithms.

5.1.2 Related Work

In recent work [14], asynchronous balancing algorithms on several networks including the ring have been studied. However, the results of [14] are geared towards establishing eventual convergence in the presence of dynamic network changes, while we are interested in determining the time to convergence for static load balancing. Also related is the result of [39], where a worst-case bound on the number of token migrations is given for a model in which tokens can be transferred between any two nodes.

Our result for the asynchronous model is similar in spirit to that of [20], in that our asynchronous algorithm is not obtained by using a general synchronizer [18] in conjunction with an algorithm optimized for a synchronous model. Instead, we show that $\mathcal{A}$ is directly implementable on asynchronous rings and hence avoids the overhead and complexity of a synchronizer while achieving near-optimal bounds.

5.2 The Unidirectional Algorithm $\mathcal{A}$

In this section, we give a message-passing implementation of the unidirectional algorithm $\mathcal{A}$ introduced in Section 5.1. Recall that the nodes of the ring are assigned unique labels from the set $[n]$. For convenience, we adopt the following notational convention: any arithmetic expression referring to a node is interpreted modulo n. For example, we will often refer to the neighbors of an arbitrary node i as node $i-1$ and node $i+1$, rather than node $(i-1) \bmod n$ and node $(i+1) \bmod n$.

In $\mathcal{A}$, each node i repeatedly communicates with node $i+1$ and sends a token to $i+1$ whenever the number of tokens at i exceeds that at $i+1$. Figure 5.1(b) illustrates one step of the algorithm for the example given in Figure 5.1(a). In order to implement this balancing scheme efficiently, node i maintains three variables related to the number of tokens at $i+1$: (i) a count $x(i)$ of the number of tokens that i has sent to $i+1$ since the start of the algorithm, (ii) an estimate $y(i)$ of the number of tokens that $i+1$ has sent to $i+2$ since the start of the algorithm, and (iii) the number $z(i)$ of tokens at $i+1$ initially. At a given point in the execution of the algorithm, let $w(i)$ denote the number of tokens at i. Thus, $w(i)$ equals $w_0(i)$ initially.

In $\mathcal{A}$, the nodes communicate with their neighbors using three types of messages: (i) *height*, a message that i sends to $i-1$ indicating the number of tokens at i, (ii) *update*, a message that i sends to $i-1$ indicating that i has sent a new token to $i+1$, and (iii) *token*, a message consisting of a token sent by i to $i+1$. In terms of these messages, the algorithm can be described as follows.

- In the initial step, node i performs the following operations: (i) set $x(i)$ and $y(i)$ to zero and set $z(i)$ to ∞, and (ii) send a *height* message with value $w(i)$ to $i-1$.

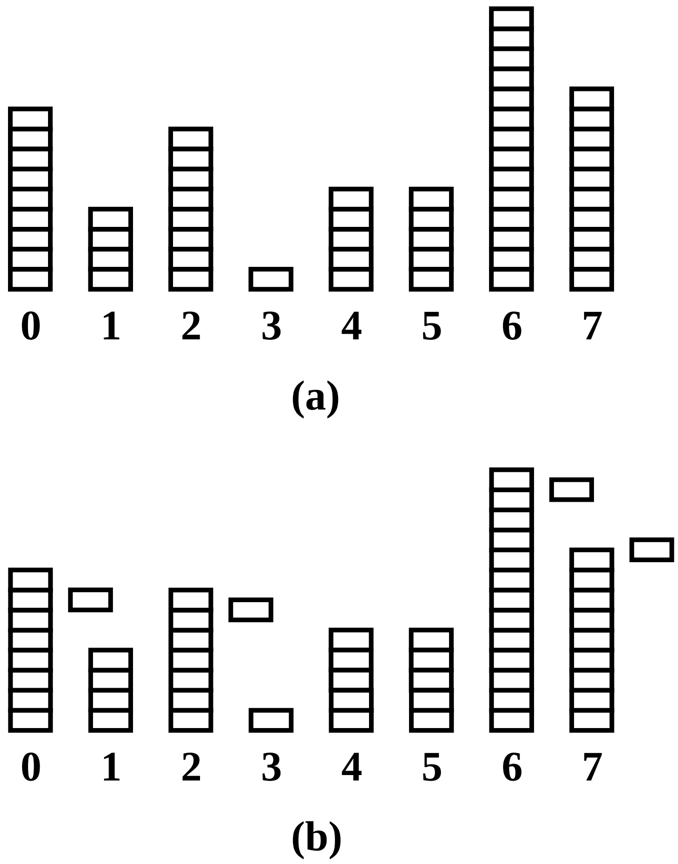


Figure 5.1: (a) An 8-node ring with an initial distribution of tokens. The value of ρ for this distribution is 7. (b) One step of $\mathcal{A}$ on the example given in part (a). Nodes 0, 2, 6, and 7 send one token each to nodes 1, 3, 7, and 0, respectively.

- In each subsequent step, i performs the following operation. If $w(i) > z(i) + x(i) - y(i)$, then: (i) decrement $w(i)$ by 1, (ii) increment $x(i)$ by 1, (iii) send a *token* message to $i+1$, and (iv) send an *update* message to $i-1$.

- On receipt of a *height* message, i sets $z(i)$ to the value of the message. On receipt of an *update* message, i increments $y(i)$. On receipt of a token, i increments $w(i)$.

5.3 Preliminaries

Let $\mathbf{Z}$ and $\mathbf{N}$ denote the integers and nonnegative integers, respectively. Let $V = [n] \mapsto \mathbf{Z}$ denote the set of n-tuples of integers. For any t in $\mathbf{N}$ and i in $[n]$, let w_t be defined as follows: $w_t(i)$ is the number of tokens at node i at the start of step t. (We number the steps from 0.) For any b in V, let $\rho(b) = \frac{1}{n}\sum_{i\in[n]} b(i)$ denote the average number of tokens in b. We say that the ring is *balanced* in step t if $w_t(i)$ is $\lfloor \rho(b) \rfloor$ or $\lceil \rho(b) \rceil$ for all i in $[n]$, where b is the initial distribution. For any subset S of $[n]$, let $w_t(S)$ denote the total number of tokens in S at the start of step t.

For any i and j in $[n]$, let $d(b, i, j)$ denote the total "imbalance" associated with the set of contiguous nodes obtained when going from node i to node j in the clockwise direction (i and j included). Formally, we have:

$$d(b, i, j) = \sum_{0 \le k \le (j-i) \bmod n} (b(i + k) - \rho(b)).$$

(In other words, $d(b, i, j)$ equals $\sum_{i\le k\le j}(b(k)-\rho(b))$ if $i \le j$, and $[\sum_{j\le k<n}(b(k)-\rho(b)) + \sum_{0\le k\le i}(b(k) - \rho(b))]$ otherwise.) For example, if b is the distribution given in Figure 5.1(a), then $d(b, 2, 4)$ is -7, and $d(b, 7, 2)$ is 2.

Let $\ell(b)$ and $m(b)$ be two integers such that $d(b, \ell(b), m(b))$ is $\max_{i,j} d(b, i, j)$. We define the *discrepancy* of a distribution to be the maximum imbalance among all sets of contiguous nodes of the ring. Thus, the discrepancy $D(b)$ of a distribution b is given by $d(b, \ell(b), m(b))$. For the example in Figure 5.1(a), the values of ℓ and m are 6 and 1, respectively, and the discrepancy is 12.

Without loss of generality, we assume for the remainder of this chapter that $\ell(b)$ is zero as we can relabel the nodes appropriately otherwise. Moreover, we will be concerned with applying the functions ρ, d, and m with respect to the initial token distribution. Therefore, as a shorthand, we let ρ, $d(i, j)$, and m, denote $\rho(w_0)$, $d(w_0, i, j)$, and $m(w_0)$, respectively.

5.4 Analysis for Synchronous Rings

In this section, we analyze $\mathcal{A}$ under the synchronous model of computation. For simplicity, we assume, in both this section as well as Section 5.5, that ρ, the average number of tokens in the initial distribution, is an integer.

In the synchronous model, each node executes in a lock-step manner and each message is transmitted in a single step. By the definitions of $x(i)$, $y(i)$, and $z(i)$, we obtain that the value of $z(i) + x(i) - y(i)$ at the start of step t equals $w_t(i+1)$ for any $t > 0$. Therefore, each step of node i can be expressed as follows: if $w_t(i) > w_t(i+1)$, then send a token to $i+1$. For our analysis, it is helpful to consider a generalization of $\mathcal{A}$ given by Definition 5.2 below.

Definition 5.1: *We say that a step t of an algorithm is an S-***step***, where S is a subset of $[n]$, if each node not in S is idle in step t and each node i in S performs the following operation: if $w_t(i) > w_t(i+1)$, then i sends a token to $i+1$.* □

Definition 5.2: *A* **partial algorithm** *$\mathcal{B}$ is one in which each step is an S-step for some subset S of $[n]$. For any t in $\mathbf{N}$, we let $\mathcal{B}(t)$ denote the set S such that step t of $\mathcal{B}$ is an S-step.* □

It follows from Definitions 5.1 and 5.2 that $\mathcal{A}$ is a partial algorithm in which each step after step 0 is an $[n]$-step. We obtain a bound on the running time of $\mathcal{A}$ by providing a general analysis that applies to all partial algorithms. Before proceeding to this analysis, which is given in Section 5.4.1, we present some additional definitions.

Given two partial algorithms $\mathcal{B}$ and $\mathcal{C}$, we say that $\mathcal{B}$ *covers* (resp., *is covered by*) $\mathcal{C}$ if for all t, $\mathcal{B}(t)$ is a superset of (resp., subset of) $\mathcal{C}(t)$. Consider a partial algorithm $\mathcal{B}$. For i in $\mathbf{N}$, let r_i be defined as follows: r_0 is -1 and for all $i > 0$, r_i is the smallest integer greater than r_{i-1} such that $\cup_{r_{i-1} < j \leq r_i} \mathcal{B}(j) = [n]$. We define the *ith round* of $\mathcal{B}$ to be the sequence of steps in the interval $[r_i + 1, r_{i+1}]$.

5.4.1 Analysis of Partial Algorithms

While the number of tokens present at each node of the ring after t steps of $\mathcal{A}$ or any other partial algorithm is easy to calculate, the particular token distribution obtained does not directly provide a good barometer for the progress of the algorithm. In Chapter 4, we analyze load balancing algorithms by first assigning to each node a potential that grows exponentially with the imbalance at the node and then showing that the sum of the potentials of the nodes decreases rapidly with time. While the preceding measure proves to be useful in reducing the complexity of the worst-case analysis for general networks, it appears to be overly simplistic for our purposes since information about the manner in which the imbalance is distributed is lost.

By exploiting the simple structure of ring networks, we are able to capture the precise distribution of the imbalance of the network in a measure, referred to as the *prefix sum vector*. For each t in $\mathbf{N}$, let p_t be defined as follows:

$$p_t(i) = \sum_{0 \le j \le i} (w_t(j) - \rho) \tag{5.1}$$

for all i in $[n]$. (In other words, p_t is the n-tuple of the prefix sums of the difference between the number of tokens at each node at the start of step t and the average.) Given an initial token distribution $w_0 = b$, let $T(b)$ denote $\sum_{i \in [n]} p_0(i)$.

Figure 5.2 gives the value of $w_0(j) - \rho$ for each node j in the instance given in Figure 5.1(a). (Note that the nodes have been relabeled so that ℓ is 0, i.e., the interval of nodes with the largest total imbalance begins with 0.) Applying Equation 5.1, we find that the prefix sum vector for the example is $(7, 10, 12, 9, 10, 4, 2, 0)$.

The following lemma gives a lower bound on the time complexity of any balancing algorithm and the number of token transmissions of any unidirectional balancing algorithm in terms of $D(b)$ and $T(b)$, respectively, where b is the initial token distribution. Recall that $D(b)$, which is formally defined in Section 5.3, is the discrepancy of b.

Lemma 5.1: *Any algorithm takes at least $D(b)/2$ steps to balance b. Any unidirectional algorithm incurs at least $T(b)$ token transmissions to balance b.*

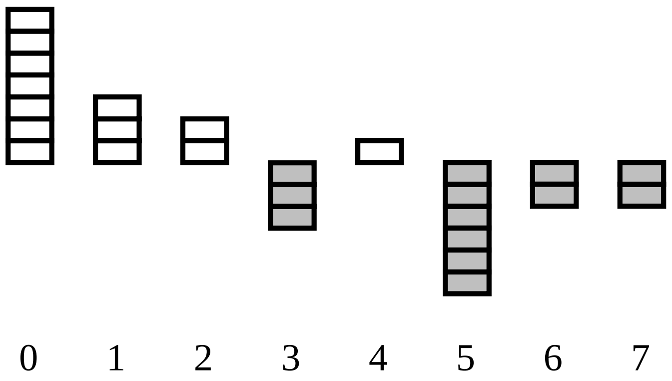


Figure 5.2: The imbalance at each node of the ring given the distribution in Figure 5.1(a). The unshaded tokens represent the number of tokens more than the average while the shaded tokens represent the number of tokens less than the average. Note that the nodes have been relabeled so that the value of ℓ is 0. Hence, the set $[0, 2]$ has the maximum total imbalance, 12, which equals the discrepancy of the distribution. By Equation 5.1, the prefix sum vector is $(7, 10, 12, 9, 10, 4, 2, 0)$.

Proof: Consider the set $S = \{i : 0 \leq i \leq m(b)\}$ of nodes. By definition, $w_0(S)$ is $D(b) + \rho|S|$. (Recall that $\ell(b)$ is 0.) If the ring is balanced in t steps, then for each node i in S, $w_t(i)$ is ρ. Therefore, $w_t(S)$ is at most $\rho|S|$, and hence, at least $D(b)$ tokens are sent out of S in t steps. Since at most two tokens can be sent out of S per step, t is at least $D(b)/2$.

For each i in $[n]$, the number of token transmissions across edge $(i, i+1)$ required by any unidirectional algorithm is at least $p_0(i)$ since $p_0(i)$ is the excess number of tokens in the interval $[0, i]$. Therefore, the total number of token transmissions needed by any unidirectional algorithm to balance b is at least $T(b)$. $\square$

The remainder of this section is devoted to proving that the number of rounds taken by any partial algorithm to balance a distribution b is at most $2D(b) + n - 1$. We begin by determining the effect of a step of a partial algorithm on the prefix sum vector. For this purpose, it is useful to define a partial order $\preceq$ on V as follows: $b \preceq c$ if and only if $b(i) \leq c(i)$ for all i in $[n]$. For convenience, we use 0 to denote the n-tuple each of whose components is 0. Lemma 5.2 expresses a partial algorithm as a recurrence relation among the prefix sum vectors.

Lemma 5.2: *For any partial algorithm $\mathcal{B}$, we have: if i is in $\mathcal{B}(t)$ and $2p_t(i) > p_t(i-1) + p_t(i+1)$, then $p_{t+1}(i)$ is $p_t(i) - 1$; otherwise, $p_{t+1}(i)$ is $p_t(i)$.*

The proof of Lemma 5.2 follows from Lemmas 5.3 and 5.5 below.

Lemma 5.3: *For any partial algorithm $\mathcal{B}$, if $0 \preceq p_t$, then: if i is in $\mathcal{B}(t)$ and $2p_t(i) > p_t(i-1) + p_t(i+1)$, then $p_{t+1}(i)$ is $p_t(i) - 1$; otherwise, $p_{t+1}(i)$ is $p_t(i)$.*

Proof: Since $0 \preceq p_t$, $p_t(0)$ is nonnegative, and hence $w_t(0)$ is at least ρ. Moreover, by definition, $p_t(n-1)$ is 0. Since $p_t(n-2)$ is nonnegative, $w_t(n-1)$ is at most ρ. Therefore, no token is sent from node $n-1$ to node 0. It follows that for each i in $[n]$, if node i sends a token to node $i+1$, then $p_{t+1}(i)$ is $p_t(i) - 1$; otherwise, $p_{t+1}(i)$ is $p_t(i)$. Node i sends a token to $i+1$ if and only if i is in $\mathcal{B}(t)$ and $p_t(i) - p_t(i-1)$ is greater than $p_t(i+1) - p_t(i)$. The desired claim follows. ∎

Lemma 5.4: *For any token distribution, we have $0 \preceq p_0$.*

Proof: The proof is by contradiction. Let i be the smallest nonnegative integer such that $p_0(i)$ is negative. From the definition of i, it follows that $d(i+1, m)$ equals $d(0, m) - d(0, i)$. Since $d(0, i) = p_0(i) < 0$, we obtain that $d(i+1, m)$ is greater than $d(0, m)$ contradicting the definition of m. ∎

Lemma 5.5: *For any partial algorithm $\mathcal{B}$ and all t in $\mathbf{N}$, we have $0 \preceq p_t$.*

Proof: The proof is by induction on t. The induction basis follows from Lemma 5.4.

For the induction hypothesis, we assume that $0 \preceq p_t$. For the induction step, we consider step t. We need to show that $0 \preceq p_{t+1}$. By Lemma 5.3, we have: if $2p_t(i) > p_t(i-1) + p_t(i+1)$, then $p_{t+1}(i)$ is $p_t(i) - 1$; otherwise, $p_{t+1}(i)$ is $p_t(i)$. In either case, since $p_t(i)$ is nonnegative (by the induction hypothesis) and is an integer for all i, we obtain that $p_{t+1}(i)$ is nonnegative for all i, thus completing the induction step.
∎

Lemma 5.6 shows that the each step of a partial algorithm, when viewed as a function on the prefix sum vector, is monotonic with respect to $\preceq$.

Lemma 5.6: *Let S be an arbitrary subset of $[n]$. Let p and q denote the prefix sum vectors associated with token distributions b and c, respectively. Let p' and q' denote the prefix sum vectors associated with the token distributions obtained after performing an S-step on distributions b and c, respectively. If $p \preceq q$, then we have $p' \preceq q'$.*

Proof: Consider any i in $[n]$. If $p(i)$ is less than $q(i)$, then $p'(i) \leq q(i) - 1 \leq q'(i)$. Otherwise, we have $p(i) = q(i)$. By Lemma 5.2, if $q'(i)$ is $q(i)-1$, then $2q(i) > q(i-1)+q(i+1)$. It then follows from the hypothesis of the lemma that $2p(i) > p(i-1)+p(i+1)$, which together with Lemma 5.2 implies that $p'(i)$ is $p(i) - 1$. Thus, the desired claim holds. □

Corollary 5.6.1: *Consider a partial algorithm $\mathcal{B}$. Let p_0 and q_0 denote the prefix sum vectors at the start of step 0 when the initial token distributions are b and c, respectively. If $p_0 \preceq q_0$, then the number of rounds taken by $\mathcal{B}$ to balance b is at most that taken to balance c.* □

Corollary 5.7.1 states that if $\mathcal{B}$ covers $\mathcal{C}$, then $\mathcal{B}$ balances any distribution at least as quickly as $\mathcal{C}$ does.

Lemma 5.7: *Let $\mathcal{B}$ and $\mathcal{C}$ be two partial algorithm such that $\mathcal{B}$ covers $\mathcal{C}$. Given an initial token distribution, let p_t and q_t denote the prefix sum vectors at the start of step t of $\mathcal{B}$ and $\mathcal{C}$, respectively. Then, for each step t, $p_t \preceq q_t$.*

Proof: The proof is by induction on step t. The induction base is trivial since $p_0 = q_0$. For the induction hypothesis, we assume that $p_t \preceq q_t$. Consider step t of $\mathcal{B}$ and $\mathcal{C}$.

Let $\mathcal{D}$ be a partial algorithm that is identical to $\mathcal{C}$ except that $\mathcal{D}(t) = \mathcal{B}(t)$. Let r represent the prefix sum vector obtained after step t of $\mathcal{D}$. Since $\mathcal{D}(t) \supset \mathcal{C}(t)$, it follows from Lemma 5.2 that $r \preceq q_{t+1}$. By Lemma 5.6 and the induction hypothesis, it follows that $p_{t+1} \preceq r$. By the transitivity of $\preceq$, it follows that $p_{t+1} \preceq q_{t+1}$. □

Corollary 5.7.1: *Let $\mathcal{B}$ and $\mathcal{C}$ be two partial algorithm such that $\mathcal{B}$ covers $\mathcal{C}$. For any* *initial token distribution b, the number of rounds taken by $\mathcal{B}$ to balance b is at most that taken by $\mathcal{C}$.* □

Given a nonnegative integer h, consider the set $U(h)$ of token distributions with discrepancy h. Let $P(h)$ denote the set of prefix sum vectors associated with the distributions in $U(h)$. It is easy to see that $f(h) = (2h, h, \ldots, h, 0)$ is the distribution whose prefix sum vector $g(h) = (h, h, \ldots, h, 0)$ is the unique least upper bound (with respect to $\preceq$) of $P(h)$. It thus follows from Corollary 5.6.1 that the number of rounds taken by a partial algorithm $\mathcal{B}$ to balance any distribution in $U(h)$ is at most the number of rounds taken by $\mathcal{B}$ to balance $f(h)$. We now place an upper bound on the number of rounds taken by any partial algorithm to balance $f(h)$.

Lemma 5.8: *For any nonnegative integer h, the number of rounds taken by any partial algorithm $\mathcal{B}$ to balance $f(h)$ is at most $2h + n - 1$.*

Proof: For any i, let the ith round of $\mathcal{B}$ consist of the steps $[r_i + 1, r_{i+1}]$. In order to establish the desired claim, we construct a partial algorithm $\mathcal{C}$ that is covered by $\mathcal{B}$. Since the rounds of $\mathcal{C}$ may differ from those of $\mathcal{B}$, to avoid ambiguity, we refer to $[r_i + 1, r_{i+1}]$ as interval i.

Given interval i and a node j, let $y_{i,j}$ be the smallest integer such that j is in $\mathcal{B}(y_{i,j})$. We now define $\mathcal{C}$ as follows. For each interval i, and each step t in interval i, node j is in $\mathcal{C}(t)$ if and only if: (i) t is $y_{i,j}$ and (ii) j equals $t - 2k$ for some $k \le \min\{\lceil t/2 \rceil, h\}$ (i.e., j has the same parity as t). It follows directly from the definition that $\mathcal{C}$ is a partial algorithm and that $\mathcal{B}$ covers $\mathcal{C}$. We now show that $\mathcal{C}$ balances $f(h)$ before the start of interval $2h + n - 1$.

We mark the h excess tokens on node 0 with the labels 0 through $h - 1$ from the top. We show that during the execution of $\mathcal{C}$ the following property holds: at the start of interval t, if $i \le \min\{\lceil t/2 \rceil, h\}$, token i is at node $\min\{t - 2i, n - 1\}$; otherwise, token i is at node 0. The proof is by induction on $t \le 2h + n$. The induction base is trivial. For the induction hypothesis, we assume that the above statement holds at the start of interval t.

Consider interval t. By the definition of $\mathcal{C}$, we obtain that in interval t, if $t-j$ is even, then node j sends token $(t-j)/2$ to node j; otherwise, node j does not send any token. Thus, each node j sends at most one token to node $i+1$ in any interval. Furthermore, by the induction hypothesis, if $t-j$ is even then node j has token $(t-j)/2$ while node $j+1$ has no marked token, thus completing the induction step.

It follows from the aforementioned property that $\mathcal{C}$ balances $f(h)$ before the start of interval $2h+n-1$. Hence, by Corollary 5.7.1, $\mathcal{B}$ balances $f(h)$ within $2h+n-1$ rounds. ∎

The following lemma shows that $g(D(b))$ is an upper bound (with respect to $\preceq$) on the initial prefix sum vector of c.

Lemma 5.9: *For any initial token distribution b, we have $p_0 \preceq g(D(b))$.*

Proof: By the definition of D and p_0, for each i in $[n]$, $p_0(i)$ is at most $D(b)$. Moreover, since ρ is an integer, $p_0(n-1)$ is zero. It thus follows from the definition of g that $p_0 \preceq g(D(b))$. ∎

The upper bound on the time complexity of a partial algorithm now follows from Corollary 5.6.1 and Lemma 5.9.

Lemma 5.10: *Given any initial token distribution b, the number of rounds taken by any partial algorithm to balance b is at most $2D(b)+n-1$.*

Proof: By Lemma 5.9, $p_0 \preceq g(D(b))$. Therefore, by Corollary 5.6.1, the number of rounds taken to balance b is at most that taken to balance $f(D(b))$. By Lemma 5.8, the number of rounds taken to balance $f(D(b))$ is at most $2D(b)+n-1$. The desired claim follows. ∎

We now place a bound on the number of token transmissions before balancing a distribution b. Whenever a node i sends a token in step t, we have $p_{t+1}(i) = p_t(i) - 1$. Therefore, the total number of token transmissions by any partial algorithm is exactly $T(b)$, which, by Lemma 5.1, is optimal with respect to all unidirectional algorithms.

Lemma 5.11: *Given an initial token distribution b, the number of token transmissions by any partial algorithm is $T(b)$.* □

5.4.2 Complexity of Algorithm $\mathcal{A}$

Every step of $\mathcal{A}$ after step 0 is an $[n]$-step. It thus follows from Lemma 5.10 that the number of steps taken to balance any distribution b with an integral average is at most $2D(b) + n$.

We now consider the message complexity of $\mathcal{A}$. In step 0 of the algorithm, n *height* messages are transmitted. The number of *update* messages transmitted is at most the total number of token transmissions since an *update* message is sent by a node i in step t only if i sends a token in step t. By Lemma 5.11, the number of token transmissions is at most $T(b)$. Hence the total number of message transmissions is at most $2T(b) + n$. This completes the proof of the following theorem.

Theorem 5.1: *Consider the synchronous model of a ring network with n processors model. If the initial token distribution is b, then the number of steps taken by $\mathcal{A}$ to balance b is at most $2D(b) + n$. The number of token transmissions and the number of message transmissions are $T(b)$ and $2T(b) + n$, respectively.* □

5.5 Analysis for asynchronous rings

In this section, we analyze $\mathcal{A}$ under an asynchronous model of computation. We consider the ring network as consisting of $3n$ different components: n nodes given by the set $[n]$ and $2n$ directed edges given by the set $\{(i, i+1), (i, i-1)\}$. As defined in Section 5.2, each step of a node consists of sending a constant number of messages to its neighbors together with performing a small number of local operations. Each edge (i, j) is a directed channel that transmits messages from i to j in FIFO order. At any instant, there may be several messages in transit from i to j on edge (i, j). Each step of edge (i, j) consists of delivering the first message (if any) in FIFO order among the messages currently in transit from i to j.

We model asynchrony by means of an adversary $\mathcal{X}$ that schedules the components of the network over a sequence of steps. In step t, each component in a set $\mathcal{X}(t)$ of components chosen by the adversary executes its next step simultaneously. Given adversaries $\mathcal{X}_1$ and $\mathcal{X}_2$, we say that $\mathcal{X}_1$ is *weaker* (resp., *stronger*) than $\mathcal{X}_2$ if for all t, $\mathcal{X}_1(t)$ is a superset (resp., subset) of $\mathcal{X}_2(t)$. The notions of an adversary and that of weakness generalize the notions of a partial algorithm and that of covering defined in Section 5.4. Indeed, we establish our results for the asynchronous model by generalizing some of the claims of Section 5.4.

As mentioned above, when an edge is scheduled, the first message (if any) in FIFO order is delivered to the destination node. In the definition of $\mathcal{A}$, there are some operations that are performed at the node on receipt of a message. (An example of such an operation is changing the value of $y(i)$ at node i on receipt of an *update* message.) Such operations can be executed either during the scheduling of the edge delivering the particular message or at the next scheduling of the destination node, as determined by the adversary.

Given an adversary, we define a *round* to consist of a minimal sequence of steps in which each component of the network is scheduled at least once by the adversary. The sequence of steps is partitioned into a sequence of non-overlapping rounds. The time complexity of an algorithm is defined to be maximum, over all adversaries, of the number of rounds taken to balance the ring. The message complexity of an algorithm is the maximum, over all adversaries, of the number of messages transmitted by the algorithm.

We now begin the analysis of $\mathcal{A}$ under the asynchronous model defined above. For any $t \geq 0$ and any i in $[n]$, let $u_t(i)$ denote the number of tokens in transit along edge $(i, i+1)$ at the start of step t. In analogy to Equation 5.1, we define two notions of prefix sums. For each t in $\mathbf{N}$, we define p_t and q_t as follows:

$$\begin{aligned} p_t(i) &= \sum_{0 \leq j \leq i} (w_t(j) + u_t(j) - \rho) \text{ and} \\ q_t(i) &= p_t(i) - u_t(i) \end{aligned}$$

for all i in $[n]$. We refer to p_t and q_t as the *upper prefix sum vector* and the *lower prefix sum vector*, respectively. Let $\pi_t(i)$ denote the last step $t' < t$ such that a *height* or an *update* message sent by $i+1$ in step t' is received by i in some step before step t. If no *height* or *update* message is received by i in any of the first t steps, we set $\pi_t(i)$ to -1. For convenience, we let $q_{-1}(i)$ equal ∞ for all i.

Lemmas 5.12, 5.13, 5.14, and 5.15 generalize Lemmas 5.3, 5.5, 5.2, and 5.6, respectively. The proofs of Lemmas 5.12 and 5.13 follow the same lines as the proofs of Lemmas 5.3 and 5.5, respectively.

Lemma 5.12: *Consider the execution of $\mathcal{A}$ against an adversary $\mathcal{X}$. Assume that $0 \preceq q_s$ for all $s \leq t$. If i is in $\mathcal{X}(t)$ and $2q_t(i) > p_t(i-1) + q_{\pi_t(i)}(i+1)$, then $q_{t+1}(i)$ is $q_t(i) - 1$; otherwise, $q_{t+1}(i)$ is $q_t(i)$.*

Proof: Consider any $s \leq t$. Since $0 \preceq q_s$, $q_s(0)$ is nonnegative, and hence $w_s(0)$ is at least ρ. Moreover, by definition, $p_s(n-1)$ is 0. Since $0 \leq q_s(n-1) \leq p_s(n-1)$, $q_s(n-1)$ and $u_s(n-1)$ are both 0. Since $p_s(n-2)$ is nonnegative, $w_s(n-1)$ is at most ρ. Therefore, no token is sent from node $n-1$ to node 0 in step t. It follows that for each i in $[n]$, if node i sends a token to node $i+1$, then $q_{t+1}(i)$ is $q_t(i) - 1$; otherwise, $q_{t+1}(i)$ is $q_t(i)$.

We now show that node i sends a token to node $i+1$ if and only if i is in $\mathcal{X}(t)$ and $q_t(i) - p_t(i-1)$ is greater than $q_{\pi_t(i)}(i+1) - q_t(i)$. It follows from the definitions of q_t and p_t that $w_t(i)$ equals $q_t(i) - p_t(i-1)$. If $\pi_t(i)$ equals -1, then it follows from the definition of $\pi_t(i)$ that the value of the variable $z(i)$ (see Section 5.2) at the start of step t is ∞. Since the values of $x(i)$, $y(i)$ and $q_t(i)$ are all finite, we obtain that if $\pi_t(i)$ is -1 then at the start of step t, $z(i) + x(i) - y(i) = \infty = q_{\pi_t(i)}(i+1) - q_t(i)$. We now consider the case when $\pi_t(i)$ does not equal -1. By the definition of $\pi_t(i)$, the value of $z(i)$ at the start of step t is finite. Moreover, by the definitions of the variables $x(i)$, $y(i)$, and $z(i)$ of Section 5.2 and the fact that no token is sent from node $n-1$ to node n in any step $s \leq t$, we obtain that the value of the expression $z(i) + x(i) - y(i)$ at the start of step t equals

$$
\begin{aligned}
& q_0(i+1) - q_0(i) + q_0(i) - q_t(i) - (q_0(i+1) - q_{\pi_t(i)}(i+1)) \\
= \quad & q_{\pi_t(i)}(i+1) - q_t(i).
\end{aligned}
$$

Therefore, in any step t, if node i is scheduled by the adversary, then node i sends a token to $i+1$ if and only if $q_t(i) - p_t(i-1)$ is greater than $q_{\pi_t(i)}(i+1) - q_t(i)$. The desired claim follows. ∎

Lemma 5.13: *Given any adversary, $0 \preceq q_t$ and $0 \preceq p_t$ hold for all t in $\mathbf{N}$.*

Proof: The proof is by induction on t. The induction basis follows from Lemma 5.4.

For the induction step, consider step t. By the induction hypothesis, $0 \preceq q_t$. We first show that $0 \preceq q_{t+1}$. By Lemma 5.12, we have: if $2q_t(i) > p_t(i-1) + q_{\pi_t(i)}(i+1)$, then $q_{t+1}(i)$ is $q_t(i)-1$; otherwise, $q_{t+1}(i)$ is $q_t(i)$. In either case, since $p_t(i)$, $p_t(i-1)$, and $q_{\pi_t(i)}(i+1)$ are nonnegative integers by the induction hypothesis, we obtain that $q_{t+1}(i)$ is nonnegative for all i. Since $q_{t+1} \preceq p_{t+1}$, it follows that $0 \preceq p_{t+1}$, thus establishing the induction step. ∎

Lemmas 5.12 and 5.13 together imply the following lemma.

Lemma 5.14: *Given any adversary $\mathcal{X}$, if i is in $\mathcal{X}(t)$ and $2q_t(i) > p_t(i-1)+q_{\pi_t(i)}(i+1)$, then $q_{t+1}(i)$ is $q_t(i) - 1$; otherwise, $q_{t+1}(i)$ is $q_t(i)$.* ∎

Given a fixed initial distribution of tokens and two different adversaries, we now relate the prefix sum vectors obtained after t steps of $\mathcal{A}$ against the two adversaries. Lemma 5.15 states that both the upper and lower prefix sum vectors associated with the weaker adversary are lower bounds (with respect to $\preceq$) on the upper and lower prefix sum vectors associated with the stronger adversary.

Lemma 5.15: *Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two adversaries such that $\mathcal{X}_1$ is weaker than $\mathcal{X}_2$. Given an initial token distribution, let p_t^1 (resp., q_t^1) denote the upper (resp., lower) prefix sum vector at the start of step t of $\mathcal{A}$ against adversary $\mathcal{X}_1$, and let p_t^2 (resp., q_t^2) denote the* *upper (resp., lower) prefix sum vector at the start of step t of $\mathcal{A}$ against adversary $\mathcal{X}_2$. For each step t, we have $q_t^1 \preceq q_t^2$ and $p_t^1 \preceq p_t^2$.*

Proof: Let $\alpha_t(i)$ and $\beta_t(i)$ denote the value of $\pi_t(i)$ under adversaries $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. We prove by induction on t that: (i) $q_t^1 \preceq q_t^2$, (ii) $p_t^1 \preceq p_t^2$, and (iii) for all i, $q^1_{\alpha_t(i)}(i+1) \le q^2_{\beta_t(i)}(i+1)$. The induction base is trivial since $q_0^1 = q_0^2$ and $p_0^1 = p_0^2$ and $\alpha_t(i) = \beta_t(i) = -1$ for all i. For the induction hypothesis we assume that (i), (ii), and (iii) hold for all steps less than or equal to t.

We first show that $q_{t+1}^1 \preceq q_{t+1}^2$. Consider any i in $[n]$. If $q_t^1(i) < q_t^2(i)$, then it trivially follows from Lemma 5.14 that $q_{t+1}^1(i) \le q_{t+1}^2(i)$. Otherwise, $q_t^1(i) = q_t^2(i)$. In this case, by Lemma 5.14, if $q_{t+1}^2(i) = q_t^2(i) - 1$, then $2q_t^2(i) > p_t^2(i-1) + q_s^2(i+1)$, where s equals $\beta_t(i)$. Let s' equal $\alpha_t(i)$. By the induction hypothesis, $q_{s'}^1(i+1) \le q_s^2(i+1)$ and $p_t^1(i-1) \le p_t^2(i-1)$. Since $q_t^1(i) = q_t^2(i)$, we thus obtain $2q_t^1(i) > p_t^1(i-1) + q_{s'}^1(i+1)$, which together with Lemma 5.14 implies that $q_{t+1}^1(i) = q_t^1(i) - 1$. Thus, we have $q_{t+1}^1 \preceq q_{t+1}^2$.

We next show that $p_{t+1}^1 \preceq p_{t+1}^2$. In order to prove that $p_{t+1}^1(i) \le p_{t+1}^2(i)$, we need only consider the case in which $(i, i+1)$ is in $\mathcal{X}_1(t)$, as otherwise the desired claim follows directly from the induction hypothesis. Accordingly, assume that $(i, i+1)$ is in $\mathcal{X}_1(t)$. Let $u_t^1(i)$ and $u_t^2(i)$ denote the values of $u_t(i)$ associated with adversaries $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. If $u_t^1(i)$ is positive, then $p_{t+1}^1(i) = p_t^1(i) - 1 \le p_t^2(i) - 1 \le p_{t+1}^2(i)$. Otherwise, we have $p_{t+1}^1(i) = p_t^1(i) = q_t^1(i)$, and $p_{t+1}^2(i) \ge p_t^2(i) - u_t^2(i) = q_t^2(i)$. Therefore, $p_{t+1}^1(i)$ is at most $p_{t+1}^2(i)$.

We now complete the induction step by showing that for all i, $q^1_{\alpha_{t+1}(i)}(i+1) \le q^2_{\beta_{t+1}(i)}(i+1)$. If $(i+1, i)$ is not in $\mathcal{X}_2(t)$, then $\alpha_{t+1}(i) \ge \alpha_t(i)$ and $\beta_{t+1}(i) = \beta_t(i)$, and hence the desired claim follows from the induction hypothesis and the trivial inference from Lemma 5.14 that q_j^1 is nonincreasing as j increases. We now consider the case in which $(i+1, i)$ is in $\mathcal{X}_2(t)$. If $\alpha_{t+1}(i) \ne \alpha_t(i)$, then the desired claim holds since $q^1_{\alpha_{t+1}(i)}(i+1) = q^1_{\alpha_t(i)}(i+1) - 1$, while $q^2_{\beta_{t+1}(i)}(i+1) \ge q^2_{\beta_t(i)}(i+1) - 1$. Otherwise, we consider two subcases: $\alpha_{t+1}(i) \ge \beta_{t+1}(i)$ and $\alpha_{t+1}(i) < \beta_{t+1}(i)$. In the first subcase, since q_j^1 is nonincreasing as j increases, $q^1_{\alpha_{t+1}(i)}(i+1) \leq q^1_{\beta_{t+1}(i)}(i+1)$. For the second subcase, we note that since $\alpha_{t+1}(i) = \alpha_t(i)$, no *update* message is received by i in step t under adversary $\mathcal{X}_1$, which implies that no *update* message was sent by node $i+1$ to node i in the interval $[\alpha_{t+1}(i)+1, t-1]$ of steps under adversary $\mathcal{X}_1$. Since every token transmission from node $i+1$ to node $i+2$ is accompanied by an *update* message from node $i+1$ to node i, we obtain that for every t' in $[\alpha_{t+1}(i)+1, t-1]$, $q^1_{t'}(i+1) = q^1_{\alpha_{t+1}(i)}(i+1)$. In particular, $q^1_{\alpha_{t+1}(i)}(i+1) = q^1_{\beta_{t+1}(i)}(i+1)$. Thus, in either of the two subcases, $q^1_{\alpha_{t+1}(i)}(i+1) \leq q^1_{\beta_{t+1}(i)}(i+1)$. The desired claim now follows from the induction hypothesis. □

Corollary 5.15.1: *Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two adversaries such that $\mathcal{X}_1$ is weaker than $\mathcal{X}_2$. For any initial token distribution b, the number of rounds taken by $\mathcal{A}$ to balance b against $\mathcal{X}_1$ is at most the number of rounds taken by $\mathcal{A}$ to balance b against $\mathcal{X}_2$.*

We are now ready to establish the main result for asynchronous rings.

Theorem 5.2: *The number of rounds taken by $\mathcal{A}$ to balance any initial token distribution b is at most $4D(b)+2n-2$. The number of token transmissions is at most $T(b)$ and the number of message transmissions is at most $2T(b)+n$.*

Proof: Given any adversary $\mathcal{X}_1$, we construct a stronger adversary $\mathcal{X}_2$ that schedules each component exactly once in each round, as follows: component α is in $\mathcal{X}_2(t)$ if and only if α is in $\mathcal{X}_1(t)$ and t is the first step in the current round such that α is in $\mathcal{X}_1(t)$. We next construct an adversary $\mathcal{X}_3$ that is stronger than $\mathcal{X}_2$ such that each round of $\mathcal{X}_3$ consists of scheduling the components in the following order: first all edges of the form $(i, i-1)$ in any order, then all the nodes in any order, and finally all edges of the form $(i, i+1)$ in any order.

By the definition of $\mathcal{X}_2$, the number of rounds taken by $\mathcal{A}$ against $\mathcal{X}_1$ is at most that taken by $\mathcal{A}$ against $\mathcal{X}_2$. By the definition of $\mathcal{X}_3$, the number of rounds taken by $\mathcal{A}$ against $\mathcal{X}_3$ equals the number taken by $\mathcal{A}$ in the synchronous model, which is at most $2D(b)+n$ by Theorem 5.1. Moreover, it is easy to see that $\mathcal{X}_3$ can be constructed such that for any t, the number of rounds completed at the start of step t of $\mathcal{X}_3$ is at least half the number completed at the start of step t of $\mathcal{X}_2$. It thus follows from Corollary 5.15.1 that for $\mathcal{A}$, the number of rounds taken against $\mathcal{X}_1$ is at most twice the number taken against $\mathcal{X}_3$. Thus, the number of rounds taken by $\mathcal{A}$ to balance any initial token distribution b is at most $4D(b) + 2n$.

The bounds on the number of token and message transmissions follow as in the synchronous case. $\square$

5.6 Concluding Remarks

In Sections 5.4 and 5.5, we obtained bounds on the time taken for $\mathcal{A}$ to converge to a balanced state. One unfortunate characteristic of the bounds is the additive linear term in the time complexity of $\mathcal{A}$ (see Theorems 5.1 and 5.2). We claim that such an additive linear term is unavoidable for any distributed algorithm. To observe this, consider two initial token distributions b and c that are defined as follows. In both distributions, node 0 has two tokens and every other node except node $\lfloor n/2 \rfloor$ has one token. In distribution b, $\lfloor n/2 \rfloor$ has zero tokens, while in c, $\lfloor n/2 \rfloor$ has one token. The optimal centralized algorithm for either distribution takes at most one step. However, any distributed algorithm takes at least linear time to terminate for at least one of the two distributions, since it takes linear time for some node in the ring to distinguish between the two distributions.

We would like to extend our study to more complicated, yet structured, networks such as the d-dimensional mesh. Although we do not expect the bounds to be as sharp as for the ring, results of a similar flavor are conceivable.

Chapter 6

Dynamic Load Balancing on Arbitrary Networks

6.1 Introduction

We now turn to the dynamic aspect of load balancing. As before, we represent a distributed system by an arbitrary network, and assume that the load consists of independent equally-sized tokens, that may be processed anywhere. The dynamic nature of the problem is modeled by an on-line process that adds new tokens to and deletes old tokens from the system. Since an arbitrary on-line process is difficult to analyze, previous work in this area has typically made certain probabilistic assumptions about the token arrival process (for example, see [85, 111]). We depart from this approach and instead study a simplified scenario of load balancing under a model that does not rely on any probabilistic assumptions about the process of token generation and destruction.

Our model is based on an adversarial model that has been proposed recently for studying routing problems [12, 32]. We assume that an adversary controls the on-line process of token arrival and departure. In each step, the adversary determines the locations and the number of tokens that are to be added to or deleted from the system. Given such a model, a natural question to ask is whether there exist *stable* balancing algorithms, that is, algorithms which ensure that the imbalance of the network does not exceed a fixed (time-independent) bound.

It is easy to see that we need to place certain restrictions on the adversary to allow for the possibility of stable algorithms. In our model, the sole restriction on the adversary is that there must exist $r \leq 1$ such that: in any step, for every set S of nodes, the increase in the imbalance of S due to the actions of the adversary does not exceed r times the number of edges coming out of S. (The imbalance of a set S is the difference between the total number of tokens in S and the product of the number of nodes in S and the average number of tokens per node in the network.) See Section 6.2 for a formal description of the model.

By a straightforward argument based on edge-cuts, we find that there is no stable balancing algorithm if r is allowed to exceed 1. The main result of this chapter is that the multi-port local balancing algorithm defined in Section 4.1.1 is stable for all networks for all $r < 1$. Section 6.3 contains a proof of this result.

6.2 An Adversarial Model

Let $G = (V, E)$ denote a network, where V is the set of nodes and E is the set of bidirectional links. As before, let $w_t(v)$ denote the number of tokens at node v at the start of step t. For any subset S of V, let $w_t(S)$ denote the total number of tokens in S at the start of step t. Let the average number of tokens $(w_t(V)/|V|)$ at the start of step t be denoted by ρ_t. We define the *imbalance* of G at the start of step t to be $\max\{|w_t(v) - \rho_t| : v \in V\}$.

Each step of the computation proceeds in two phases. In the first phase, the balancing phase, one "step" of the balancing algorithm is executed. We assume multi-port communication, whereby each node can send/receive at most one token along each of its incident links.

In the second phase of each step, the adversarial phase, an adversary inserts and/or deletes tokens from the network. Let $e(S)$ denote the number of edges coming out of a set S of nodes. Let $d_t(S)$ denote the net increase in the number of tokens at nodes in set S in the second phase of step t. (Note that $d_t(S)$ may be negative.) An adversary with *rate* r, where $r \geq 0$, can insert and/or delete any number of tokens on any subset of nodes subject to the following constraint for every subset S of nodes:

$$|d_t(S) - (\rho_{t+1} - \rho_t)|S|| \leq r \cdot e(S) \tag{6.1}$$

6.3 Stability of the Multi-Port Algorithm

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

Theorem 6.1: *For any $r < 1$, the multi-port local balancing algorithm is stable for rate r.*

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

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

Lemma 6.1: *If there exists a node with height at least $5n^2d^2/(2\varepsilon)$ at the start of step t, then Φ_{t+1}^+ is at most Φ_t^+. If there exists a node with height at most $-5n^2d^2/(2\varepsilon)$ at the start of step t, then Φ_{t+1}^- is at most Φ_t^-.*

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

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

Assume that there exists a node with height at least $5n^2d^2/(2\varepsilon)$ at the start of step t. We divide V_t^+ into distinct sets in the following way. For any i in $\mathbf{N}$, if $\cup_{0 \le j < i} S_j$ is not equal to V_t^+, then let S_i denote the minimal nonempty set of nodes such that for all u in S_i and v in $V_t^+ \setminus \cup_{0 \le j \le i} S_j$, $w_t(v) - w_t(u)$ is at least $4d$. Let k be the maximum value of i for which S_i is defined. Since $5n^2d^2/(2\varepsilon) > 4nd$, k is positive. Let $S_{\ge i}$ denote $\cup_{j \ge i} S_j$. Let h_i and ℓ_i denote $\max_{u \in S_i}(w_t(u) - \rho_t)$ and $\min_{u \in S_i}(w_t(u) - \rho_t)$, respectively. Thus, $S_0, S_1, \ldots, S_k$ are disjoint sets that satisfy the following property: for $0 \le i < k$, each node in S_i has at least $4d$ tokens less than each node in S_{i+1}. Moreover, $\cup_{0 \le i \le k} S_i$ equals V^+.

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

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

$$-nd^2 + \sum_{i > 0} 2e(S_{\ge i})(\ell_i - h_{i-1} - 2d). \tag{6.2}$$

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

$$\begin{aligned} & \sum_i \sum_{u \in S_i} \phi'_t(u) + 2(w'_t(u) - \rho_t)(d_t(u) - (\rho_{t+1} - \rho_t)) + (d_t(u) - (\rho_{t+1} - \rho_t))^2 \\ \leq \; & (\sum_i (\phi'_t(S_i) + \sum_{u \in S_i} 2h_i(d_t(u) - (\rho_{t+1} - \rho_t)))) + nd^2. \end{aligned}$$

In addition to nodes in V_t^+ that remain in V_{t+1}^+, there may be nodes that do not belong to V_t^+ but belong to V_{t+1}^+. By Equation 6.1, the potential of any such node is at most d^2. Thus, the total increase in potential in the adversarial phase is at most:

$$2nd^2 + \sum_i 2h_i(d_t(S_i) - (\rho_{t+1} - \rho_t)|S_i|). \tag{6.3}$$

It follows from Lemma 6.2 below that the right-hand side of Equation 6.3 is maximized if the adversary adds as many tokens to S_k as the constraint in Equation 6.1 allows, then adds as many tokens to S_{k-1} as the constraint allows and so on. We thus obtain an upper bound on the increase in potential by making the following substitution in Equation 6.3 for all i in $[k+1]$:

$$d_t(S_i) - (\rho_{t+1} - \rho_t)|S_i| = (1-\varepsilon)(e(S_{\geq i}) - e(S_{>i})). \tag{6.4}$$

By Equations 6.2, 6.3, and 6.4, we obtain that the net decrease in potential is at least:

$$\begin{aligned} & -3nd^2 - \sum_{i \geq 0} 2h_i(1-\varepsilon)(e(S_{\geq i}) - e(S_{>i})) + \sum_{i>0} 2e(S_{\geq i})(\ell_i - h_{i-1} - 2d) \\ = \; & -3nd^2 - 2h_0(1-\varepsilon)e(S_{\geq 0}) + \sum_{i>0} 2e(S_{\geq i})(\ell_i - (1-\varepsilon)h_i - \varepsilon h_{i-1} - 2d) \\ = \; & -3nd^2 - 2h_0(1-\varepsilon)e(S_{\geq 0}) - \sum_{i>0} 2e(S_{\geq i})(h_i - l_i) + \sum_{i>0} 2e(S_{\geq i})(\varepsilon(h_i - h_{i-1}) - 2d) \\ \geq \; & -3nd^2 - 4n^2d^2 + \sum_{i>0} 2e(S_{\geq i})(\varepsilon(h_i - h_{i-1}) - 2d) \\ \geq \; & -3nd^2 - 4n^2d^2 + 2\varepsilon(h_k - 4nd) - 2nd^2 \\ = \; & -(4n^2d^2 + 5nd^2 + 8\varepsilon nd) + 2\varepsilon h_k \\ \geq \; & -5n^2d^2 + 2\varepsilon h_k, \end{aligned}$$

for n sufficiently large. (In the first and second equations, we rearrange the summands. For the third equation we note that: (i) $h_i - \ell_i$ is at most $4nd$ for all d, and (ii) the total number of edges in the network is at most $nd/2$. For the fourth equation, we note that: (i) $e(S_{\geq i})$ is at least one for all i, (ii) $e(S)$ is at most $nd/2$ for all S, and (iii) h_0 is at most $4nd$.)

Since there exists a node with height at least $5n^2d^2/(2\varepsilon)$, h_k is at least $5n^2d^2/(2\varepsilon)$. Hence, the net decrease in potential is nonnegative. $\square$

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

Lemma 6.2: *Let $\langle \alpha \rangle$ be a sequence of k nonincreasing nonnegative reals. Let $\langle \beta \rangle$ be a sequence of k nondecreasing reals. Then, an optimal solution to the linear program $\mathcal{P}$:*

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

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

Proof: Given any solution $\langle y \rangle$ to $\mathcal{P}$ and any i in $[k]$, we say that i is *good* if $\sum_{0 \leq j \leq i} y_j$ equals β_i. Let $\langle x^* \rangle$ be defined as follows: x_0^* is β_0 and x_i^* is $\beta_i - \beta_{i-1}$ for all i in $[1,k)$. We note that $\langle x^* \rangle$ is the unique solution that has k good indices.

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

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

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

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

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

Proof of Theorem 6.1: We now show that for all t, Φ_t^+ and Φ_t^- are both at most $n(5n^2d^2/(2\varepsilon) + d)^2$. Let, if possible, t be the first step such that Φ_t^+ is greater than $n(5n^2d^2/(2\varepsilon) + d)^2$. Therefore, there exists at least one node v such that $h_t(v)$ is at least $(5n^2d^2/(2\varepsilon) + d)$. Since the height of a node increases by at most d in the balancing phase and by at most d in the adversarial phase, we obtain that $h_t(v) \le h_{t-1}(v) + 2d$ for all v. Therefore, for all v in V, $h_{t-1}(v)$ is at least $5n^2d^2/(2\varepsilon)$. This implies that by Lemma 6.1, Φ_{t-1}^+ is at least Φ_t^+, which contradicts our choice of t. A symmetric argument establishes that Φ_t^- is at most $n(5n^2d^2/(2\varepsilon) + d)^2$.

Hence, the imbalance at the start of any step t is at most $\sqrt{n(5n^2d^2/(2\varepsilon) + 2d)^2}$, which is at most $3n^{5/2}d^2/\varepsilon$ for n sufficiently large. This establishes the stability of the multi-port algorithm. ∎

6.4 Concluding Remarks

Consider the following variation of the adversarial load balancing problem that is motivated by job scheduling. We are given an arbitrary network and a dynamic token arrival/departure process in which each token represents a job that takes one unit of time to process at any node of the network. In each step, we allow an adversary to create at most n new tokens and distribute them among the nodes of the network arbitrarily subject to the constraint that for every subset S of nodes, $d_t(S) - |S| \leq r \cdot e(S)$, where $r \leq 1$. (Recall that $e(S)$ denotes the number of edges coming out of S.) A single step of any scheduling algorithm consists of: (i) a balancing phase in which each node can send and/or receive at most one token along each of its incident edges, and (ii) a processing phase in which each node may process at most one of its tokens. Once a token is processed, it is deleted from the network. We say that a scheduling algorithm is stable if each token is processed within a bounded (time-independent) number of steps.

It is easy to see that if we strengthen the adversary by either allowing more than n tokens to be added per step or letting r exceed 1, then no stable algorithm exists. On the other hand, we can show that the local load balancing algorithm can be combined with any work-preserving processing strategy to obtain a stable algorithm against all adversaries provided $r < 1$. (In a work-preserving strategy, a node is idle only if it has no tokens to process.)

The main technical problem left open by this chapter concerns the stability of local load balancing algorithm for rate 1. We conjecture that the algorithm is stable for rate 1 too. A proof technique different from the one used in this chapter, however, may be needed to establish such a result (if it holds).

Conclusions

In this dissertation, we have developed and analyzed mechanisms for sharing resources in distributed systems. We have demonstrated that simple local algorithms can efficiently solve certain problems related to sharing memory and processors in a distributed system. We now conclude by summarizing our results and by presenting a few directions for future research.

Sharing Memory

In the first half of the dissertation, we considered the question of how to share memory in a distributed system. Since the general problem is quite complicated, we focused on two specific aspects of the problem: memory contention and faults. Our main result is a protocol that provides fast access to shared objects in an environment in which memory contention can be unlimited and a constant fraction of the nodes and communication links can be faulty at any time. We showed that if each access request is chosen according to a fixed probability distribution over the set of objects, then our protocol reaches a steady state in $O(\log n)$ steps. In the steady state, each request is satisfied in expected $O(1)$ steps and the throughput of the protocol is asymptotically optimal. We also proved that the protocol continues to remain in a steady state if changes in the access pattern are moderate.

There are a number of directions in which our work can be extended. The most pressing need, perhaps, is that of proposing a model and developing some new techniques to incorporate nonuniform communication costs. One simplifying, yet useful, model for nonuniform networks is to set the cost of sending a message of length ℓ from a node u to v as some fixed function of u, v, and ℓ. In addition, as done in this dissertation, we may place a bound on the number of messages that a node can send or receive per unit time. Even though the preceding model does not consider issues such as internal network congestion, the following interesting questions can be posed for the model:

- Placement of copies: In order to support high degrees of concurrency, on-line replication will play a central role in any efficient access scheme. The results in Chapters 2 and 3 show that if the cost model is uniform, random hashing can be used to determine where to place the copies. For a nonuniform network, however, we would like the placement of copies of any object to be determined by the current distribution of requests for the object across the network. This leads us to a well-studied problem in combinatorial optimization, the facility location problem (e.g., see [40]). While the facility location problem is NP-complete, constant-factor approximation algorithms for important special cases have been obtained recently [84, 106]. All of the current solutions devised for this problem, however, assume centralized control; it would be interesting to obtain solutions that can be adapted to a dynamic and distributed environment. (For recent related work in this area on specific network topologies, see [89].)

- Locating nearby copies of objects: In a nonuniform network, changes in the pattern of accesses across the network may cause changes in the locations of object copies. Hence, there is a need for a mechanism to dynamically locate nearby copies of objects. While this problem has been well-studied (see [96] for early work, and [19, 26, 103] for recent results), existing solutions either hold for restricted cost models only or suffer from large overhead in storage requirements. (By overhead in storage, we refer to the memory required to store information about the locations of the objects.) Since the above problem has direct applications to the Internet [66, 117], an efficient solution would be of great interest.

Sharing Processors

In the second part of the dissertation, we analyzed the effectiveness of a local balancing strategy in which each node repeatedly balances its load with its neighbors. Our main results in this part concern the static aspect of the problem, i.e., we assume that the total workload does not change with time. We showed in Chapter 4 that the local balancing approach is worst-case optimal for all networks. In Chapter 5, we improved the preceding result for the special case of ring networks by showing that the local balancing approach is optimal (up to an additive linear term) for all distributions on the ring. Our results for static load balancing hold in asynchronous environments as well.

Our work on static load balancing leaves a number of interesting open questions. Can we improve the bounds obtained in Chapter 4 for arbitrary networks? In particular, we would like to determine whether the multi-port algorithm balances any network G with any initial distribution b in $O(\mathrm{OPT}(b)) + f(G)$ steps for some $f(G)$ independent of b, where $\mathrm{OPT}(b)$ is the time taken by an optimal centralized algorithm to balance b on G. Recall that such a result was derived in Chapter 5 for the case of ring networks. A less ambitious goal is to extend the techniques of Chapter 5 to obtain similar bounds for regular topologies closely related to the ring, e.g., fixed-dimensional meshes.

We concluded the second half of the dissertation by showing that the local balancing algorithm is a good candidate for dynamic load balancing as well. Our result in this area, however, is limited in scope. An interesting enhancement of the basic load balancing problem is the following scheduling problem that considers dynamic job arrival. Let $G = (V, E)$ be an arbitrary network with arbitrary capacities on edges. Given an on-line stream of jobs indexed by $\mathbf{N}$, job j characterized by the triple (a_j, v_j, t_j) such that: (i) job j arrives at node v_j at time a_j, and (ii) t_j is the execution time of job j (on all nodes), the *job scheduling problem* is to allocate the jobs such that the average response time is minimized. This problem has been studied under a model in which edge capacities are infinite [7, 21, 46]. For our model, however, results to date are limited to specific instances of the problem [71].

Appendix A

Tails of Probability Distributions

Theorems A.1 and A.2 provide bounds on the tails of the binomial and hypergeometric distributions, respectively.

Theorem A.1 ([37]): *Let X be a random variable drawn from $B(n,p)$, i.e., X is the number of successes in n independent Bernoulli trials, where each trial succeeds with probability p. Then,*

$$\begin{aligned}
\Pr[X \leq (1-\varepsilon)np] &\leq e^{-\varepsilon^2 np/2},\ 0 \leq \varepsilon \leq 1 && \text{(A.1)}\\
\Pr[X \geq (1+\varepsilon)np] &\leq e^{-\varepsilon^2 np/3},\ 0 \leq \varepsilon \leq 1 && \text{(A.2)}\\
\Pr[X \geq (1+\varepsilon)np] &\leq [e^{\varepsilon}(1+\varepsilon)^{-(1+\varepsilon)}]^{np} && \text{(A.3)}
\end{aligned}$$

$\square$

Theorem A.2 ([70, 38]): *Let S be a set of s balls, T be a subset of S, $t = |T|$, and $p = t/s$. Let s' balls be chosen uniformly at random from S, and t' be the random variable representing the number of balls that are chosen from T. Then, for any real $\varepsilon \geq 0$,*

$$\begin{aligned}
\Pr[t' \geq (p+\varepsilon)s'] &\leq e^{-2\varepsilon^2 s'}, \text{ and}\\
\Pr[t' \leq (p-\varepsilon)s'] &\leq e^{-2\varepsilon^2 s'}.
\end{aligned}$$

Proof: By [38, 70],

$$\Pr[t' \geq (p+\varepsilon)s'] \leq e^{-2\varepsilon^2 s'}.$$

The lower bound on t' can be proved by using the upper bound on $s' - t'$. Thus,

$$\Pr[t' \leq (p - \varepsilon)s'] = \Pr[s' - t' \geq (1 - p + \varepsilon)s'] \leq e^{-2\varepsilon^2 s'}.$$

$\square$

Appendix B

Martingales

The theory of martingales provides a useful tool for analyzing certain random processes that are not completely independent. Our presentation in this appendix is based on that of [8]. A martingale is a sequence $X_0, \ldots, X_m$ of random variables so that for $0 \leq i < m$, $E[X_{i+1} \mid X_i] = X_i$. We use Azuma's inequality to obtain bounds on large deviations for martingales.

Theorem B.1 (Azuma's Inequality [8]): *Let $X_0, \ldots, X_k$ be a martingale with $|X_{i+1} - X_i| \leq 1$, for all $0 \leq i < k$. Then for real $\lambda > 0$,*

$$\Pr\left[|X_k - X_0| > \lambda\sqrt{k}\right] < 2e^{-\lambda^2/2}.$$

$\square$

The following theorem identifies certain conditions that are sufficient for applying Azuma's inequality.

Theorem B.2 ([8]): *Let $\Omega = A^B$ denote the set of functions $g : B \to A$. Fix a gradation $\emptyset = B_0 \subset B_1 \subset \cdots \subset B_m = B$. Let L be a function from Ω to $\mathbf{R}$. Define a martingale $X_0, \ldots, X_m$ by setting*

$$X_i(h) = E[L(g) \,|\, g(b) = h(b) \text{ for all } b \in B_i].$$

We say that L *satisfies the Lipschitz condition if the following holds for all* i*: whenever* h *and* h' *differ only on* $B_{i+1} - B_i$*, we have* $|L(h') - L(h)| \leq 1$*. If* L *satisfies the Lipschitz condition, then* $|X_{i+1}(h) - X_i(h)| \leq 1$ *for all* $0 \leq i < m$ *and* $h \in \Omega$*.* $\square$

Appendix C

Technical Inequalities

In this appendix, we prove certain inequalities concerning functions f and g of Section 2.4.2 and function ϕ of Section 4.4.

C.1 Expected Number of Non-Singletons and Non-Pairs

Recall that $f(m,n)$ and $g(m,n)$ are defined as follows:

$$\begin{aligned} f(m,n) &= m\left(1-\left(1-\frac{1}{n}\right)^{m-1}\right), \text{ and} \\ g(m,n) &= m\left(1-\left(1-\frac{1}{n}\right)^{m-1} - \frac{m-1}{n}\left(1-\frac{1}{n}\right)^{m-2}\right). \end{aligned}$$

Lemma C.1: *For all integers m and n such that $3 \le m \le n$, we have*

$$m^2/3n \le f(m) \le m^2/n.$$

Proof: By definition,

$$f(m) = m(1-(1-1/n)^{m-1}).$$

Since $(1-1/n)^{m-1} \ge 1-(m-1)/n$,

$$\begin{aligned} f(m) &\le m(1-1+(m-1)/n) \\ &\le m^2/n. \end{aligned}$$

For the lower bound, since $(1-1/n)^{m-1} \leq 1 - \binom{m-1}{1}/n + \binom{m-1}{2}/n^2$,

$$\begin{aligned} f(m) &\geq m(1 - 1 + \binom{m-1}{1}/n - \binom{m-1}{2}/n^2) \\ &\geq (m(m-1)/n)(1 - (m-2)/2n) \\ &\geq m(m-1)/(2n) \\ &\geq m^2/3n. \end{aligned}$$

In the penultimate derivation, we use $(m-2)/2n \leq 1/2$, and in the last derivation, we use $(m-1)/2 \geq m/3$ for $m \geq 3$. □

Lemma C.2: *For all integers m and n such that $6 \leq m \leq n$, we have*

$$m^3/12n^2 \leq g(m) \leq m^3/n^2.$$

Proof: By definition,

$$g(m) = m(1 - (1 - 1/n)^{m-1} - ((m-1)/n)(1 - 1/n)^{m-2}.$$

Since $(1-1/n)^{m-1} \geq 1 - (m-1)/n$ and $(1-1/n)^{m-2} \geq 1 - (m-2)/n$,

$$\begin{aligned} g(m) &\leq m(1 - 1 + \binom{m-1}{1}/n - (m-1)/n + (m-1)\binom{m-2}{1}/n^2) \\ &\leq m^3/n^2. \end{aligned}$$

For the lower bound,

$$\begin{aligned} g(m) &\geq m(1 - 1 + \binom{m-1}{1}/n - \binom{m-1}{2}/n^2 + \binom{m-1}{3}/n^3 - \binom{m-1}{4}/n^4 \\ &\quad -(m-1)/n + (m-1)\binom{m-2}{1}/n^2 - (m-1)\binom{m-2}{2}/n^3 \\ &\quad +(m-1)\binom{m-2}{3}/n^4 - (m-1)\binom{m-2}{4}/n^5) \\ &\geq m((m-1)(m-2)/2n^2 - (m-1)(m-2)(m-3)/3n^3 \\ &\quad +(m-1)(m-2)(m-3)(m-4)/24n^4 - (m-1)\binom{m-2}{4}/n^5) \\ &\geq (m(m-1)(m-2)/n^2)(1/2 - (m-3)/3n) \\ &\geq m(m-1)(m-2)/6n^2 \\ &\geq m^3/12n^2. \end{aligned}$$

In the last derivation we use $(m-1)(m-2) \geq m^2/2$ for $m \geq 6$. $\square$

Lemma C.3: *If n and m are integers such that $2\sqrt{n} \leq m \leq n$, then for all real $x \geq 0$,*

$$f(m(1+x)) \leq (1+x)^2 f(m).$$

Proof: By the definition of f,

$$f(m(1+x)) = m(1+x)(1-(1-1/n)^{m(1+x)-1}).$$

We establish the desired inequality by proving that $(1-(1-1/n)^{m-1+mx}) \leq (1+x)(1-(1-1/n)^{m-1})$, which is equivalent to showing that $(1-1/n)^{m-1+mx} \geq (1-1/n)^{m-1}(1+x)-x$. Since $(1-1/n)^{mx} \geq (1-mx/n)$,

$$\begin{aligned}
(1-1/n)^{m-1+mx} &\geq (1-1/n)^{m-1}(1-mx/n) \\
&= (1-1/n)^{m-1}(1+x) - x(1+m/n)(1-1/n)^{m-1} \\
&\geq (1-1/n)^{m-1}(1+x) - x.
\end{aligned}$$

The last derivation follows from the observation that for $m \geq 2\sqrt{n}$, $(1+m/n) \leq (1-1/n)^{1-m}$. $\square$

Corollary C.3.1: *If n and m are integers and x is real such that $0 \leq x < 1$ and $2\sqrt{n} \leq m(1-x) \leq m \leq n$, then:*

$$\begin{aligned}
f(m(1+x)) &\leq (1+x)^2 f(m), \text{ and} \\
f(m(1-x)) &\geq (1-x)^2 f(m).
\end{aligned}$$

Proof: The first inequality follows directly from Lemma C.3. The second inequality is proved by applying Lemma C.3 substituting $(m(1-x), 1/(1-x)-1)$ for (m, x). $\square$

Lemma C.4: *If n and m are integers such that $n \geq 9$ and $10\sqrt{n} \leq m \leq n$, then for real $x \geq 0$,*

$$g(m(1+x)) \leq (1+x)^4 g(m)$$

Proof: By the definition of g,

$g(m(1+x)) = m(1+x)(1-(1-1/n)^{m(1+x)-1} - ((m(1+x)-1)/n)(1-1/n)^{m(1+x)-2})$.

We establish the desired inequality by showing that $(1-(1-1/n)^{m(1+x)-1} - ((m(1+x)-1)/n)(1-1/n)^{m(1+x)-2}) \leq (1+x)^3(1-(1-1/n)^{m-1} - ((m-1)/n)(1-1/n)^{m-2})$. This is equivalent to showing that $(1-1/n)^{m-1+mx} + ((m(1+x)-1)/n)(1-1/n)^{m-2+mx} \geq (1+x)^3(1-1/n)^{m-1} + (1+x)^3((m-1)/n)(1-1/n)^{m-2} - x^3 - 3x^2 - 3x$.

$$\begin{aligned}
& (1-1/n)^{m-1+mx} + ((m(1+x)-1)/n)(1-1/n)^{m-2+mx} \\
\geq\ & (1-1/n)^{m-1}(1-mx/n) + ((m(1+x)-1)/n)(1-1/n)^{m-2} \\
& -((mx(m(1+x)-1))/n^2)(1-1/n)^{m-2} \\
\geq\ & (1-1/n)^{m-1}(1-mx/n) + ((m-1)(1+x)/n)(1-1/n)^{m-2} \\
& +(x/n)(1-1/n)^{m-2} - ((mx(m(1+x)-1))/n^2)(1-1/n)^{m-2} \\
=\ & (1+x)^3(1-1/n)^{m-1} - (x^3+3x^2+3x+mx/n)(1-1/n)^{m-1} \\
& +(1+x)^3((m-1)/n)(1-1/n)^{m-2} - (x^3+3x^2+2x)((m-1)/n)(1-1/n)^{m-2} \\
& +(x/n - m^2x/n^2 - m^2x^2/n^2 + mx/n^2)(1-1/n)^{m-2} \\
\geq\ & (1+x)^3(1-1/n)^{m-1} - (x^3+3x^2+3x+mx/n)(1-1/n)^{m-2} \\
& +(1+x)^3((m-1)/n)(1-1/n)^{m-2} - (x^3+3x^2+2x)((m-1)/n)(1-1/n)^{m-2} \\
& +(x/n - m^2x/n^2 - m^2x^2/n^2 + mx/n^2)(1-1/n)^{m-2} \\
\geq\ & (1+x)^3(1-1/n)^{m-1} + (1+x)^3((m-1)/n)(1-1/n)^{m-2} \\
& -(1-1/n)^{m-2}(x^3+3x^2+3x+mx/n+mx^3/n-x^3/n \\
& +3mx^2/n - 3x^2/n + 2mx/n - 2x/n - x/n \\
& +m^2x/n^2 + m^2x^2/n^2 - mx/n^2) \\
\geq\ & (1+x)^3(1-1/n)^{m-1} + (1+x)^3((m-1)/n)(1-1/n)^{m-2} \\
& -(1-1/n)^{m-2}(x^3(1+m/n-1/n) \\
& +3x^2(1+m/n-1/n+m^2/3n^2) + 3x(1+m/n-1/n+m^2/3n^2-m/3n^2)) \\
\geq\ & (1+x)^3(1-1/n)^{m-1} + (1+x)^3((m-1)/n)(1-1/n)^{m-2} - (x^3+3x^2+3x).
\end{aligned}$$

The last derivation follows from the inequalities $(1 + (m-1)/n + m^2/3n^2) \leq (1 - 1/n)^{-(m-2)}$ for $n \geq 9$ and $10\sqrt{n} \leq m \leq n$. $\square$

Corollary C.4.1: *If n and m are integers and x is real such that $0 \leq x < 1$, $n \geq 9$, and $10\sqrt{n} \leq m(1-x) \leq m \leq n$, then:*

$$\begin{aligned} g(m(1+x)) &\leq (1+x)^4 g(m), \text{ and} \\ g(m(1-x)) &\geq (1-x)^4 g(m). \end{aligned}$$

Proof: The first inequality follows directly from Lemma C.4. The second inequality is proved by applying Lemma C.4 substituting $(m(1-x), 1/(1-x) - 1)$ for (m, x). $\square$

C.2 The Potential Function of Section 4.4

The function ϕ is defined in Section 4.4 as follows:

$$\phi(x) = \begin{cases} 0 & \text{if } x \leq 24jd - 11d, \\ (1+\nu)^x & \text{otherwise,} \end{cases}$$

where ν equals $\alpha/(cd^2)$. For the following we set c large enough so that $(1+\nu)^{12d} \leq 3/2$.

Lemma C.5: *For any integer x, if $\phi(x) > 0$, then $\phi(x + 12d) \leq 3\phi(x)/2$.*

Proof: Since $\phi(x) > 0$, we have $\phi(x + 12d) = (1+\nu)^{12d}\phi(x) \leq 3\phi(x)/2$. (Note that if $\phi(x) = 0$ then $\phi(x + 12d)$ may not equal $(1+\nu)^{12d}\phi(x)$.) $\square$

Lemma C.6: *For any integer x we have*

$$\max\{\phi(24jd), \phi(x - 12d)\} \geq 2\phi(x)/3$$

Proof: If $\phi(x - 12d) > 0$, then $2\phi(x)/3 \leq \phi(x - 12d)$ by Lemma C.5. Otherwise, $x - 12d \leq 24jd - 11d$, which implies that $x \leq 24jd + d$. Therefore, $\phi(x) \leq \phi(24jd + d) \leq \phi(24jd)(1+\nu)^d \leq 3\phi(24jd)/2$. $\square$

Lemma C.7: *For any integers x and y, if $\phi(x) > 0$ and $x - y \geq 11d$, then we have $\phi(x) - \phi(y) \geq 2(\phi(x+11d) - \phi(y))/5$.*

Proof:

$$\begin{aligned} 2(\phi(x+11d) - \phi(y))/5 &= 2(\phi(x+11d) - \phi(x))/5 \\ &\quad +2(\phi(x) - \phi(y))/5 \\ &\leq 2(1+\nu)^{11d}(\phi(x) - \phi(x-11d))/5 \\ &\quad +2(\phi(x) - \phi(y))/5 \\ &\leq 2(1+\nu)^{11d}(\phi(x) - \phi(y))/5 \\ &\quad +2(\phi(x) - \phi(y))/5 \\ &\leq \phi(x) - \phi(y). \end{aligned}$$

(In the second equation we use: $x - 11d \geq y$. In the last equation we use: $(1+\nu)^{11d} \leq 3/2$.) $\square$

Appendix D

Proof of Lemma 4.7

This appendix contains a proof of Lemma 4.7 that is stated in Section 4.4. We begin by defining a notion of *goodness* of the tokens. Initially, all tokens are unmarked. After any step t, for every token p that is moved along an edge, p is marked *good* if $h_{t-1}(p) - h_t(p) \geq 6d$; otherwise, p is marked *bad*. The marking of tokens that do not move is unchanged.

Lemma D.1: *For any two bad tokens p_1 and p_2 present at any node v at the start of any step t, if p_1 and p_2 are last sent to v by the same neighbor u of v, then $|h_t(p_1) - h_t(p_2)| > 4d$.*

Proof: Let t_1 (resp., t_2) be the step during which p_1 (resp., p_2) is last sent to v. Without loss of generality, we assume $t_1 < t_2 < t$. Thus we have $h_t(p_1) < h_t(p_2)$. Since u's estimate of the number of tokens at v is updated in step t_1, we have $e^u_{t_1}(v) \geq \rho + h_{t_1}(p_1) - d$. (Note that $e^u_{t_1}(v)$ is u's estimate of the number of tokens at v after step t_1.) Since p_1 remains at v during the interval $[t_1, t_2)$, we find that $e^u_{t'}(v) \geq \rho + h_{t'}(p_1) - d$ for every step t' in $[t_1, t_2)$. In particular, we have $e^u_{t_2-1}(v) \geq \rho + h_{t_2-1}(p_1) - d$. Since u sends p_2 to v in step t_2, $h_{t_2-1}(p_2) \geq h_{t_2-1}(u) - d \geq e^u_{t_2-1}(v) - \rho + 11d \geq h_{t_2-1}(p_1) + 10d$. Since p_2 is bad, we also have $h_{t_2}(p_2) > h_{t_2-1}(p_2) - 6d \geq h_{t_2-1}(p_1) + 4d$. Since $h_t(p_2) = h_{t_2}(p_2)$ and $h_t(p_1) = h_{t_2-1}(p_1)$, the lemma follows. $\square$

Corollary D.1.1: *At any time, for any node u and integer $i > 0$, there are at most d bad tokens with heights in $(i, i + 4d]$.* $\square$

Proof of Lemma 4.7: Consider an arbitrary step t of the algorithm. For every token p transferred from u to v in step t, we assign some credit to every edge adjacent to u or v. Specifically, if p is marked good after step t we assign an *outgoing credit* of $9(\phi(h_{t-1}(p)) - \phi(h_t(p)))/(20d)$ units to every edge adjacent to u and an *incoming credit* of the same amount to every edge adjacent to v. If p is marked bad we assign an outgoing credit of $(\phi(h_t(p) + d) - \phi(h_t(p)))/(20d) + (\phi(h_{t-1}(p)) - \phi(h_{t-1}(p) - d))$ units to every edge adjacent to u and an incoming credit of the same amount to every edge adjacent to v. Also, for each edge (u, v), we assign an *initial credit* of $2\max\{\phi(24jd), (\phi(h_0(u) - d) + \phi(h_0(v) - d))\}$ units at the start of the analysis. The total initial credit I is bounded as follows:

$$
\begin{aligned}
I &\leq 2\binom{n}{2}\phi(24jd) + \sum_{(u,v)\in E} 2(\phi(h_0(u) - d) + \phi(h_0(v) - d)) \\
&\leq n^2\phi(24jd) + \sum_{u\in V}\sum_{0\leq \ell<d} 2\phi(h_0(u) - \ell) \\
&\leq n^2\phi(24jd) + 2\Phi_0.
\end{aligned}
$$

(The first equation follows from the fact that the maximum of two quantities is at most the sum of the particular quantities. We also note that each undirected edge (u, v) appears at most once in the summation. For the second equation, we note that each node has at most d edges. Hence for any node u, the term $2\phi(h_0(u) - d)$ appears in at most d terms of the sum. We complete the derivation of the second equation by observing that $\phi(h_0(u) - \ell)$ is at least $\phi(h_0(u) - d)$ for $0 \leq \ell < d$. The third equation is obtained by the fact that $\sum_{0\leq\ell<d}\phi(h_0(u) - \ell)$ is at most $\phi(u)$.) The above bound on I corresponds to the negative term in Equation (4.4).

We now show that using the above accounting method, we can account for the amortized potential drop of $(\phi(h_{t-1}(u) - d) - \phi(h_{t-1}(v) + d))/50$ units at step t for every edge $(u, v) \in E_t$. To accomplish this, for every live edge (u, v) ((u, v) not necessarily in

E_t), we consider three cases: (i) a token p sent from u to v is marked good, (ii) a token p sent from u to v is marked bad, (iii) no token is sent from u to v.

We first consider case (i). When a token p is marked good after being sent along (u, v), we use the actual potential drop of p to pay for the amortized drop D_1 associated with (u, v) as well as the total credit D_2 assigned to the edges adjacent to u or v due to the transfer of a good token.

$$\begin{aligned} D_1 + D_2 &\leq (\phi(h_{t-1}(u) - d) - \phi(h_{t-1}(v) + d))/50 + 9(\phi(h_{t-1}(p)) - \phi(h_t(p))))/10 \\ &\leq (\phi(h_{t-1}(p)) - \phi(h_t(p)))/50 + 9(\phi(h_{t-1}(p)) - \phi(h_t(p)))/10 \\ &\leq \phi(h_{t-1}(p)) - \phi(h_t(p)). \end{aligned}$$

(The first term in the right-hand side of the first equation is the amortized potential drop. The second term is an upper bound on D_2 since the number of edges adjacent to either u or v is at most $2d$. The second equation follows from the fact that $h_{t-1}(p)$ is at least $h_{t-1}(u) - d$ and $h_t(p)$ is at most $h_t(u) + d$.)

We now consider case (ii). In this case we need to account for: (1) if $h_t(p) > h_{t-1}(p)$, an amount equal to the potential increase of $D_1 = \phi(h_t(p)) - \phi(h_{t-1}(p))$ units, and (2) a credit of at most $(\phi(h_t(p)+d) - \phi(h_t(p)))/10 + (\phi(h_{t-1}(p)) - \phi(h_{t-1}(p) - d))/10$ units. We pay for $(\phi(h_{t-1}(p)) - \phi(h_t(p)))/10$ units of the credit using the potential change. The remainder of the credit we need to account for is at most the sum of $D_2 = (\phi(h_t(p) + d) - \phi(h_t(p)))/10$ and $D_3 = (\phi(h_t(p)) - \phi(h_{t-1}(p) - d))/10$. (Note that this is true regardless of whether the potential of p decreases in step t.)

We have two subcases, depending on whether t is the first step (u, v) is live (subcase (a)) or not (subcase (b)). In subcase (a), if $h_0(u) \geq h_t(p) - d$, the initial credit C_0 associated with (u, v) is at least $2\max\{\phi(24jd), \phi(h_t(p) - 2d)\}$. Since $\phi(h_t(p) - 2d) \geq \phi(h_t(p) - 12d)$, it follows from Lemma C.6 that $3C_0/4 \geq \phi(h_t(p)) \geq D_1$. Since $\phi(h_t(p) - 2d) \geq \phi(h_t(p) - 11d)$, $C_0/4 \geq \phi(h_t(p) + d)/3 \geq \phi(h_t(p) + d)/10 + \phi(h_t(p))/10 \geq D_2 + D_3$. Therefore, we have $C_0 \geq D_1 + D_2 + D_3$.

We now consider subcase (a) under the assumption that $h_0(v) \leq h_t(p) - d$. In order to do the accounting, we use part of the incoming credit associated with the edge

(u, v) due to the set X of good tokens of v with heights in the interval $(h_0(v), h_t(p) - d]$. (Note that each token in X is marked and thus, has contributed incoming credit to all edges adjacent to v.) Since each token x in X is good, the height of the token before the transfer to node v was at least $h_t(q) + 6d$. Therefore, the incoming credit assigned to (u, v) by a token x in X is at least $9(\phi(h_t(q) + 6d) - \phi(h_t(q)))/(20d)$ units. For each token x in X, we use $c_x = 8(\phi(h_t(q) + 6d) - \phi(h_t(q)))/(20d)$ units of this incoming credit. Let C_1 denote $\sum_{x \in X} c_x$. We obtain the following lower bound C_1. By invoking Corollary D.1.1, we obtain:

$$\begin{aligned}
C_1 &\geq \frac{8}{20d} \sum_{1 \leq i \leq \lfloor \frac{h_t(p)-d-h_0(v)}{4d} \rfloor} \sum_{1 \leq k \leq 3d} (\phi(h_t(p) - d - 4id + k + 6d) - \phi(h_t(p) - 4id + k)) \\
&\geq \frac{8}{20d} \sum_{1 \leq k \leq 3d} \sum_{1 \leq i \leq \lfloor \frac{h_t(p)-d-h_0(v)}{4d} \rfloor} (\phi(h_t(p) - d - 4id + k + 6d) - \phi(h_t(p) - 4id + k)) \\
&\geq \frac{8}{20d} \sum_{1 \leq k \leq 3d} (\phi(h_t(p) - d - 4d + k + 6d) - \phi(h_t(p) - 4d \lfloor \frac{h_t(p) - d - h_0(v)}{4d} \rfloor + k)) \\
&\geq \frac{8}{20d} \sum_{1 \leq k \leq 3d} (\phi(h_t(p) + d) - \phi(h_0(v) + 8d)) \\
&= 6(\phi(h_t(p) + d) - \phi(h_0(v) + 8d))/5.
\end{aligned}$$

(In the first equation we partition the interval $(h_0(v), h_t(p) - d]$ into subintervals of $4d$ consecutive integers starting from $h_t(p) - d$. The last subinterval may have fewer than $4d$ integers; if so, we ignore the last subinterval in the sum. The second summation in the first equation is a lower bound on the sum of c_x over each good token x in each subinterval. To obtain the second summation, we invoke Corollary D.1.1 which implies that there are at least $3d$ good tokens in every subinterval of $4d$ tokens. The second equation is obtained by interchanging the order of sums. For the third equation, we use the fact that $\phi(h_t(p) - d - 4(i-1)d + k + 6d) \geq \phi(h_t(p) - 4id + k)$ and then note that the sum telescopes. For the fourth equation, note that: (i) the index k is at least 0 and at most $3d$, and (ii) $h_t(p) - 4d \lfloor \frac{h_t(p)-d-h_0(v)}{4d} \rfloor \leq h_0(v) + 5d$.)

Since p is marked bad after step t, we have $h_t(p) > h_{t-1}(p) - 6d$. Therefore,

$$C_0 + C_1 \geq 2 \max\{\phi(24jd), \phi(h_0(v) - d)\} + 6(\phi(h_t(p) + d) - \phi(h_0(v) + 8d))/5$$

$$
\begin{aligned}
&\geq 6\phi(h_t(p)+d)/5 \\
&\geq \phi(h_t(p)) - \phi(h_{t-1}(p)) + (\phi(h_t(p)+d) - \phi(h_t(p)))/10 \\
&\quad +(\phi(h_t(p)) - \phi(h_{t-1}(p)-d))/10 \\
&\geq D_1 + D_2 + D_3.
\end{aligned}
$$

(The first equation states the lower bounds on C_0 and C_1 obtained above. For the second equation, we invoke Lemma C.6 as follows: $2\max\{\phi(24jd), \phi(h_0(v)-d)\} \geq 4\phi(h_0(v)+11d)/3 \geq 6\phi(h_0(v)+8d)/5$. The third equation is obtained from the following three observations: (i) $\phi(h_t(p)+d) \geq \phi(h_t(p)) - \phi(h_{t-1}(p))$, (ii) $\phi(h_t(p)+d)/10 \geq (\phi(h_t(p)+d) - \phi(h_t(p)))/10$, and (iii) $\phi(h_t(p)+d)/10 \geq (\phi(h_t(p)) - \phi(h_{t-1}(p)-d))/10$.)

We use a similar argument as above to handle subcase (b) where t is not the first step in which (u,v) is live. The set X is the set of good tokens of v with heights in the interval $(e^u_{t-1}(v) - \rho, h_t(p) - d]$. Let c_x and C_1 be defined as in subcase (a). That is, c_x equals $8(\phi(h_t(x)+6d) - \phi(h_t(x)))/(20d)$ units of the incoming credit assigned to (u,v) by a token x in X, and C_1 equals $\sum_{x \in X} c_x$. We will show that $11C_1/12 \geq D_1 + D_3$, and $C_1/12 \geq D_2$, and hence obtain that $C_1 \geq D_1 + D_2 + D_3$.

We first show that $11C_1/12 \geq D_1 + D_3$. If $h_t(p) \leq h_{t-1}(p) - d$, then D_1, and D_3 are both nonpositive and hence the desired claim holds trivially. We now assume that $h_t(p) > h_{t-1}(p) - d$. Let y denote $e^u_{t-1}(v) - \rho + 8d$. We observe that since u sent a token to v during step t, $y = e^u_{t-1}(v) - \rho + 8d \leq h_{t-1}(u) - 4d \leq h_{t-1}(p) - 3d$. Since p is a bad token, we have $y \leq h_{t-1}(p) - 3d < h_t(p) - 2d$. As in subcase (a), we divide the interval $(e^u_{t-1}(v) - \rho, h_t(p) - d]$ into subintervals consisting of $4d$ consecutive integers. Note that $e^u_{t-1}(v) - \rho \leq h_t(p) - 11d$, and hence the number of subintervals is at least 1. We obtain the following lower bound on $11C_1/12$.

$$
\begin{aligned}
11C_1/12 &\geq (11/12)\cdot 6(\phi(h_t(p)+d) - \phi(y))/5 \\
&\geq 11(\phi(h_t(p)+d) - \phi(h_{t-1}(p)-2d))/10 \\
&\geq (\phi(h_t(p)) - \phi(h_{t-1}(p))) + (\phi(h_t(p)) - \phi(h_{t-1}(p)-d))/10 \\
&= D_1 + D_3
\end{aligned}
$$

(The first equation is obtained in the same manner as the upper bound on C_1 in subcase (a). While the interval considered in subcase (a) is $(h_0(v), h_t(p)-d]$, we consider here the interval $(e^u_{t-1}(v)-\rho, h_t(p)-d] = [y-8d, h_t(p)-d]$. Hence, the term $\phi(h_0(v)+8d)$ obtained in the lower bound on C_1 in subcase (a) is replaced by $\phi(y)$ above. The second equation is obtained by the upper bound on y.)

We now show that $C_1/12 \geq D_2$. Since a token is sent by u to v in step t, $e^u_{t-1}(u)-\rho \leq h_{t-1}(u)-12d \leq h_{t-1}(p)-11d$. Moreover, since p is a bad token, $h_{t-1}(p) \leq h_t(p)-6d$. Therefore, $e^u_{t-1}(u)-\rho \leq h_t(p)-5d$. It follows that $(h_t(p)-5d, h_t(p)-d]$ is a subinterval of $(e^u_{t-1}(u)-\rho, h_t(p)-d]$. Hence, C_1 can be lower bounded by adding c_x over all good tokens x whose height is in $(h_t(p)-5d, h_t(p)-d]$. By Corollary D.1.1, at least $3d$ of the tokens in $[h_t(p)-5d, h_t(p)-d]$ are good. We thus obtain:

$$
\begin{aligned}
C_1/12 &\geq (3d/12)\cdot 8(\phi(h_t(p)+d)-\phi(h_t(p)-d))/(20d) \\
&= (\phi(h_t(p)+d)-\phi(h_t(p))/10 \\
&\geq D_2.
\end{aligned}
$$

(For the first equation, note that $c_x = 8(\phi(h_t(x)+6d)-\phi(h_t(x)))/(20d) \geq 8(\phi(h_t(p)+d)-\phi(h_t(p)-d))/(20d)$, for $h_t(x)$ in $[h_t(p)-5d, h_t(p)-d]$. The last equation follows from the definition of D_2.)

To complete the proof for case (ii), we show that for any token x of v, any incoming credit assigned by x to edge (u,v) that is used at step t for case (ii) is not used again for case (ii). To prove this, we note that for any x in X, for every further step $t' > t$ until x is transferred by u, we have $h_{t'}(x) \geq e^u_{t'-1}(v)-\rho$. While establishing case (ii) for step t, we only use the incoming credit assigned by tokens in $(e^u_{t'-1}(v)-\rho, h_{t'}(p)-d]$. Hence the incoming credit assigned by x to edges adjacent to u that is used at step t will never be used again.

We need to consider case (iii) only under the assumption that $(u,v) \in E_t$, i.e., (u,v) is live in step t. In this case we account for $D = (\phi(h_{t-1}(u)-d)-\phi(h_{t-1}(v)+d))/50$ units of potential. Again we consider two subcases depending on whether t is the last step in which (u,v) is live (subcase (a)) or not (subcase (b)). We first consider subcase (a). If $h_0(u) \geq h_{t-1}(u) - 12d$, then we use $C_0 = 2\max\{\phi(24jd), \phi(h_0(u) - d)\}$ units of the initial credit associated with (u, v). Since $h_{t-1}(u) - d \leq h_0(u) - d + 12d$, it follows from Lemma C.6 that $C_0 \geq 4\phi(h_{t-1}(u) - d)/3 \geq \phi(h_{t-1}(u) - d)/50 \geq D$.

We now consider subcase (a) of case (iii) under the assumption that $h_0(u) < h_{t-1}(u) - 12d$. In addition to C_0, we also use part of the incoming credit associated with the set of tokens $Y = \{y : y$ is a token of u and $h_0(u) < h_t(y) \leq h_{t-1}(u)\}$. Specifically, for every token y in Y, we use $(\phi(h_t(y) + d) - \phi(h_t(y)))/(20d)$ units of incoming credit that is assigned to (u, v) by y. Note that since $h_t(y) > h_0(u)$, token y has moved and hence has assigned some incoming credit to (u, v). If y is good, this credit is at least $9(\phi(h_t(y) + 6d) - \phi(h_t(y)))/(20d)$ units; otherwise, this credit is at least $(\phi(h_t(y) + d) - \phi(h_t(y)))/(20d)$. Moreover, if y is a good token, then at most $8(\phi(h_t(y) + 6d) - \phi(h_t(y)))/(20d)$ units of incoming credit were used in the analysis of case (ii). If y is a bad token, none of the incoming credit was used in the analysis of case (ii). In either case, at least $(\phi(h_t(y) + d) - \phi(h_t(y)))/(20d)$ units of incoming credit still remain. Let this credit be denoted C_1. We obtain the following lower bound on $C_0 + C_1$.

$$
\begin{aligned}
C_0 + C_1 &\geq C_0 + \sum_{h_0(u) < k \leq h_{t-1}(u)} (\phi(k + d) - \phi(k))/(20d) \\
&= C_0 + \frac{1}{20d} \sum_{1 \leq i \leq d} (\phi(h_{t-1}(u) + i) - \phi(h_0(u) + i)) \\
&\geq C_0 + (\phi(h_{t-1}(u)) - \phi(h_0(u) + d))/20 \\
&\geq \phi(h_{t-1}(u))/20 \\
&\geq D.
\end{aligned}
$$

(The second equation holds since the sum in the first equation can be expressed as a sum of d telescoping sums. For the third equation we invoke Lemma C.6 and obtain that $C_0 \geq 2\phi(h_0(u) + 11d)/3 \geq \phi(h_0(u) + d)/20$.)

We now consider subcase (b) of (iii). Recall that by the definition of E_t, u is in $S_{>j}$ at the start of step t. Therefore, $h_{t-1}(u) \geq 24(j + 1)d - 12d \geq 24jd + 12d$. Since no token was sent along (u, v) in step t, we have $e^u_{t-1}(v) - \rho > h_{t-1}(u) - 12d$ $(\geq 24jd)$. By the definition of E_t, we also have $h_{t-1}(u) \geq h_{t-1}(v) + 24d$. It follows that $e^u_{t-1}(v) - \rho > h_{t-1}(v) + 12d$. Subsequent to the last step in which (u, v) was live, at least $e^u_{t-1}(v) - \rho - h_{t-1}(v)$ tokens have left v. We use the outgoing credit assigned to (u, v) due to these token transmissions. Consider a token x that is transmitted by v in step t'. If x is marked good after the step, then the outgoing credit assigned by x to (u, v) is at least $9(\phi(h_{t'-1}(p)) - \phi(h_{t'}(p)))/(20d) \geq 9(\phi(h_{t'-1}(p)) - \phi(h_{t'-1}(p) - 6d))/(20d)$ units. Otherwise, the outgoing credit assigned by x to (u, v) is at least $(\phi(h_{t'-1}(p)) - \phi(h_{t'-1}(p) - d))/(20d)$ units. In either case, the outgoing credit is at least $(\phi(h_{t'-1}(p)) - \phi(h_{t'-1}(p) - d))/(20d)$ units. We thus obtain the following lower bound on the total outgoing credit C_2 assigned to (u, v) by the at least $e^u_{t-1}(v) - \rho - h_{t-1}(v)$ tokens.

$$\begin{aligned}
C_2 &\geq \sum_{h_{t-1}(v) < k \leq e^u_{t-1}(v) - \rho} (\phi(k) - \phi(k-d))/(20d) \\
&= \frac{1}{20d} \sum_{1 \leq i \leq d} (\phi(e^u_{t-1}(v) - \rho - d + i) - \phi(h_{t-1}(v) - d + i)) \\
&\geq (\phi(e^u_{t-1}(v) - \rho - d) - \phi(h_{t-1}(v)))/20 \\
&\geq (\phi(e^u_{t-1}(v) - \rho + 11d) - \phi(h_{t-1}(v) + d))/50 \\
&\geq (\phi(h_{t-1}(u) - d) - \phi(h_{t-1}(v) + d))/50 \\
&= D.
\end{aligned}$$

(The second equation holds since the sum in the first equation can be expressed as a sum of d telescoping sums. For the third and fourth equations, we first note that since no token was sent by u to v in step t, we have $e^u_{t-1}(v) - \rho > h_{t-1}(u) - 12d \geq 24jd - d$. The third equation now follows from Lemma C.7 and the equation $\phi(e^u_{t-1}(v) - \rho - d) > 0$. The fourth equation follows directly from the lower bound on $e^u_{t-1}(v) - \rho$.)

We note that the outgoing credit assigned to edge (u, v) in the above analysis of case (iii) is used at most once in case (iii). To prove this, we observe that after step t, the value of $e^u(v)$ is updated by u to $h_{t-1}(v) + \rho$. Therefore, if case (iii) of the analysis subsequently uses any outgoing credit assigned by a token x that leaves v and whose height in v is in $(h_{t-1}(v), e^u_{t-1}(v)]$, then x must have arrived at v after step t. Hence, the outgoing credit assigned by the $e^u_{t-1}(v) - h_{t-1}(v)$ tokens that are used in the analysis for step t is not used again for a later step. $\square$

Bibliography

[1] N. Abramson. The ALOHA system. In N. Abramson and F. Kuo, editors, *Computer-Communication Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[2] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel randomized load balancing. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 238–247, May 1995.

[3] M. Adler, P. B. Gibbons, Y. Matias, and V. Ramachandran. Modeling parallel bandwidth: Local vs. global restrictions. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 94–105, June 1997.

[4] Y. Afek, E. Gafni, and A. Rosen. The slide mechanism with applications to dynamic networks. In *Proceedings of the 11th Annual ACM Symposium on Principles of Distributed Computing*, pages 35–46, August 1992.

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

[6] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.

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

[8] N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley, New York, NY, 1991.

[9] H. Alt, T. Hagerup, K. Mehlhorn, and F. P. Preparata. Deterministic simulation of idealized parallel computers on more realistic ones. *SIAM Journal on Computing*, 16(5):808–835, 1987.

[10] R. J. Anderson and G. L. Miller. Optical communication for pointer based algorithms. Technical Report CRI–88–14, Computer Science Department, University of Southern California, 1988.

[11] T. E. Anderson, M. D. Dahlin, J. N. Neefe, D. A. Patterson, D. S. Rosselli, and R. Y. Wang. Serverless network file systems. In *Proceedings of the 15th Symposium on Operating Systems Principles*, pages 109–126, 1995.

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

[13] E. Arjomandi, M. J. Fischer, and N. A. Lynch. Efficiency of synchronous versus asynchronous distributed systems. *Journal of the ACM*, 30:449–456, 1983.

[14] A. Arora and M. Gouda. Load balancing: An exercise in constrained convergence. In J-M. Hélary and M. Raynal, editors, *Proceedings of the 9th International Workshop on Distributed Algorithms,* Lecture Notes in Computer Science, volume 972, pages 183–197. Springer-Verlag, 1995.

[15] J. Aspnes, M. Herlihy, and N. Shavit. Counting networks. *Journal of the ACM*, 41:1020–1048, 1994.

[16] H. Attiya and M. Mavronicolas. Efficiency of semi-synchronous versus asynchronous networks. *Mathematical Systems Theory*, 27:547–571, 1994.

[17] Y. Aumann, Z. Kedem, K. V. Palem, and M. O. Rabin. Highly efficient asynchronous execution of large-grained parallel programs. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 271–280, November 1993.

[18] B. Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32:804–823, 1985.

[19] B. Awerbuch, Y. Bartal, and A. Fiat. Distributed paging for general networks. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 574–583, January 1996.

[20] B. Awerbuch, L. Cowen, and M. Smith. Efficient asynchronous distributed symmetry breaking. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 214–223, 1994.

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

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

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

[24] B. Awerbuch, Y. Mansour, and N. Shavit. End-to-end communication with polynomial overhead. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 358–363, October 1989.

[25] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 593–602, May 1994.

[26] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51:341–358, 1995.

[27] H. Bast and T. Hagerup. Fast parallel space allocation, estimation and integer sorting. *Information and Computation*, 123:72–110, 1995.

[28] E. Berlekamp and L. Welch. Error correction of algebraic block codes. U.S. Patent Number 4,633,470.

[29] D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[30] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[31] M. A. Blaze. Caching in large-scale distributed file systems. Technical Report TR-397-92, Department of Computer Science, Princeton University, January 1993. PhD Thesis.

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

[33] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. In *Proceedings of the 2nd International World Wide Web Conference*, pages 763–771, October 1994.

[34] A. Broder, A. M. Frieze, E. Shamir, and E. Upfal. Near-perfect token distribution. *Random Structures and Algorithms*, pages 559–572, 5 1994.

[35] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[36] A. Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz, and K. Worrell. A hierarchical Internet object cache. In *Proceedings of the USENIX 1996 Technical Conference*, pages 22–26, January 1996.

[37] H. Chernoff. A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[38] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25:285–287, 1979.

[39] E. Cohen. On the convergence span of greedy load balancing. *Information Processing Letters*, 52:181–182, 1994.

[40] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, pages 119–171. John Wiley and Sons, Inc., New York, New York, 1990.

[41] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 1–12, May 1993.

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

[43] A. Czumaj, Meyer auf der Heide F., and V. Stemann. Shared memory simulations with triple-logarithmic delay. In *Proceedings of the 3rd Annual European Symposium on Algorithms*, pages 46–59, September 1995.

[44] A. Czumaj, F. Meyer auf der Heide, and V. Stemann. Improved optimal shared memory simulations, and the power of reconfiguration. In *Proceedings of the 3rd Israel Symposium on Theory of Computing and Systems*, pages 11–19, 1995.

[45] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, pages 85–111, 1990.

[46] X. Deng, H. N. Liu, L. Long, and B. Xiao. Competitive analysis of network load balancing. *Journal of Parallel and Distributed Computing*, 40:162–172, 1997.

[47] M. Dietzfelbinger and F. Meyer auf der Heide. Simple, efficient shared memory simulations. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 110–119, June 1993.

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

[49] S. Fortune and J. Wyllie. Parallelism in random access machines. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 114–118, May 1978.

[50] J. E. Gehrke, C. G. Plaxton, and R. Rajaraman. Rapid convergence of a local load balancing algorithm for asynchronous rings. In *Proceedings of the 11th International Workshop on Distributed Algorithms*, September 1997. To appear.

[51] M. Geréb-Graus and T. Tsantilas. Efficient optical communication in parallel computers. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 41–48, June 1992.

[52] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 548–558, May 1995.

[53] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two local

load balancing algorithms. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 548–558, May 1995.

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

[55] P. Gibbons, Y. Matias, and V. Ramachandran. The QRQW PRAM: Accounting for contention in parallel algorithms. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 638–648, January 1994. To appear in *SIAM Journal on Computing.*

[56] P. Gibbons, Y. Matias, and V. Ramachandran. The queue-read queue-write asynchronous PRAM model. In *Proceedings of Euro-Par'96,* Lecture Notes in Computer Science, pages 279–292. Springer-Verlag, August 1996. To appear in the special issue of *Theoretical Computer Science* on Parallel Computing.

[57] P. B. Gibbons, Y. Matias, and V. Ramachandran. Efficient low-contention parallel algorithms. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 236–247, June 1994. To appear in the special issue of *Theoretical Computer Science* on Parallel Computing.

[58] P. B. Gibbons, Y. Matias, and V. Ramachandran. Can a shared-memory model serve as a bridging model for parallel computation? In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 72–83, June 1997.

[59] B. Goldberg and P. Hudak. Implementing functional programs on a hypercube multiprocessor. In *Proceedings of the 4th Conference on Hypercubes, Concurrent Computers and Applications*, pages 489–503, 1989.

[60] L. A. Goldberg and M. Jerrum. A sub-logarithmic communication algorithm for the completely connected optical communication parallel computer. Technical Re-

port ECS–LFCS–92–234, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, September 1992.

[61] L. A. Goldberg, M. Jerrum, F. T. Leighton, and S. B. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 300–309, June 1993.

[62] L. A. Goldberg, M. Jerrum, and P. D. Mackenzie. An $\Omega(\sqrt{\log\log n})$ lower bound for routing on optical networks. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 147–156, June 1994.

[63] L. A. Goldberg, Y. Matias, and S. B. Rao. An optical simulation of shared memory. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 257–267, June 1994.

[64] A. G. Greenberg, P. Flajolet, and R. E. Ladner. Estimating the multiplicities of conflicts to speed their resolution in multiple access channels. *Journal of the ACM*, 34:289–325, 1987.

[65] A. G. Greenberg and S. Winograd. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *Journal of the ACM*, 32:589–596, 1985.

[66] J. D. Guyton and M. F. Schwartz. Locating nearby copies of replicated Internet servers. In *Proceedings of ACM SIGCOMM*, pages 288–298, 1995.

[67] J. S. Gwertzman and M. Seltzer. The case for geographical push-caching. In *Proceedings of the 5th Workshop on Hot Topics in Operating Systems*, pages 51–57, May 1995.

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

[69] K. Herley. A note on the token distribution problem. *Information Processing Letters*, 28:329–334, 1991.

[70] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[71] B. Hoppe and É. Tardos. The quickest transshipment problem. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 512–521, January 1996.

[72] J. Jájá and K. W. Ryu. Load balancing and routing on the hypercube and related networks. *Journal of Parallel and Distributed Computing*, 14:431–435, 1992.

[73] M. R. Jerrum and A. Sinclair. Conductance and the rapid mixing property for Markov chains: The approximation of the permanent resolved. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 235–244, May 1988.

[74] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Relieving hot spots on the World Wide Web. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 654–663, May 1997.

[75] R. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 318–326, May 1992.

[76] R. Karp and Y. Zhang. A randomized parallel branch-and-bound procedure. *Journal of the ACM*, 40:765–789, 1993.

[77] R. M. Karp. Parallel combinatorial computing. In J. P. Mesirov, editor, *Very Large Scale Computation in the 21st Century*, pages 221–238. Society for Industrial and Applied Mathematics, 1991.

[78] M. R. Klugerman and C. G. Plaxton. Small-depth counting networks. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 417–428, May 1992.

[79] L. Lamport and N. Lynch. Distributed computing: Models and methods. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 1157–1199. Elsevier/MIT Press, 1990.

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

[81] C. E. Leiserson and B. M. Maggs. Communication-efficient parallel graph algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988.

[82] V. Leppänen. *Studies on the Realization of PRAM*. PhD thesis, Department of Computer Science, University of Turku, November 1996.

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

[84] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing*, pages 771–782, May 1997.

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

[86] R. Lüling and B. Monien. Load balancing for distributed branch and bound algorithms. In *Proceedings of the 6th International Parallel Processing Symposium*, pages 543–549, March 1992.

[87] N. Lynch and M. Fisher. On describing the behavior and implementation of distributed systems. *Theoretical Computer Science*, 13:17–43, 1981.

[88] P. D. MacKenzie, C. G. Plaxton, and R. Rajaraman. On contention resolution protocols and associated probabilistic phenomena. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 153–162, May 1994.

[89] B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for data management in systems if limited bandwidth. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, October 1997. To appear.

[90] Y. Mansour, N. Nisan, and U. Vishkin. Trade-offs between communication throughput and parallel time. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 372–381, May 1994.

[91] K. Mehlhorn and U. Vishkin. Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories. *Acta Informatica*, 21:339–374, 1984.

[92] R. Metcalfe and D. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.

[93] F. Meyer auf der Heide, B. Oesterdiekhoff, and R. Wanka. Strongly adaptive token distribution. *Algorithmica*, 15:413–427, 1996.

[94] F. Meyer auf der Heide, C. Scheideler, and V. Stemann. Exploiting storage redundancy to speed up randomized shared memory simulations. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science, volume 900, pages 267–278. Springer-Verlag, March 1995.

[95] M. Mihail. Conductance and convergence of Markov chains – A combinatorial treatment of expanders. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 526–531, October 1989.

[96] S. J. Mullender and P. M. B. Vitányi. Distributed match-making. *Algorithmica*, 3:367–391, 1988.

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

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

[99] L. L. Peterson and B. S. Davie. *Computer Networks: A Systems Approach*. Chapter 9. Morgan Kaufmann, San Francisco, California, 1996.

[100] A. Pietracaprina, G. Pucci, and J. F. Sibeyn. Constructive deterministic PRAM simulation on a mesh-connected computer. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 248–256, June 1994.

[101] C. G. Plaxton. Load balancing, selection, and sorting on the hypercube. In *Proceedings of the 1st Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 64–73, June 1989.

[102] C. G. Plaxton and R. Rajaraman. Fast fault-tolerant concurrent access to shared objects. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 570–579, October 1996.

[103] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, June 1997.

[104] M. O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36:335–348, 1989.

[105] A. G. Ranade. How to emulate shared memory. *Journal of Computer and System Sciences*, 42:307–326, 1991.

[106] D. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, May 1997.

[107] A. Siegel. On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 20–25, November 1989. Revised version.

[108] V. Stemann. Parallel balanced allocations. In *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 261–269, June 1996.

[109] R. Subramanian and I. D. Scherson. An analysis of diffusive load balancing. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 220–225, June 1994.

[110] M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* PhD thesis, Department of Computer Science, University of California at Berkeley, October 1992.

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

[112] R. M. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4:180–209, 1979.

[113] E. Upfal and A. Wigderson. How to share memory in a distributed system. *Journal of the ACM*, 34:116–127, 1987.

[114] L. Valiant. A combining mechanism for parallel computers. Technical Report TR-24-92, Center for Research in Computing Technology, Harvard University, January 1992.

[115] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[116] L. G. Valiant. General purpose parallel architectures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 943–971. Elsevier/MIT Press, 1990.

[117] M. Van Steen, F. J. Hauck, and A. S. Tanenbaum. A model for worldwide tracking of distributed objects. In *Proceedings of TINA '96*, pages 203–212, September 1996.

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

[119] C.-Z. Xu and F. C. M. Lau. Iterative dynamic load balancing in multicomputers. *Journal of the Operational Research Society*, 45:786–796, 1994.

Vita

Rajmohan Rajaraman was born on January 4, 1971 in Calcutta, India, the son of Lalita Rajaraman and Neelagudi Subramaniam Rajaraman. He did his schooling in Sawai Madhopur and New Delhi, India. He received the Bachelor of Technology degree in Computer Science and Engineering from the Indian Institute of Technology at Kanpur in July 1991. Thereafter, he entered graduate school at the University of Texas at Austin in August 1991. He received the Master of Sciences degree in Computer Sciences in May 1993. He was employed as a consultant at Sandia National Laboratories, Albuquerque, in the summer of 1996. He was awarded the MCD and Schlumberger graduate fellowships in August 1991 and November 1996, respectively.

Permanent Address: c/o Mrs. Lalita Rajaraman
28, Janak Road
Calcutta 700-029, India

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1]The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin.