Fast Fault-Tolerant Concurrent Access to Shared Objects*

C. Greg Plaxton *Rajmohan Rajaraman*

Abstract

We consider a synchronous model of distributed computation in which n nodes communicate via point-to-point messages, subject to the following constraints: (i) in a single "step", a node can only send or receive $O(\log n)$ words, and (ii) communication is unreliable in that a constant fraction of all messages may be lost at each step due to node and/or link failures. We design and analyze a simple local protocol for providing fast concurrent access to shared objects in this faulty network environment. In our protocol, clients use a hashing-based method to access shared objects. 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. Our protocol has practical potential for supporting high levels of concurrency in distributed file systems over wide area networks.

1 Introduction

A basic problem in distributed memory environments is to provide efficient access to remote objects (e.g., files, words of memory). This is a complicated problem because of the large number of (often competing) considerations involved, including: object size, network topology, latency, throughput, buffer space, degree of concurrency to be supported, consistency requirements, and communication overhead.

This paper describes a hashing-based protocol for supporting fast fault-tolerant concurrent access to shared objects in a distributed network. The protocol is 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 (i.e., the number of users trying to access the object) does not constantly change by a significant amount. 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 fluctuate arbitrarily.

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 [15], and multicast [8]. 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.

---

*Department of Computer Science, University of Texas at Austin, Austin, TX 78712. This research was supported by the National Science Foundation under Grant No. CCR–9504145, and the Texas Advanced Research Program under Grant No. 93–003658–461. E-mail: {plaxton,rraj}@cs.utexas.edu.

Modern distributed networks tend to have complex topologies that can support many independent point-to-point connections simultaneously. In fact, it has been widely observed that the communication capabilities of modern distributed networks are similar to those previously associated only with tightly-coupled parallel machines. For this reason, we have chosen to implement and evaluate the performance of our protocol under a model of computation that may be loosely viewed as a variant of Valiant's bulk-synchronous parallel (BSP) model [18]. As in the BSP model, we assume the existence of a simple point-to-point router with no built-in combining or multicast capability. (See [18] for a detailed justification of this assumption.) In order to demonstrate the fault-tolerance of our protocol, our model of computation incorporates 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. See Section 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" 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 fact, fault-tolerance considerations alone 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 [14] (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. [3] as part of an efficient scheme for emulating large-grained PRAM programs on an asynchronous parallel machine. In this paper, 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 a small number (e.g., $O(\log n)$) of messages in a single time step, it is clear that further replication (either of the individual fragments, or of the object as a whole) is needed in order to rapidly service all $\sqrt{n}$ requests. In our scheme, we choose to replicate whole copies of popular 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 [14].)

At a high level, our protocol achieves 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 increase dramatically during subsequent steps, each of the outstanding requests is serviced in $O(1)$ expected steps, and in $O(\log n)$ steps whp[1].)

The methods used to establish these invariants, discussed in Section 3, are loosely related to Valiant's hashing-based combining mechanism for simulating CRCW PRAM algorithms on parallel computers [17]. In related work, Gibbons et al. [9] 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 Gibbons et al. and Valiant 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. See Section 4 for a formal statement of our main results.

We remark that existing implementations for handling concurrent access to shared objects (e.g., [2, 5, 10]) do not provide fast concurrent access in the sense considered in this paper. While these schemes incorporate replication of objects, the only way for a client to determine where the copies of a given object are stored is to consult the "manager" of the object. The manager is usually implemented as a process running at a single node and thus constitutes a sequential bottleneck.

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. In a BSP-like model [18], where communication is assumed to take the form of an h-relation, we might now tend to add a requirement that the total number of words in all messages destined to a single node in one step must be $O(\log n)$. We make a slight variant of this assumption which will ultimately allow us to obtain a more efficient protocol. Namely, 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. As in the c-arbitrary crossbar model of MacKenzie et al. [13] (where all messages have unit length), 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. (The related c-arbitrary DMM model of Karp et al. [12] does not take into account contention among clients trying to access the same object and hence is not well-suited for our study.)

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

[1]We use the abbreviation "whp" throughout the paper to mean "with high probability" or, more precisely, "with probability $1 - n^{-c}$, where n is the number of nodes in the network and c is a constant that can be set arbitrarily large by appropriately adjusting other constants defined within the relevant context."

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 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 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 5. See Section 7 for a discussion of write operations. As mentioned in Section 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 approximately 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. 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. If the popularity of A is low (i.e., $O(\log n)$ where the hidden constant is sufficiently small), then whp 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.)

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 whp 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 whp 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. 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, whp 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 whp. 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 $i+1$. As a result, at time $t+1$, a constant fraction of the servers in block $i+1$ of A hold a copy of A whp.

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. (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.

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. Because 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 whp: (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 [6], 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.

4 Results

In this section, we state our main performance bounds for the read-only protocol, which is formally defined in Section 5. We analyze the protocol under different access pattern models. Our time bounds are stated in terms of *rounds*; for any nonnegative integer t, round t consists of steps $2t$ and $2t+1$ (steps are numbered from 0). 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 *steady* if: (i) there exists a real constant δ, $0 \le \delta < 1$, such that $r_{i+1}(t) \le \delta s_i(t)$ 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 steady, 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 1 *In the fixed model, there exists $t_0 = O(\log n)$ such that for any t, $t_0 \le t \le \mathrm{poly}(n)$, round t is steady whp.*

It follows from Theorem 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 1.1 *In t rounds of the fixed model, where $\Omega(\log n) \leq t \leq O(\text{poly}(n))$, whp, 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.* ■

(Note that in Corollary 1.1 the per-request communication is optimal since each object is of size $\Theta(\log n)$.)

The bounds stated in Corollary 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 $t > 0$, $q_i(t) \leq c \max\{q_i(t') : \max\{0, t - \Delta\} \leq t' < t\}$, where $c > 1$ 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 2 *In the dynamic model, for an appropriately chosen ρ it holds wvhp that each round is steady, 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 c in the above theorem depends on the rate of growth of b_i with i. As defined in Section 5, b_i grows as $2^i \log n$. We can establish the claim of Theorem 2 for arbitrarily larger values of c by choosing an appropriately large growth rate for b_i.

5 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 be the size of the ith block, i.e., b_i equals $c_1 2^i \log n$.

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.) 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 all object copies that may be received by the node in Δ rounds. Thus, the minimum cache capacity is $\Theta(\Delta)$ objects. We assume that Δ 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.

In our description of the protocol, we differentiate between several kinds of messages; these are listed in Table 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*.

The protocol is defined in Figure 1, 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. (Recall that round t consists of steps $2t$ and $2t+1$.) 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 1, 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 1, π_0, π_1, π_2, π_3, π_4, π_5, and π_6 denote positive integer constants.

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 1: Types of messages.

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

6 Analysis

Our analysis proceeds in two parts. In the first part, Section 6.1, we define the notion of a good round and show that Invariants 1 and 2 of Section 1 hold whp in a good round. The claims in Section 6.1 hold for an arbitrary access pattern. In the second part, we restrict our attention to the fixed and dynamic models. In Section 6.2 we analyze the fixed model and prove Therorem 1. In Section 6.3, we analyze the dynamic model and establish Theorem 2.

To simplify the presentation, we assume in this section that there is no contention among messages of distinct types. The message priorities defined in Section 5 can easily be used to remove this assumption.

6.1 Protocol Properties

Definition 6.1 *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,ε,τ) **experiment***, we are given a set $\{V_i : i \in X, V_i \subseteq U,$ and $|V_i| \le \varepsilon|U|\}$, 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 6.2 *Let X denote a set of balls, and let U, ε, and τ be as defined in Definition 6.1. In a* **random** (X,U,ε,τ) **experiment***, we are given a set $\{V_i : i \in X, V_i \subseteq U,$ and $|V_i| \le \varepsilon|U|\}$, 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.*

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 $0 \leq i < b$, 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 $D(S,t) \geq \pi_1 b_0$ then attempt to send a *server-frag* message to each server in $B_1(A)$.
- Server, $i > 0$. If $|D(S,t)| \geq \pi_2$ then attempt to send a *client-obj* message to $\min\{\pi_3, |D(S,t)|\}$ random clients in $D(S,t)$, and if $D(S,t) \geq \pi_4$ 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 1: The read-only protocol (object A, client C, server S, round t).

For convenience, we refer to any message of type α as an *α-message*. 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 instances of Definitions 6.1 and 6.2.

Definition 6.3 *Round t is* **good** *if there exists a sufficiently small real ε such that, for every object A, the following conditions hold:*

1. *For 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 6.1 *For any i in $[m]$, any round t, and any α, $\|T_\alpha(t)\|$ is $O(n \log n)$ whp.*

Proof: The messages attempted in step $2t$ are described in Phase 0 of Figure 1. (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 1. (No messages are attempted in Phase 3.) We consider different cases based on α. Let x_i equal $s_i(t)$.

- Case α = *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 $O(n \log n)$ whp. 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 $x_i \le \log n$, 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 $b_j \le x_i$. 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 attemped 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 C.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 $O(x_i)$ whp. Thus, the total number of attempted *client-obj* messages associated with objects in H is $O(n)$ whp.

  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 C.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:

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

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

■

Lemma 6.2 *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 whp.*

Proof: We fix indices $i \in [b]$ and $j \in [m]$. We prove the desired claim by establishing the four parts of Definition 6.3. 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 6.1, $\|T_\alpha(t)\|$ is $O(n \log n)$ whp. 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 [7], 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 sufficently large, we obtain that $|N'_\alpha(A_j, i, t)|$ is $\Omega(|N_\alpha(A_j, i, t)|)$ whp.

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 whp, 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 6.1 is established by substituting $(G(A_j, t), B_0(A_j), (1-c), \Theta(\log n), B_0(A_j) \setminus W_v)$ for $(X, U, \varepsilon, \tau, V_i)$.

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 $(M_\alpha(A_j, i, t), B_i(A_j), \varepsilon, \Theta(\log n)/\,\mathrm{size}(\alpha))$ 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 whp, 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 6.2 is established by substituting $(M_\alpha(A_j, i, t), B_i(A_j), (1-c), \Theta(\log n)/\operatorname{size}(\alpha), B_i(A_j) \setminus W_y)$ for $(X, U, \varepsilon, \tau, V_i)$.

■

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 $t+1 \leq t' \leq t+\Delta$, $s(A,1,t')$ 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 $t+1 \leq t' \leq t+\Delta$, $s(A,i,t')$ 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 1. We assume that $\pi_3 \geq \max\{48\pi_2, 4\pi_0\}$.

Lemma 6.3 *If round t is good, then the following predicates hold for every object A whp: (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 6.3 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 B.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 1), i.e., $N_\beta(A,0,t) = X$. Since step $2t+1$ is good, it follows from part (i) of Definition 6.3 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 6.3, $M'_\beta(A,1,t)$ 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 B.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 6.3 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 C.3.1, $|X|$ 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 6.3, 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 6.3 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 C.3.1, the number of servers in $B_{i+1}(A)$ that receive at least $\pi_5/20$ β-messages is $9b_{i+1}/10$ whp. Thus, $s(A,i+1,t+1)$ is complete whp, 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. 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 6.3 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 6.3, 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 6.3 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 C.3.1 and C.4.1, 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 whp. Hence, $|N_\beta(A,i,t+1)|$ is $4b_i/5$ whp. Since round t is good, by part (i) of Definition 6.3, 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 whp, 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 C.3.1 and Lemma C.4, 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 6.3, 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.

■

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 for every server S in $\cup_{k\ge j} B_k(A_i)$, $a(S,t)$ is 0. In the following lemma, we use Lemma 6.3 to relate $d_i(t)$, $e_i(t)$, $r_i(t)$, and $s_i(t)$ when t is good.

Lemma 6.4 *Let t be a good round. For any i in $[m]$:*

1. *If $s_i(t)$ is at most $\pi_0 b_0$, then $r_i(t+1) = 0$ whp.*

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 $s_i(t)/20$ whp.

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 6.3 to prove the claims.

1. The claim directly follows from Part (iii) of Lemma 6.3.
2. There exists an integer $j \le d_i(t)$ such that $\pi_3 b_j/12 \le s_i(t) \le \pi_3 b_j/6$. Therefore, by Part (iv) of Lemma 6.3, $s_i(t) - r_i(t+1)$ is at least $s_i(k)/20$ whp.
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 6.3 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 $\pi_3 b_{e_i(t)}$.

■

6.2 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.

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

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 appropriately chosen. (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 $\mathcal{A}_{\ge i}$ denote the set $\cup_{j \ge i} \mathcal{A}_j$. We define $\mathcal{A}_{\le i}$, $\mathcal{A}_{>i}$, and $\mathcal{A}_{<i}$ similarly. For each i in $[m]$, object A_i is said to be *steady* in round t if $s_i(t) \le a_2 b_{d_i(t)}$; otherwise, A_i is said to be *unsteady* in round t. 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 $p \ge a_0 \log n/n$, $a_4 np \le B(n, p) \le a_3 np$ whp; a_3 and a_4 may obtained from standard Chernoff bounds [6] given in Theorem A.1.

Lemma 6.5 *Let rounds* 0 *through* $r-1$ *be good. If object* A_i *is not steady in rounds* 0 *through* r*, then whp we have* $d_i(j) = j$ *and* $e_i(j) = j+1$ *for* $0 \le j < r$*.*

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 unsteady 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 6.4, $d_i(k)$ is k whp. Since $e_i(k-1)+1 \le e_i(k) < d_i(k)$ and $e_i(k-1) = k$, $e_i(k)$ is $k+1$ whp. ∎

Lemma 6.6 *Let rounds* 0 *through* $r-1$ *be good. If* A_i *is not in* $\mathcal{A}_0$ *and* A_i *is unsteady in rounds* 0 *through* r*, then* $s_i(r) \ge a_2 a_4 n p_i / (3\pi_3)$ *whp.*

Proof: By Lemma 6.5, $d_i(j)$ is j for all j in $[r+1]$. By Part 4 of Lemma 6.4, it follows that $s_i(j) - r_i(j+1)$ is at most $2\pi_3 b_j$. Thus, we have whp:

$$\begin{aligned}
s_i(r) &= s_i(0) + \sum_{0 \le j \le r} q_i(j) - \sum_{0 \le j < r} (s_i(j) - r_i(j+1)) \\
&\ge s_i(0) - \sum_{0 \le j < r} 2\pi_3 b_j \\
&\ge s_i(0) - 2\pi_3 b_r \\
&\ge a_4 n p_i - 2\pi_3 b_r \\
&\ge a_4 n p_i - 2\pi_3 s_i(r)/a_2 \\
&\ge a_4 n p_i / (1 + 2\pi_3/a_2) \\
&\ge a_2 a_4 n p_i / (3\pi_3).
\end{aligned}$$

(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 unsteady in round r. The last equation holds since $a_2 \le \pi_3$.) ∎

Lemma 6.7 places an upper bound on the number of requests for object A_i during any round after the first round in which A_i is steady.

Lemma 6.7 *Let rounds* 0 *through* $r-1$ *be good, where* r *is at most* Δ*. Assume that* A_i *does not belong to* $\mathcal{A}_0$*, and let* $j < r$ *be the smallest integer such that* A_i *is steady in round* j*. There exist constants* $a_5 > 1$ *and* a_6 *such that, for* $j \le k < r$*, we have whp:*

$$s_i(k) \le \max\left\{ \frac{s_i(j)}{a_5^{k-j}}, a_6 n p_i \right\}. \tag{1}$$

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

$$s_i(k) > (a_6 - a_3) n p_i. \tag{2}$$

We consider two cases depending on whether $s_i(j) \ge a_6 n p_i$.

- Case $s_i(j) \ge a_6 n p_i$: We show by induction on k that Equation 1 holds whp. The induction basis is trivially true. Let Equation 1 be true for rounds j through k, where $j \le k < r-1$. Since j is the first round in which A_i is steady, Lemma 6.5 implies that $d_i(j)$ is j. Therefore, $s_i(j) \le a_2 b_j$ whp.

By the induction hypothesis, $s_i(k) \le s_i(j)$. Moreover, since r is at most Δ, $d_i(k)$ is at least j. Therefore, $s_i(k) \le a_2 b_j \le a_2 b_{d_i(k)}$. Since a_2 is at most $\pi_3/6$, Part 2 of Lemma 6.4 implies whp:

$$s_i(k) - r_i(k+1) \ge s_i(k)/20. \tag{3}$$

Therefore we have whp:

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

(The second equation follows from the definition of s_i. The third equation follows from Equations 2 and 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 $s_i(k) \le a_6 n p_i$ whp. The proof is by induction on k. The induction basis is trivial. Let the claim be true for rounds j through k where $j \le k < r$. In the induction step, we need to show that $s_i(k+1) \le a_6 n p_i$.

  Let ℓ be the last round in which $s_i(\ell) \le a_2 b_{d_i(\ell)}$. (Such an ℓ exists since $s_i(j) \le a_2 b_{d_i(j)}$.) By Part 3 of Lemma 6.4 and the definition of ℓ, $d_i(k) \ge d_i(\ell) + (k - \ell)$ whp. By a Chernoff bound, $s_i(\ell)$ is at least $(a_6 - (k-\ell)a_3)np_i$ whp. Therefore, $b_{d_i(\ell)}$ is at least $(a_6 - (k-\ell)a_3)np_i/a_2$ whp. Moreover, since ℓ is at most Δ, $d_i(\ell) \ge j$. Therefore, we have whp:

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

  (The second equation holds since A_i is steady in round j. The third equation follows from Lemma 6.6.) Therefore, we have whp:

$$b_{d_i(\ell)} \ge \max\left\{(a_6 - (k-\ell)a_3), \frac{2a_2^2 a_4}{9\pi_3^2}\right\} \frac{np_i}{a_2}. \tag{4}$$

  We select a_6 and a new constant a_7 such that $2a_3a_7 \le a_6 \le 2^{a_7+2} a_2 a_4/(9\pi_3^2)$. If $k - \ell$ is at most a_7, then $b_{d_i(k)}$ is at least $a_6 n p_i/2a_2$. By the induction hypothesis, $s_i(k)$ is at most $a_6 n p_i$. Therefore, $s_i(k) \le 2a_2 b_{d_i(k)}$. If $k - \ell$ is at least a_7, then $b_{d_i(k)}$ is at least $2^{a_7+1} a_2 a_4 n p_i/(9\pi_3^2)$. By the choice of a_6 and a_7, we obtain that $s_i(k) \le 2a_2 b_{d_i(k)}$ whp.

  By Part 2 of Lemma 6.4, we have whp: $s_i(k) - r_i(k+1)$ is at least $s_i(k)/20$. Therefore, $s_i(k+1)$ is at most $19 s_i(k)/20 + a_3 n p_i$, which is at most $a_6 n p_i$ by the choice of the constants.

■

We use Lemma 6.8 to relate the number of requests, in round r, for any two objects that are unsteady in rounds 0 through $r - 1$.

Lemma 6.8 *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 not steady in rounds* 0 *through* r*. Then, whp we have:*

$$s_{i_1}(r) \leq \frac{3a_3\pi_3 p_{i_1}}{a_2a_4p_{i_2}} s_{i_2}(r).$$

Proof: Consider any round j, $0 \leq j < r$. We are given that A_{i_2} does not become steady in any of the r rounds. We invoke Lemma 6.5 and obtain that $d_{i_2}(j) = j$ whp. Therefore, $s_{i_2}(j) > a_2 b_j$ whp. By Part 4 of Lemma 6.4, $(s_{i_2}(j) - r_{i_2}(j+1))$ is at most $2\pi_3 b_j$ whp. 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 whp: $\sum_{0\leq j\leq r} q_{i_1}(j)$ is at least $a_4 p_{i_1} q$ and $\sum_{0\leq j\leq r} q_{i_2}(j)$ is at most $a_3 p_{i_2} q$. We thus have:

$$\begin{aligned} s_{i_1}(r) &\leq a_3 p_{i_1} q, &(5)\\ s_{i_2}(j) &\geq a_4 p_{i_2} q - \sum_{0\leq j<r} 2\pi_3 b_j. &(6)\end{aligned}$$

From Equations 5 and 6, we obtain whp:

$$\begin{aligned} s_{i_1}(r) &\leq a_3 p_{i_1} q\\ &\leq \frac{a_3 p_{i_1}}{a_4 p_{i_2}}(s_{i_2}(r) + 2\pi_3 b_r)\\ &\leq \frac{3a_3\pi_3 p_{i_1}}{a_2a_4p_{i_2}} s_{i_2}(r).\end{aligned}$$

(The first equation follows from Equation 5. The second equation follows from Equation 6. The last equation holds since A_{i_2} is unsteady in round r and $a_2 \leq \pi_3$.) ■

Lemma 6.9 *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 steady 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 steady 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 steady 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 not steady in rounds 0 through $j-1$, $d_{i_1}(j) = j$ whp by Lemma 6.5. Since A_{i_1} is steady in round j, we have $s_{i_1}(j) \leq a_2 b_j$ whp, and we obtain an upper bound on $s_{i_1}(j-k+1)$ as follows:

$$\begin{aligned} s_{i_1}(j-k+1) &= s_{i_1}(j) + \left(\sum_{0\leq \ell<k}(s_{i_1}(j-\ell) - r(j-\ell+1)) - q(j-\ell+1)\right)\\ &\leq s_{i_1}(j) + \left(\sum_{0\leq \ell<k}(s_{i_1}(j-\ell) - r(j-\ell+1))\right)\\ &\leq s_{i_1}(j) + 2\pi_3 b_j\\ &\leq 3\pi_3 b_j. &(7)\end{aligned}$$

(The first equality follows from the definition of s_{i_1}. The third equation follows from Part 1 of Lemma 6.4. The last equation holds since A_{i_1} is steady in round j and $a_2 \leq \pi_3$.)

If A_{i_2} is steady in some round $j' < j-k+1$, then the claim holds. Otherwise, by Lemma 6.8, it holds whp that $s_{i_2}(j-k+1) \leq \frac{3a_3\pi_3}{a_2a_4a_1^k} s_{i_1}(j-k+1)$. Hence, A_{i_2} is steady in round $j-k+1$ whp 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 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 steady in some round $j' < j + a_8$, then the claim holds. Otherwise, A_{i_2} is steady in round $j + a_8$ whp because:

$$\begin{aligned}
s_{i_2}(j + a_8) &\leq s_{i_2}(j-1) + a_4a_8np_{i_2} \\
&\leq s_{i_2}(j-1)(1 + 3\pi_3a_8/a_2) \\
&\leq \frac{3(1+3\pi_3a_8/a_2)a_1a_3\pi_3}{a_2a_4}s_{i_1}(j-1) \\
&\leq \frac{3(1+3\pi_3a_8/a_2)a_1a_3\pi_3}{a_2a_4}(s_{i_1}(j) + \pi_3b_j) \\
&\leq \frac{3(1+3\pi_3a_8/a_2)a_1a_3\pi_3(a_2+\pi_3)}{a_2a_4}b_j \\
&\leq a_2b_{j+a_8}.
\end{aligned}$$

(The first equation follows from the definition of s_{i_2}. The second equation follows from Lemma 6.6. The third equation follows from Lemma 6.8. The fourth equation follows from the definition of s_{i_1}. The fifth equation holds since A_{i_1} is steady in round j. The last equation follows from the choice of a_8.) ■

Lemma 6.10 *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 π_0b_0 whp. Since round j is good, by Part 1 of Lemma 6.4, it follows that $r_i(j+1)$ equals zero whp. ■

Definition 6.4 *For nonnegative integers i and j, $0 \leq i < m$, we define:*

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

Lemma 6.11 *For all $j > 0$, $\sum_{0 \leq i < m} s_i^*(j)$ is $O(n)$ whp.*

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

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

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_4n whp. The second sum is bounded as follows:

$$\sum_{i:A_i \in \mathcal{A}_{\leq j} \cap \mathcal{A}_{>0}} s_i^*(j) = \left(\sum_{0 < \ell \leq j} \sum_{i:A_i \in A_\ell} \frac{np_i}{a_5^{j-\ell}\sum_{A_k \in \mathcal{A}_{\geq \ell}} p_k}\right) + \sum_{i:A_i \in \mathcal{A}_{\leq j}} np_i$$

$$
\begin{aligned}
&\leq \left(\sum_{0<\ell\leq j}\sum_{i:A_i\in\mathcal{A}_\ell}\frac{np_i}{a_5^{j-\ell}\sum_{A_k\in\mathcal{A}_\ell}p_k}\right)+\sum_{i:A_i\in\mathcal{A}_{\leq j}}np_i \\
&= \sum_{0<\ell\leq j}\frac{n}{a_5^{j-\ell}}+\sum_{i:A_i\in\mathcal{A}_{\leq j}}np_i \\
&\leq \frac{n}{1-a_5}+\sum_{i:A_i\in\mathcal{A}_{\leq j}}np_i. \qquad (8)
\end{aligned}
$$

Similarly, we bound the third sum as follows:

$$
\begin{aligned}
\sum_{i:A_i\in\mathcal{A}_{>j}}s_i^*(j) &= \left(\sum_{i:A_i\in A_{\geq j}}\frac{np_i}{\sum_{A_k\in\mathcal{A}_{\geq j}}p_k}\right)+\sum_{i:A_i\in\mathcal{A}_{>j}}np_i \\
&= n+\sum_{i:A_i\in\mathcal{A}_{>j}}np_i. \qquad (9)
\end{aligned}
$$

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

Lemma 6.12 *Let rounds* 0 *through* $r-1$ *be good, where* r *is at most* Δ*. Whp, round* r *is good.*

Proof: We show that there exists an integer h such that for all i in $[m]$, $s_i(r)$ is $O(s_i^*(h))$ whp. 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 steady 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 6.10, 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 6.9 and the definition of $\mathcal{A}_i$, it follows whp that for i in $[h]$, every object in $\mathcal{A}_i$ is steady 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 steady. By Lemma 6.9 and the definition of $\mathcal{A}_i$, it holds whp that every object $A_k\in\mathcal{A}_{\geq i}$ is unsteady in some round $r'=r_j-O(1)$. By Lemma 6.8, it holds that for $A_k\in\mathcal{A}_{\geq i}$, we have $s_k(r')=\Omega(p_k s_j(r')/p_j)$ whp. Therefore we have whp:

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

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 6.8. The second equation follows from the earlier claim that $r_j\leq r-h+i+1$. 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 6.9, h' is at least $h-1$ whp. By Lemma 6.9, it holds whp that every object in $\mathcal{A}_{\geq h'}$ is unsteady in some round $r' = r - O(1)$. Consider any object $A_j \in U \cap A_i$, $i \geq h'$. By Lemma 6.8, for all objects $A_k \in \mathcal{A}_{\geq h'}$, it holds whp that $s_k(r') = \Omega(p_k s_j(r')/p_j)$. Therefore we have whp:

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

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

We have shown that for each j in $[m]$, $s_j(r)$ is $O(s_j^*(h))$ whp. By Lemmas 6.2 and 6.11, it follows that round r is good whp. ■

Let t_0 equal $b + \log_{a_5} n$. We assume that Δ is at least $t_0 + a_9$, where a_9 is a constant that is specified in the proof of Lemma 6.13 below.

Lemma 6.13 *For any i in $[m]$ and for any $t \geq t_0$, we have whp:*

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 6.12, rounds 0 through t_0 are good whp. For any i in $[m]$, if A_i is not steady in rounds 0 through $b-2$, then by Lemma 6.5, $d_i(b-1)$ is $b-1$. Since $a_2 c_1 2^{b-1} \log n > n \geq s_i(b-1)$, A_i is steady in round $b-1$. We have thus shown that for each i in $[m]$, A_i is steady 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 6.7, $s_i(t)$ is at most $a_6 np_i \leq a_{10} np_i$ whp, thus establishing claim (i) (we set $a_{10} \geq a_6$). Claim (ii) follows from claim (i) and Lemma 6.2. Claim (iii) holds vacuously. Since Δ is $\Omega(\log n)$, Lemma 6.6 implies that $b_{d_i(t_0)}$ is at least $a_2 a_4 np_i/(9\pi_3^2) \geq a_{11} np_i$, thus establishing claim (iv) (we set $a_{11} \leq a_2 a_4/(9\pi_3^2)$). 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 $s_i(t+1) \leq s_i(t) + q_i(t+1) \leq a_{10} np_i$ whp, 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 $s_i(\ell) \leq 9 a_6 \pi_3^2 b_{d_i(\ell)}/(a_2 a_4)$. (Such an ℓ exists as t_0 satisfies the inequality.) Since $9 a_6 \pi_3^2/(a_2 a_4) \geq 4\pi_4$, Part 3 of Lemma 6.4 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 $(a_{10} - (t-\ell) a_3) np_i$ whp. Therefore, $b_{d_i(\ell)}$ is at least $2 a_2 a_4 (a_{10} - (t-\ell) a_3) np_i/(9 a_6 \pi_3^2)$ whp. Moreover, by the induction hypothesis, $b_{d_i(\ell)}$ is at least $a_{11} np_i$. Thus,

$$b_{d_i(\ell)} \geq \max\{\frac{2 a_2 a_4 (a_{10} - (t-\ell) a_3)}{9 a_6 \pi_3^2}, a_{11}\} np_i. \tag{10}$$

We choose a_{13} and a_{10} such that $2 a_3 a_{13} \leq a_{10} \leq a_2 a_{11} 2^{a_{13}}$. If $t - \ell$ is at most a_{13}, then $b_{d_i(t)}$ is at least $a_2 a_3 a_4 a_{10} np_i/(9\pi_3^2)$. By the induction hypothesis, $s_i(t)$ is at most $a_{10} np_i$. Therefore, $s_i(t)$ is at most $18 \pi_3^2 b_{d_i(t)}/(a_2 a_4)$. By Part (iii) of Lemma 6.3, we have whp: $s_i(t) - r_i(t+1)$ is at least $a_2 a_4 s_i(t)/(2160 a_6 \pi_3)$

whp. If $t-\ell$ is at least a_{13}, then $b_{d_i(t)}$ is at least $2^{a_{13}}a_{11}np_i$ whp. By the choice of constants, we obtain that $s_i(t)$ is at most $a_2 b_{d_i(t)}$ whp. By Part 2 of Lemma 6.4, $s_i(t)-r_i(t+1)$ 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 6.2. 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 6.3, $s_i(t)-r_i(t+1)$ is at least $a_{11}\pi_3 s_i(t)/(120a_{10})$ whp. 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 $a_{12}np_i$ whp 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 6.3 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 6.3), 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 $t \le t_0 + a_9$, 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 $t-a_9 \le t' \le t$, $s_i(t')$ is at least $a_{12}np_i$. By Lemma C.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 $a_{11}np_i$. ■

Proof of Theorem 1: The desired claim follows directly from Lemmas 6.10 and 6.13, and Part (iv) of Lemma 6.3. ■

6.3 The Dynamic Model

The proof for the dynamic model follows easily from the properties of the protocol derived in Sections 6. 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 $t > 0$, $q_i(t) \le (21/20)\max\{q_i(t') : \max\{0, t-\Delta\} \le t' < t\}$.

Lemma 6.14 *For all i in $[m]$ and $0 \le t \le \mathrm{poly}(n)$, we have wvhp:*

1. *$s_i(t)$ is at most $\sum_{0\le j\le t}(19/20)^{(t-j)}q_i(j)$,*
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 $\sum_{0\le j\le t}(19/20)^{t-j}q_i(j)$. For all i and t', $q_i(t')$ is chosen independent of the hash functions. Moreover, $\sum_{0\le i<m} q_i(t')$ is at most n for all t'. Therefore, $\sum_{0\le i<m} x_i$ is $O(n)$. It thus follows from Lemma 6.2 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 6.4 that $r_i(t+1)$ is at most $19s_i(t)/20$. We thus have:

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

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 $[\max\{0, t-\Delta\}, t]$, $s_i(t')$ 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 6.4, $d_i(t+1)$ is at least $j+1$. Therefore, $s_i(t') < 4\pi_1 b_{d_i(t+1)}$.

After round t, we have:

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

thus completing the proof of the induction step. ■

Proof of Theorem 2: It follows from Part (iv) of Lemma 6.3 and Parts 2 and 3 of Lemma 6.14 that each round is steady. Since each round is steady, 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. ■

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 4 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 whp. 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. On the positive side, it is possible to prove a good amortized bound on the total number of messages used by the extended protocol for processing a given number of read/write accesses.

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 [4] (see also [16, 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 paper 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.

Acknowledgments

The authors would like to thank Pawan Goyal, Vijaya Ramachandran, and David Zuckerman for several useful discussions, and Micah Adler, Funda Ergun, and S. Ravikumar for helpful comments on a preliminary draft of this manuscript.

References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley, New York, NY, 1991.

[2] 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.

[3] 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.

[4] E. Berlekamp and L. Welch. Error correction of algebraic block codes. U.S. Patent Number 4,633,470.

[5] 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.

[6] 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.

[7] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25:285–287, 1979.

[8] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, pages 85–111, 1990.

[9] 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.*

[10] 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.

[11] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[12] 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.

[13] 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.

[14] M. O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36:335–348, 1989.

[15] A. G. Ranade. How to emulate shared memory. *Journal of Computer and System Sciences*, 42:307–326, 1991.

[16] 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.

[17] L. Valiant. A combining mechanism for parallel computers. Technical Report TR-24-92, Center for Research in Computing Technology, Harvard University, January 1992.

[18] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

A Large Deviations

We make frequent use of bounds on large deviations for the binomial and hypergeometric distributions, and martingales. The particular form of Theorem A.2 below is from [13].

Theorem A.1 ([6]) *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,*

$$\begin{aligned}
\Pr[X \le (1-\varepsilon)np] &\le e^{-\varepsilon^2 np/2},\ 0 \le \varepsilon \le 1 \\
\Pr[X \ge (1+\varepsilon)np] &\le e^{-\varepsilon^2 np/3},\ 0 \le \varepsilon \le 1 \\
\Pr[X \ge (1+\varepsilon)np] &\le [e^{\varepsilon}(1+\varepsilon)^{-(1+\varepsilon)}]^{np}
\end{aligned}$$

■

Theorem A.2 ([11, 7]) *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 $\varepsilon \ge 0$,*

$$\begin{aligned}
\Pr[t' \ge (p+\varepsilon)s'] &\le e^{-2\varepsilon^2 s'},\ \text{and} \\
\Pr[t' \le (p-\varepsilon)s'] &\le e^{-2\varepsilon^2 s'}.
\end{aligned}$$

■

Theorem A.3 (Azuma's Inequality [1]) *Let $X_0, \ldots, X_k$ be a martingale with $|X_{i+1} - X_i| \le 1$, for all $0 \le i < k$. Then for real $\lambda > 0$,*

$$\Pr\left[|X_k - X_0| > \lambda\sqrt{k}\right] < 2e^{-\lambda^2/2}.$$

■

B Analysis of the uniform experiment

In this section, we analyze a uniform $(X, U, \varepsilon, \tau)$ experiment, defined in Definition 6.1. 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 B.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 B.1.1 *The number of bins that receive at most $x/2$ balls is at most $u/10$.* ■

C Analysis of the random experiment

In this section, we analyze a random $(X, U, \varepsilon, \tau)$ experiment, defined in Definition 6.2. We are interested in bounds on random variables associated with the number of bins that receive at least (or at most) c balls, 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.

We use the theory of martingales in our analysis of the random experiment. Our presentation is based on that of [1]. 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

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

We say that L satisfies the *Lipschitz condition* if $|L(h') - L(h)| \le 1$ whenever h and h' differ only on $B_{i+1} - B_i$. It has been shown that if L satisfies the Lipschitz condition, then $|Z_{i+1} - Z_i| \le 1$ (for example, see Theorem 4.1 of [1]).

Lemma C.1 *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 function associated with Z satisfies the Lipschitz condition. The desired claim follows from Theorem A.3. ∎

We extend the definition of threshold variables to a sequence of s random experiments, given by $(X_0, U_0, \varepsilon, \tau), \ldots, (X_{s-1}, U_{s-1}, \varepsilon, \tau)$. 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 C.2 *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 C.1. ∎

In the remainder of the section, we use Lemmas C.1 and C.2 to obtain high probability bounds on certain threshold variables. Lemmas C.3 and C.4 consider a single random $(X, U, \varepsilon, \tau)$ experiment. As in Section B, 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 C.3 *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:

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

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

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 C.1, the probability that Z is at least $(1/e^c + 1/20 + \lambda)u$ is at most $2e^{-\lambda^2 u/(8c)}$. ■

Corollary C.3.1 *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 $u/10$ whp.* ■

Lemma C.4 *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 $\binom{cu}{2ec}\frac{1}{u^{2ec}} \leq \frac{1}{2^{2ec}}$. Thus, $E[Z]$ is at most $u/2^{2ec}$. By Lemma C.1, the probability that Z exceeds $u(1/2^{2ec} + \lambda)$ is at most $2e^{-\lambda^2 u/(2c)}$. ■

Corollary C.4.1 *If c is sufficiently large and x is at most cu, then the number of bins that receive at least $2ec$ balls is at most $u/10$ whp.* ■

In the following three lemmas, we consider sequences of random $(X, U, \varepsilon, \tau)$ experiments.

Lemma C.5 *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 $c \leq x_i = |X_i| \leq u = |U_i|$ for all i in $[s]$, and $U_i \cap U_j$ is $\emptyset$ for $i \neq j$. Whp, The number of bins in $\cup_{i\in[s]} U_i$ that receive at least c balls is*

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

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 lower bounds on $E[Z_i]$ for all i, and hence a lower bound on $E[Z]$.

Consider the ith experiment, namely the random $(X_i, U, \varepsilon, \tau)$ experiment. Let U_i' be the set of bins such that j is bad for at most $100\varepsilon x_i$ 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

$$
\binom{(1-100\varepsilon)x_i}{c}\left(1-\frac{1}{u}\right)^{x_i-c}\frac{1}{u^c} = \Omega\left((x_i/u)^c\right)
$$

Thus, $E[Z_i]$ is $\Omega(x_i^c/u^{c-1})$, and $E[Z]$ is $\Omega(\sum_{i\in[s]}(x_i^c/u^{c-1}))$. The desired claim follows from Lemma C.2. ■

Lemma C.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 $c \leq x_i = |X_i| \leq u = |U_i|$ for all i in $[s]$, and $U_i \cap U_j$ is $\emptyset$ for $i \neq j$. The number of bins in $\cup_{i\in[s]} U_i$ that receive at least c balls is*

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

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 C.2. ■

Lemma C.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 s, 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 $9u/10$ whp.*

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 C.2, it follows that Z is $9u/10$ whp. ■