

# Accessing Nearby Copies of Replicated Objects in a Distributed Environment

C. Greg Plaxton<sup>1</sup>   Rajmohan Rajaraman<sup>2</sup>   Andréa W. Richa<sup>3</sup>

## Abstract

Consider a set of shared objects in a distributed network, where several copies of each object may exist at any given time. To ensure both fast access to the objects as well as efficient utilization of network resources, it is desirable that each access request be satisfied by a copy “close” to the requesting node. Unfortunately, it is not clear how to efficiently achieve this goal in a dynamic, distributed environment in which large numbers of objects are continuously being created, replicated, and destroyed.

In this paper, we design a simple randomized algorithm for accessing shared objects that tends to satisfy each access request with a nearby copy. The algorithm is based on a novel mechanism to maintain and distribute information about object locations, and requires only a small amount of additional memory at each node. We analyze our access scheme for a class of cost functions that captures the hierarchical nature of wide-area networks. We show that under the particular cost model considered: (i) the expected cost of an individual access is asymptotically optimal, and (ii) if objects are sufficiently large, the memory used for objects dominates the additional memory used by our algorithm with high probability. We also address dynamic changes in both the network as well as the set of object copies.

---

A preliminary version of this paper appeared in *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311-320, June 1997.

<sup>1</sup>Department of Computer Science, University of Texas, Austin, TX 78712. Supported by the National Science Foundation under Grant No. CCR-9504145. Email: [plaxton@cs.utexas.edu](mailto:plaxton@cs.utexas.edu).

<sup>2</sup>College of Computer Science, Northeastern University, Boston, MA 02115. Part of this work was done when the author was at the University of Texas at Austin, with support from the National Science Foundation under Grant No. CCR-9504145, and when the author was at DIMACS Center, Rutgers University. DIMACS is an NSF Science and Technology Center, funded under contract STC-91-19999 and partially supported by the New Jersey Commission on Science and Technology. Email: [rraj@ccs.neu.edu](mailto:rraj@ccs.neu.edu).

<sup>3</sup>Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287. Part of this work was done while the author was at Carnegie Mellon University, supported by National Young Investigator Award under Grant No. CCR-94-57766, and ARPA Contract F33615-93-1-1330. Email: [aricha@asu.edu](mailto:aricha@asu.edu).

## 1 Introduction

The advent of high-speed distributed networks has made it feasible for a large number of geographically dispersed computers to cooperate and share objects (e.g., files, words of memory). Indeed, the last few years have seen the emergence of large distributed databases such as the World Wide Web. The distributed nature of the databases and the rapidly growing demands of the users have in turn overloaded the underlying network resources (e.g., links, memory space at the processors, buffer space at the links and processors). In an attempt to minimize communication delays and to satisfy as many users as possible, strategies for making efficient use of network resources when providing access to shared objects have been devised.

As one might expect, the task of designing efficient algorithms for supporting access to shared objects over wide-area networks is challenging, both from a practical as well as a theoretical perspective. With respect to any interesting measure of performance (e.g., latency, throughput), the optimal bound achievable by a given network is a complex function of many parameters, including edge delays, edge capacities, buffer space, communication overhead, and patterns of user communication. Ideally, we would like to take all of these factors into account when optimizing performance with respect to a given measure. However, such a task may not be feasible in general because the many network parameters interact in a complex manner. For this reason, we adopt a simplified model in which the combined effect of the detailed network parameter values is assumed to be captured by a single function that specifies the cost of communicating a fixed-length message between any given pair of nodes.

**Accessing shared objects.** Consider a set  $\mathcal{A}$  of  $m$  objects being shared by a network  $G$ , where several copies of each object may exist. A shared object may be replicated in order to improve fault-tolerance or performance, for example. In this paper, we consider the basic problem of *reading* objects in  $\mathcal{A}$ . Motivated by the need for efficient network utilization, we seek algorithms that minimize the cost of the read operation. We do not address the *write* operation, which involves the additional consideration of maintaining consistency among the various copies of each object. The problem of consistency, although an important one, is separate from our main concern, namely, that of studying locality. Our results for the read operation apply for the write operation in scenarios where consistency either is not required or is enforced by an independent mechanism.

We differentiate between *shared* and *unshared* copies of objects. A copy is shared if any node can read this copy; it is unshared if only the node that holds the copy may read it. We say that a node  $u$  *inserts* (resp., *deletes*) a copy of object  $A$  (that  $u$  holds) if  $u$  declares the copy shared (resp., unshared).

We refer to the set of algorithms for read, insert, and delete operations as an *access scheme*. Any access scheme that efficiently supports these operations incurs an overhead in memory. It is desirable that this overhead be small, not only because of space considerations, but also because low overhead usually implies fast adaptability to changes in the network topology or in the set of object copies.

The main source of difficulty in designing an access scheme that is efficient with respect to both time and space is the competing considerations of these measures. For example, consider an access scheme in which each node stores the location of each copy of each object in the network. This allows very fast read operations since a node can easily determine the location of the closest copy of any desired object. However, such an access scheme is impractical because it incurs a prohibitively large memory overhead, and every node of the network has to be informed whenever a copy of an object is inserted or deleted. At the other extreme, one might consider an access scheme using no additional memory. In this case insert and delete operations are fast, but read operations are costly since it may be necessary to search the entire network in order to locate a copy of some desired

object.

**Our access scheme.** We design a simple randomized access scheme that exploits locality and distributes control information to achieve low memory overhead. The central part of our access scheme is a mechanism to maintain and locate the addresses of copies of objects. For a single object, say  $A$ , we can provide such a mechanism by the following approach. We embed an  $n$ -node “virtual” height-balanced tree  $T$  one-to-one into the network. Each node  $u$  of the network maintains information associated with the copies of  $A$  residing in the set of nodes that form the subtree of  $T$  rooted at  $u$ . Given the embedding of  $T$ , the read operation may be easily defined as follows. When a node  $u$  attempts to read  $A$ ,  $u$  first checks its local memory for a copy of  $A$  or information about copies of  $A$  in the subtree of  $T$  rooted at  $u$ . If this local check is unsuccessful,  $u$  forwards the request for object  $A$  to its parent.

Naive extensions of the above approach to account for all objects require significant overhead in memory for control information at individual nodes. We overcome this problem by designing a novel method to embed the different trees associated with different objects. Our embedding enables us to define simple algorithms for the read, insert, and delete operations, and to prove their efficiency for a class of cost functions that is appropriate for modeling wide-area networks.

One important property of our access scheme is that it does not require location-dependent naming of the copies of the objects. Thus it avoids renaming a copy of an object every time this copy migrates to another location in the network. Keeping track of replicated copies of the same object may also pose a problem if we have location-dependent naming of the objects, since copies of the same object located at different addresses in the network will have different names. Other properties of our scheme for the restricted class of cost functions considered are that (i) due to its distribution of control information and of shared data, our scheme is expected to avoid “hot-spots” in the network (i.e., heavily accessed nodes); and (ii) due to its distribution of data, combined with its support for object replication and fast adaptability to changes in the network, our scheme is expected to scale well. Scalability is one of the most important problems to be solved in today’s large-scale networks; for example, the World Wide Web, in spite of using scalable components (e.g., clients, servers, TCP/IP connections, DNS), has serious problems of scalability as a whole.

**The cost model.** As indicated above, we assume that a given function determines the cost of communication between each pair of nodes in the network. Our analysis is geared towards a restrictive class of cost functions which we believe to be of practical interest. The precise set of assumptions that we make with respect to the cost function are stated in Section 2. Our primary assumption is that for all nodes  $x$  and costs  $r$ , the ratio of the number of nodes within cost  $2r$  of node  $x$  to the number of nodes within cost  $r$  of node  $x$  is bounded from above and below by constants greater than 1 (unless the entire network is within cost  $2r$  of node  $x$ , in which case the ratio may be as low as 1).

There are several important observations we can make concerning this primary assumption on the cost function. First, a number of commonly studied fixed-connection network families lead naturally to cost functions that satisfy this assumption. For example, fixed-dimension meshes satisfy this assumption if the cost of communication between two nodes is defined as the minimum number of hops between them, and constant-degree trees satisfy this assumption if the cost of communication between two nodes is given by the distance between these nodes in a physical layout (e.g., a wide-area layout, or a VLSI layout) of the tree.

Following the latter example, fat-tree topologies [12] satisfy our assumption if the cost of communication between two nodes is determined by the total cost of a shortest path between the nodes, where the cost assigned to individual edges grows at an appropriate geometric rate as we move higher in the tree. Fat-trees are of particular interest here, because of all the most commonly

studied fixed-connection network families, the fat-tree captures the hierarchical structure of most wide-area networks, and may provide the most plausible approximation to the structure of current such networks.

Even so, it is probably inappropriate to attempt to model the Internet, say, with any kind of uniform topology, including the fat-tree. Note that our assumption on the cost function is purely “local” in nature, and allows for the possibility of a network with a highly irregular global structure. This may be the most important characteristic of our cost model.

**Performance bounds.** We show that our access scheme achieves optimality or near-optimality in terms of several important complexity measures for the restricted class of cost functions discussed above. In particular, our scheme achieves the following bounds:

- The expected cost for any read request is asymptotically optimal.
- The expected cost of an insert (resp., delete) operation at node  $u$  is  $O(C)$  (resp.,  $O(C \log n)$ ), where  $C$  is the maximum cost of communicating a single word message between any two nodes.
- If the number of objects that can be stored at each node is  $q$ , then the additional memory required is  $O(q \log^2 n)$  words whp<sup>1</sup>, where a word is an  $O(\log n)$ -bit string. Thus, if the objects are sufficiently large — i.e.,  $\Omega(\log^2 n)$  words — the memory for objects dominates the additional memory.
- The number of nodes that need to be updated upon the addition or removal of a node is  $O(\log n)$  expected and  $O(\log^2 n)$  whp.

An obvious shortcoming of our analysis is that it only applies to the restricted class of cost functions discussed above. While we do not expect that all existing networks fall precisely within this restricted class, we stress that (i) our access scheme is well-defined, and functions correctly, for arbitrary networks, and (ii) we expect that our access scheme would have good practical performance on any existing network. (Although we have not attempted to formalize any results along these lines, it seems likely that our performance bounds would only degrade significantly in the presence of a large number of nontrivial violations of our cost function assumptions.)

**Related work.** The basic problem of sharing memory in distributed systems has been studied extensively in different forms. Most of the earlier work in this area assumes that each of the nodes of the network has knowledge of a hash function that indicates the location of any copy of any object. Examples of such work include PRAM emulation schemes for completely-connected distributed-memory machines (e.g., [11, 19]) or bounded-degree networks (e.g., [17]), and algorithms for providing concurrent access to a set of shared objects [16].

The basic problem of locating an object arises in every distributed system [14], and was formalized by Mullender and Vitányi [15] as an instance of the distributed matchmaking problem. Awerbuch and Peleg [4], and subsequently Bartal, Fiat, and Rabani [5], and Awerbuch, Bartal, and Fiat [1] give near-optimal solutions in terms of cost to a related problem by defining sparse-neighborhood covers of graphs. Their studies do not address the overhead due to control information, and hence, natural extensions of their results to our problem may require an additional memory of  $m$  words at some node. However, we note that their schemes are designed for arbitrary cost functions, whereas we have focused on optimizing performance for a restricted class of cost functions.

---

<sup>1</sup>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.”

In [2], Awerbuch and Peleg examine the problem of maintaining a distributed directory server, that enables keeping track of mobile users in a distributed network. This problem can be viewed as an object location problem, where objects migrate in the network.

In recent work, access schemes for certain Internet applications have been described in [9, 10, 20]. Some of the ideas in our scheme are similar to those in [20]; however, the two schemes differ considerably in the details. Moreover, the schemes of [9] and [20] have not been analyzed. As in our study, the results of [10] concerning locality assume a restricted cost model. However, their cost model, which is based on the ultrametric, is different from ours. Also, their algorithms are primarily designed for problems associated with “hot spots” (i.e., popular objects).

In [13], Maggs et al. investigate both the problem of determining the placement of copies of the objects in the network, and the problem of devising an efficient access scheme, with the main goal of keeping the edge congestion low. Their work considers cost models that arise in some restricted network topologies, such as trees, meshes, and clustered networks.

A closely related problem is that of designing a dynamic routing scheme for networks [3, 7]. Such a scheme involves maintaining routing tables at different nodes of the network in much the same way as our additional memory. However, in routing schemes the size of additional memory is a function of network size  $n$ , while in our problem the overhead is primarily a function of  $m$ , the number of objects. Straightforward generalizations of routing schemes result in access schemes that require an additional memory of  $m$  words at each node.

The remainder of this paper is organized as follows. Section 2 defines the model of computation. Section 3 gives an informal overview of our access scheme. Section 4 presents a formal description of our access scheme. Section 5 contains a formal statement of the main results. Section 6 gives an informal overview of the analysis of our access scheme. Section 7 presents some preliminary definitions that are used in our analysis. Sections 8 and 9 present a formal analysis and establish the main results. Finally, Section 10 discusses directions for future research.

## 2 Model of Computation

We consider a set  $V$  of  $n$  nodes, each with its own local memory, sharing a set  $\mathcal{A}$  of  $m = \text{poly}(n)$  objects. We define our model of computation by characterizing the following aspects of the problem: (i) objects, (ii) communication, (iii) local memory, (iv) local computation, and (v) complexity measures.

**Objects.** Each object  $A$  has a unique  $(\log m)$ -bit ID. For  $i$  in  $[\log m]$ , let  $A^i$  denote the  $i$ th bit of the ID of  $A$ .<sup>2</sup> Each object  $A$  consists of  $\ell(A)$  words, where a word is an  $O(\log n)$ -bit string.

**Communication.** Nodes communicate with one another by means of messages; each message consists of at least one word. We assume that the underlying network supports reliable communication.

We define the cost of communication by a function  $c : V^2 \mapsto \mathbf{R}$ . For any two nodes  $u$  and  $v$  in  $V$ ,  $c(u, v)$  is the cost of transmitting a single-word message from  $u$  to  $v$ . We assume that  $c$  is symmetric and satisfies the triangle inequality. We also assume for simplicity that for  $u, v$ , and  $w$  in  $V$ ,  $c(u, v) = c(u, w)$  if and only if  $v = w$ . (We make the latter assumption for the sake of convenience only, and with essentially no loss in generality, since an arbitrarily small perturbation in the cost function can be used to break ties.)

The cost of transmitting a message of length  $\ell$  from node  $u$  to node  $v$  is given by  $f(\ell)c(u, v)$ , where  $f : \mathbf{N} \mapsto \mathbf{R}^+$  is any nondecreasing function such that  $f(1) = 1$ .

Given any  $u$  in  $V$  and any real  $r$ , let  $M(u, r)$  denote the set  $\{v \in V : c(u, v) \leq r\}$ . We refer to  $M(u, r)$  as the *ball of radius  $r$  around  $u$* . We assume that there exist real constants  $\delta > 8$  and  $\Delta$

---

<sup>2</sup>For any positive integer  $x$ , we use  $[x]$  to denote the set  $\{0, \dots, x - 1\}$ .

such that for any node  $u$  in  $V$  and any real  $r \geq 1$ , we have

$$\min\{\delta|M(u,r)|, n\} \leq |M(u, 2r)| \leq \Delta|M(u,r)|. \quad (1)$$

**Local Memory.** We partition the local memory of each node  $u$  into two parts. The first part, the *main memory*, stores objects. The second part, the *auxiliary memory*, is for storing possible control information.

**Local Computation.** There is no cost associated with local computation. (Although the model allows an arbitrary amount of local computation at zero cost, our algorithm does not perform any particularly complex local operations.)

**Complexity measures.** We evaluate any solution on the basis of four different complexity measures. The first measure is the cost of reading an object. The second measure is the size of the auxiliary memory at any node. The remaining two measures concern the dynamic nature of the problem: We address the complexity of inserting or deleting a copy of an object, and of adding or removing a network node. The third measure is the cost of inserting or deleting a copy of an object. The fourth measure is the *adaptability*, which is defined as the number of nodes whose auxiliary memory is updated upon the addition or removal of a node. (Our notion of adaptability is analogous to that of [7].)

### 3 Informal Overview of the Access Scheme

The purpose of this section is to provide an informal high-level description of our access scheme. A formal description of the access scheme is given in Section 4. (The reader interested only in the formal description may skip the present section without loss of continuity.)

We begin by considering a simplified version of our access scheme. In this simplified scheme, each of the  $n$  nodes of the network is assigned a  $(\log n)$ -bit label uniformly at random. (Recall that each node also has a unique  $(\log n)$ -bit ID which is independent of this label.) These node labels are then used to construct a “neighbor table” at each node. For each node  $x$ , each integer  $k$  between 1 and  $\log n$ , and each  $k$ -bit string  $\alpha$ , the neighbor table at node  $x$  stores the  $(\log n)$ -bit ID of the *closest* node  $y$  to  $x$  — i.e., the node with minimum  $c(x, y)$  — such that the  $k$ -bit suffix of the label of  $y$  is equal to  $\alpha$ . Let us call this node  $y$  the  $(k, \alpha)$ -neighbor of  $x$ . (If no such node  $y$  exists, then let  $k'$  be the largest integer such that the label of some node matches  $\alpha$  in the  $k'$  lowest bit positions, let  $S$  denote the set of such nodes, and define the  $(k, \alpha)$ -neighbor of  $x$  as the closest node in  $S$  to  $x$ .) Thus the neighbor table of any node has a large number of entries, namely  $\Theta(n)$ . This large table size is the main technical deficiency of the simplified access scheme; for ease of reference, we refer to this deficiency as Problem 1 in the present section. As we develop our (non-simplified) access scheme we will address Problem 1 by reducing the size of the neighbor table to  $\Theta(\log n)$ . But let us ignore Problem 1 for the moment and continue to consider the simplified scheme. Throughout the following discussion, let  $A$  be an arbitrary object and let  $\alpha_k$  denote the  $k$ -bit suffix of the unique ID of object  $A$ .

To insert a copy of object  $A$  at node  $x$ , we place a pointer to this copy at up to  $\log n$  nodes of the network. (To delete a copy, we remove this set of pointers.) These nodes are determined as follows: For each  $k$  between 1 and  $\log n$ , we place a pointer at the  $(k, \alpha_k)$ -neighbor of  $x$ . The probability that the  $k$ -bit suffix of the label of an arbitrary node matches a particular  $k$ -bit string  $\alpha_k$  is  $2^{-k}$ . Consider a ball  $B$  around node  $x$  containing exactly  $2^k$  nodes. Note that there is a constant probability (approximately  $1/e$ ) that no node in  $B$  matches  $\alpha_k$ . Thus the radius of  $B$  is a lower bound (up to a constant factor) on the expected distance from  $x$  to its  $(k, \alpha_k)$ -neighbor. Is this radius also an upper bound? Not for an arbitrary metric, since (for example) the diameter of the smallest ball around  $x$  containing  $2^k + 1$  nodes can be arbitrarily larger than the diameter

of the smallest ball around  $x$  containing  $2^k$  nodes. Under the assumption of the left inequality of Equation 1, however, it can be shown that the radius of  $B$  provides a tight bound on the expected distance from  $x$  to its  $(k, \alpha_k)$ -neighbor. Furthermore, the right inequality of Equation 1 implies that the expected distance from  $x$  to its  $(k, \alpha_k)$ -neighbor is geometrically increasing in  $k$ . The latter observation turns out to be useful because it implies that the expected total distance from  $x$  to its  $(i, \alpha_i)$ -neighbor, summed over all  $i$  such that  $0 \leq i \leq k$ , is dominated by (i.e., within a constant factor of) the expected distance from  $x$  to its  $(k, \alpha_k)$ -neighbor.

We now describe how to process a read request for object  $A$  at a node  $x$  in our simplified access scheme. If  $x$  already has a copy of  $A$ , then it processes the read request locally. Otherwise,  $x$  processes the read request as follows, where  $x_k$  denotes the  $(k, \alpha_k)$ -neighbor of  $x$ ,  $1 \leq k \leq \log n$ . First,  $x$  consults  $x_1$ . If  $x_1$  has a pointer to a copy of  $A$  that is no more than a constant factor further away from  $x_1$  than  $x$ , then  $x_1$  returns a pointer to such a copy to  $x$ ,  $x$  retrieves this copy, and the read is complete. If  $x_1$  does not have such a pointer,  $x$  consults  $x_2$ , and so on. (In general, if  $x_i$  has a pointer to a copy of  $A$  that is no more than a constant factor further away from  $x_i$  than  $x$ , then  $x_i$  returns a pointer to such a copy to  $x$ ,  $x$  retrieves this copy, and the read is complete.) A deficiency of this scheme is that we may fail to locate a pointer to  $A$  at  $x_1$  through  $x_{\log n}$  even though there are one or more copies of  $A$  in the network. Fortunately, this deficiency may be easily rectified by making a slight modification to the definition of  $x_{\log n}$ . Thus we ignore this issue in the present informal discussion.

In general, if  $x$  completes the read operation for  $A$  by using a pointer found at  $x_k$ , the total cost of the read is proportional to the sum of the distances from  $x$  to  $x_i$ ,  $1 \leq i \leq k$ , and the distance from  $x$  to the copy of  $A$  retrieved. By the definition of the algorithm the latter distance cannot dominate. Furthermore, as discussed earlier, the expectation of the summation is dominated by the expected distance from  $x$  to  $x_k$ . Our main objective is to prove that the expected cost of the read operation is optimal, that is, proportional to the distance to the closest copy of object  $A$ . Thus, it is sufficient for us to prove that the distance from  $x$  to the closest copy of object  $A$  is proportional to the expected distance from  $x$  to  $x_k$ . Let  $y$  denote the node holding the closest copy of  $A$  to  $x$ , and let  $j$  be the least integer such that the expected distance from  $x$  to  $x_j$  is at least the distance from  $x$  to  $y$ . The intuition underlying our claim that the expected distance from  $x$  to  $x_k$  is proportional to the distance from  $x$  to  $y$  is that for all  $i \geq j$ , there is a reasonable chance that  $x_i$  is the  $(i, \alpha_i)$ -neighbor of  $y$  (in addition to  $x$ ). Our approach to formalizing this intuition involves showing that, under the assumption of Equation 1, the sets of nodes contained in two balls of radius  $\Theta(r)$  centered at nodes  $r$  apart (e.g., at  $x$  and  $y$ ) have a substantial (i.e., constant fraction) intersection. We then argue that there is a constant probability that the read operation terminates successfully at each successive node  $x_i$  such that  $i \geq j$ . Such an argument yields the desired main theorem (i.e., expected optimal cost for the read operation) as long as the probability of success at each successive iteration is sufficiently close to 1 to offset the geometrically increasing expected cost of visiting each successive  $x_i$ . (For example, if the expected distance from  $x$  to  $x^i$  is  $2^i$ , then the success probability has to be strictly greater than  $1/2$ .) In the discussion that follows, we refer to this problem of achieving a sufficiently large constant success probability as Problem 2.

The foregoing completes our description and discussion of our simplified access scheme. We now sketch the design of our (non-simplified) access scheme. We focus the discussion on our approach to overcoming Problems 1 and 2, the two main technical problems identified above. As mentioned above, we address Problem 1 by replacing the  $O(n)$ -entry neighbor tables defined above with  $O(\log n)$ -entry tables. The basic idea underlying this exponential reduction in the table size is straightforward. Consider a ball containing a set  $S$  of  $2^k$  nodes, and notice that for any  $(\log n)$ -bit string  $\alpha$ , many of the  $(\log n, \alpha)$ -neighbors of the nodes in set  $S$  are likely to be the same. In

fact, for any  $i \geq k$  and any  $i$ -bit string  $\alpha$ , many of the  $(i, \alpha)$ -neighbors of the nodes in set  $S$  are likely to be the same. Furthermore, to the extent that such  $(i, \alpha)$ -neighbors differ, it is not clear that these differences are crucial to the performance of the algorithm. These observations suggest that it may be possible to reduce the table size by allowing clusters of nodes such as  $S$  to share common values. In Section 4 we present a simple method for achieving this kind of sharing across clusters. An interesting and important feature of this method is that it does not explicitly partition the network into clusters. (Such a global partition might be difficult to maintain efficiently in a highly dynamic network.) Instead, our method is completely distributed and derives its efficiency from simple properties of the randomly-assigned node labels. For example, we exploit the fact that in any fixed set of  $2^k$  nodes, the expected number of occurrences of any given  $k$ -bit suffix is 1; it follows that the random node labels can be used to partition the “shared” table entries within any cluster.

Reducing the table size in the above manner solves Problem 1 but introduces another technical difficulty which we now describe and refer to as Problem 3. Using the revised tables, it turns out that the algorithm for performing a read operation is changed in a significant manner. In the simplified scheme described above,  $x$  successively contacts nodes  $x_1, x_2$ , et cetera until a suitable pointer to the desired object  $A$  is found. The actual access scheme is similar except that if a suitable pointer is not found at a node  $z$  then the read request is forwarded to a suitable neighbor of  $z$ , as opposed to a neighbor of  $x$ . (The placement of pointers on an insertion is modified in a similar manner.) As a result, the sequence of nodes along which the read request is forwarded in our access scheme is different from the sequence of nodes  $x_1, x_2$ , et cetera, that is used in the simplified access scheme. Let us denote the  $i$ th node in the new sequence by  $x'_i$ . Unfortunately, this change in the access scheme tends to increase the expected number of iterations until a given read operation succeeds. Informally, the reason for this increase is that the neighbors of  $x'_i$  are defined in terms of distance from  $x'_i$ , and for the purposes of a read operation originating at node  $x$ , distance from  $x$  is a more useful metric than distance from  $x'_i$ . The problem of bounding the increase in the expected cost of a read operation due to forwarding, which we refer to in the present section as Problem 3, is one of the main technical challenges faced in our analysis.

Aside from the reduced neighbor tables alluded to above, there are two main differences between the simplified access scheme described in this section and the access scheme of Section 4. These differences are designed to help us overcome Problems 2 and 3, respectively. In the following paragraphs we briefly sketch these differences and provide some intuition underlying their relevance to Problems 2 and 3.

The first difference is that, in the access scheme of Section 4, we maintain a sufficiently large constant number of neighbors in each table entry, rather than a single neighbor. In other words, rather than storing the ID of the closest node with a label satisfying a certain condition, we store the IDs of the  $d + 1$  closest nodes satisfying this condition, for some sufficiently large constant  $d$ . Accordingly, when a copy of an object is inserted, we place pointers to the copy at  $d + 1$  times as many nodes as in the simplified scheme. Furthermore, at each iteration of our algorithm for processing a read request, we search for an appropriate pointer to a copy of the object at  $d + 1$  nodes instead of one. By leaving more pointers and expanding our search for pointers in this manner, iterations at which a read operation would otherwise have had only a small constant success probability now can have a success probability close to 1. Such an improvement in the success probability is clearly helpful for dealing with Problem 2. Furthermore, the cost of this change in the access scheme is at most a constant factor in terms of both time and space.

The second difference is that, in the access scheme of Section 4, we view the random node labels as base- $2^b$  numbers for some sufficiently large constant  $b$ , rather than as base-2 numbers. Thus,



for example, the read operation proceeds in  $\lceil \log n/b \rceil$  iterations rather than in  $\log n$  iterations. As in the simplified access scheme, the ratio between the expected distance from  $x$  to  $x'_{i+1}$  and the expected distance from  $x$  to  $x'_i$  is a constant greater than 1, i.e., these distances are geometrically increasing. However, for  $b > 0$ , this ratio is a much larger constant. Increasing this ratio is helpful for dealing with Problem 3, informally because the effect of forwarding becomes negligible: The distance from  $x$  to  $x'_i$  tends to be so much smaller than the distance from  $x$  to  $x'_{i+1}$  that the ratio between the distance from  $x$  to  $x'_{i+1}$  (which is important to analyze when bounding the cost of a read from  $x$ ) and the distance from  $x'_i$  to  $x'_{i+1}$  (which is easier to analyze because it is more closely related to the table entries used by the algorithm, due to forwarding) is typically close to 1. Consequently, we are able to show that the “drift” of the sequence  $x'_1, x'_2$ , et cetera, from the sequence that we would get if requests were not forwarded and instead sent to neighbors of  $x$  (as in the simplified scheme) is small enough that the increase in the expected cost of a read operation due to forwarding is small.

#### 4 The Access Scheme

In this section, we present our access scheme for shared objects. We assume that  $n$  is a power of  $2^b$ , where  $b$  is a fixed positive integer to be specified later (see the beginning of Section 7). For each node  $x$  in  $V$ , we assign a label independently and uniformly at random from  $[n]$ . For  $i$  in  $[\log n]$ , let  $x^i$  denote the  $i$ th bit of the label of  $x$ . Note that the label of a node  $x$  is independent of the unique  $(\log n)$ -bit ID of the node. For all  $x$  in  $V$  (resp.,  $A$  in  $\mathcal{A}$ ), we define  $x[i]$  as the nonnegative integer with binary representation  $x^{(i+1)b-1} \dots x^{ib}$  (resp.,  $A[i]$  denotes  $A^{(i+1)b-1} \dots A^{ib}$ ), for  $i$  in  $[(\log n)/b]$ . We also assign a total order to the nodes in  $V$ , given by the bijection  $\beta : V \rightarrow [n]$ .

We partition the auxiliary memory of each node in two parts, namely the *neighbor table* and the *pointer list* of the node.

- **Neighbor table.** For each node  $x$ , the neighbor table of  $x$  consists of  $(\log n)/b$  levels. The  $i$ th level of the table,  $i$  in  $[(\log n)/b]$ , consists of *primary*, *secondary*, and *reverse*  $(i, j)$ -neighbors, for all  $j$  in  $[2^b]$ .

The *primary*  $(i, j)$ -neighbor  $y$  of  $x$  is such that  $y[k] = x[k]$  for all  $k$  in  $[i]$ , and either (i)  $i < (\log n)/b - 1$  and  $y$  is the node of minimum  $c(x, y)$  such that  $y[i] = j$ , if such a node exists, or (ii)  $y$  is the node with largest  $\beta(y)$  among all nodes  $z$  such that  $z[i]$  matches  $j$  in the largest number of rightmost bits. Note that the primary  $(i, j)$ -neighbor of a node  $x$  is guaranteed to exist, since  $x$  itself is a candidate node. Let  $d$  be a fixed positive integer, to be specified later (see the beginning of Section 7). Let  $y$  be the primary  $(i, j)$ -neighbor of  $x$ . If  $y[i] = j$ , then let  $W_{i,j}$  denote the set of nodes  $w$  in  $V \setminus \{y\}$  such that  $w[k] = x[k]$  for all  $k$  in  $[i]$ ,  $w[i] = j$ , and  $c(x, w)$  is at most  $O(c(x, y))$ . Otherwise, let  $W_{i,j}$  be the empty set.

The set of *secondary*  $(i, j)$ -neighbors of  $x$  is the subset  $U$  of  $\min\{d, |W_{i,j}|\}$  nodes  $u$  with minimum  $c(x, u)$  in  $W_{i,j}$ ; that is,  $c(x, u)$  is at most  $c(x, w)$  for all  $w$  in  $W_{i,j} \setminus U$ , and for all  $u$  in  $U$ . Finally, a node  $w$  is a *reverse*  $(i, j)$ -neighbor of  $x$  if and only if  $x$  is a primary  $(i, j)$ -neighbor of  $w$ .

In Figure 1, we illustrate the primary neighbors entries in the neighbor table of node  $x$  for  $b = 1$ ; suppose the level  $i$ -neighbors of  $x$  in the table are given by (i) above.

- **Pointer list.** Each node  $x$  also maintains a pointer list  $Ptr(x)$  with pointers to copies of some objects in the network. Formally,  $Ptr(x)$  is a set of triples  $(A, y, k)$ , where  $A$  is in  $\mathcal{A}$ ,  $y$  is a node that holds a copy of  $A$ , and  $k$  is an upper bound on the cost  $c(x, y)$ . We maintain the invariant that there is at most one triple associated with any object in  $Ptr(x)$ . The

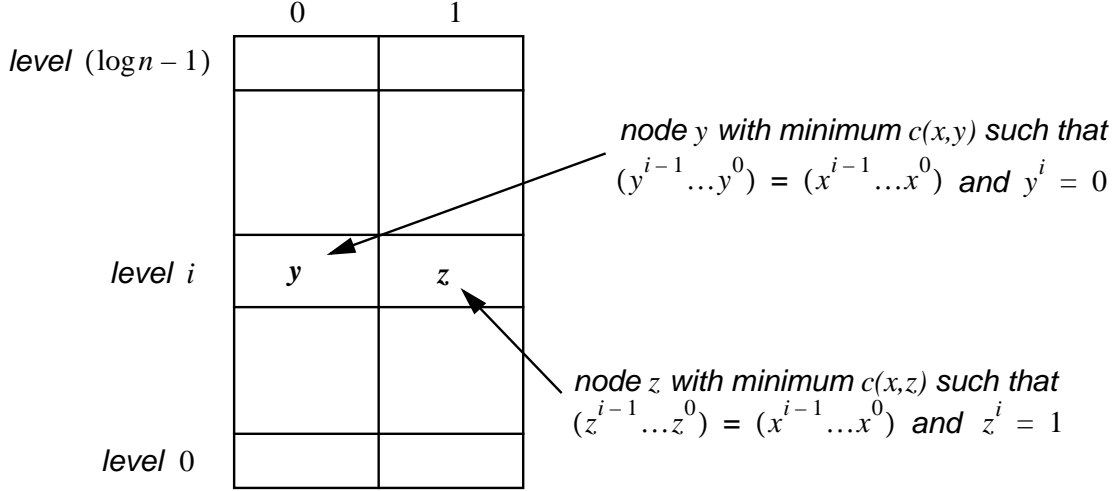


Figure 1: The primary neighbor table of node  $x$ , for  $b = 1$ .

pointer list of  $x$  may only be updated as a result of insert and delete operations. All of the pointer lists can be initialized by inserting each shared copy in the network at the start of the computation. We do not address the cost of initializing the auxiliary memories of the nodes.

Let  $r$  be the node whose label matches (in terms of binary representation) the ID of  $A$  in the largest number of rightmost bits. (In case of a tie between several nodes  $r_0, \dots, r_k$ , let  $r$  be the unique node  $r_i$  maximizing  $\beta(r_i)$ .) We call  $r$  the *root* node for object  $A$ . The uniqueness of the root node for each  $A$  in  $\mathcal{A}$  is crucial to guarantee the success of every read operation.

In this section and throughout the paper, we use the notation  $\langle \alpha \rangle_k$  to denote the sequence  $\alpha_0, \dots, \alpha_k$  (of length  $k + 1$ ). When clear from the context,  $k$  will be omitted. In particular, a *primary neighbor sequence* for  $A$  is a maximal sequence  $\langle u \rangle_k$  such that  $u_0$  is in  $V$ ,  $u_k$  is the root node for  $A$ , and  $u_{i+1}$  is the primary  $(i, A[i])$ -neighbor of  $u_i$ , for all  $i$ . It is worth noting that the sequence  $\langle u \rangle_k$  is such that the label of node  $u_i$  satisfies  $(u_i[i - 1], \dots, u_i[0]) = (A[i - 1], \dots, A[0])$ , for all  $i$ . We now give an overview of the read, insert, and delete operations.

- **Read.** Consider a node  $x$  attempting to read an object  $A$ . The read operation proceeds by successively forwarding the *read request* for object  $A$  originating at node  $x$  along the primary neighbor sequence  $\langle x \rangle$  for  $A$  with  $x_0 = x$ . When forwarding the read request, node  $x_{i-1}$  informs  $x_i$  of the current best upper bound  $k$  on the cost of sending a copy of  $A$  to  $x$ . On receiving the read request with associated upper bound  $k$ , node  $x_i$  proceeds as follows. If  $x_i$  is the root node for  $A$ , then  $x_i$  requests that the copy of  $A$  associated with the current best upper bound  $k$  be sent to  $x$ . Otherwise,  $x_i$  communicates with its primary and secondary  $(i, A[i])$ -neighbors to check whether the pointer list of any of these neighbors has an entry  $(A, z, k_1)$  such that  $k_1$  is at most  $k$ . Then,  $x_i$  updates  $k$  to be the minimum of  $k$  and the smallest value of  $k_1$  thus obtained (if any). If  $k$  is within a constant factor of the cost of following  $\langle x \rangle$  up to  $x_i$ , that is,  $k$  is  $O(\sum_{j=0}^{i-1} c(x_j, x_{j+1}))$ , then  $x_i$  requests that the copy of  $A$  associated with the upper bound  $k$  be sent to  $x$ . Otherwise,  $x_i$  forwards the read request to  $x_{i+1}$ .

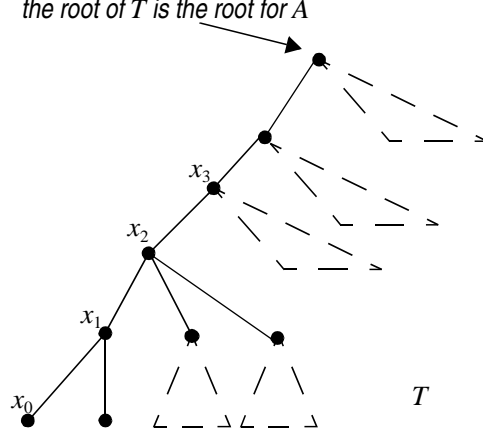


Figure 2: The tree  $T$  associated with object  $A$  and the primary neighbor sequence for  $x = x_0$ .

Relating to the more general description of the read operation of our scheme described in Section 1, the tree  $T$  associated with object  $A$  is given by the following rule: The parent of node  $x$  in  $T$  is the primary  $(i, A[i])$ -neighbor of  $x$ , where  $i$  is the maximum index such that  $(x[i-1], \dots, x[0]) = (A[i-1], \dots, A[0])$ ; in other words, the parent of node  $x$  is the node  $x_i$  in the primary neighbor sequence  $\langle x \rangle$  for  $A$  with  $x = x_0$ . Figure 2 illustrates the tree  $T$  and the sequence  $\langle x \rangle$ .

- **Insert.** An *insert request* for object  $A$  generated by node  $y$  updates the pointer lists of the nodes in some prefix of the primary neighbor sequence  $\langle y \rangle$  for  $A$  with  $y_0 = y$ . When such an update arrives at a node  $y_i$  by means of an insert message,  $y_i$  updates its pointer list if the upper bound  $\sum_{j=0}^{i-1} c(y_j, y_{j+1})$  on the cost of getting object  $A$  from  $y$  is smaller than the current upper bound associated with  $A$  in this list. In other words,  $y_i$  updates  $Ptr(y_i)$  if  $(A, \cdot, \cdot)$  is not in this list, or if  $(A, \cdot, k)$  is in  $Ptr(y_i)$  and  $k$  is greater than  $\sum_{j=0}^{i-1} c(y_j, y_{j+1})$ . Node  $y_i$  forwards the insert request to node  $y_{i+1}$  only if  $Ptr(y_i)$  is updated.
- **Delete.** A *delete request* for object  $A$  generated by node  $y$  eventually removes all triples of the form  $(A, y, \cdot)$  from the pointer lists  $Ptr(y_i)$ , where  $\langle y \rangle$  is the primary neighbor sequence for  $A$  with  $y_0 = y$ , making the copy of  $A$  at  $y$  unavailable to other nodes in the network. Upon receiving such a request by means of a delete message, node  $y_i$  checks whether the entry associated with  $A$  in its pointer list is of the form  $(A, y, \cdot)$ . If it is not, the delete procedure is completed and we need not proceed further in updating the pointer lists in  $\langle y \rangle$ . Otherwise,  $y_i$  deletes this entry from its pointer list, and checks for entries associated with  $A$  in the pointer lists of its reverse  $(i-1, A[i-1])$ -neighbors. If an entry is found,  $y_i$  updates  $Ptr(y_i)$  by adding the entry  $(A, w, k + c(w, y_i))$ , where  $w$  is the reverse  $(i-1, A[i-1])$ -neighbor of  $y_i$  with minimum upper bound  $k$  associated with  $A$  in its pointer list. A delete message is then forwarded to  $y_{i+1}$ .

The read, insert, and delete operations are summarized in Figures 3, 4, and 5. The messages and requests in the figure are all with respect to object  $A$ . A *read request* is generated by node  $x$  when  $x (= x_0)$  sends a message  $Read(x, \infty, \cdot)$  to itself, if  $x$  does not hold a copy of  $A$ . A read message  $Read(x, k, y)$  indicates that (i) a read request for object  $A$  was generated at node  $x$ , and (ii) the current best upper bound on the cost of reading a copy of  $A$  is  $k$ , the cost of accessing

the copy at  $y$ . An *insert* (resp., *delete*) request is generated when node  $y$  ( $= y_0$ ) sends a message *Insert*( $y, 0$ ) (resp., *Delete*( $y$ )) to itself. An insert message *Insert*( $y, k$ ) indicates to its recipient node  $z$  that the best known upper bound on the cost incurred by bringing the copy of  $A$  located at  $y$  to the node  $z$  is  $k$ . We assume that  $y$  holds a copy of  $A$  and that this copy is unshared (resp., shared) when an insert (resp., delete) request for  $A$  is generated at  $y$ .

*Action of  $x_i$  on receiving a message  $Read(x, k, y)$ .*

If  $i > 0$  and  $x_i[i - 1] \neq A[i - 1]$ , or  $i = (\log n)/b - 1$  (that is,  $x_i$  is the root for  $A$ ) then

- Node  $x_i$  sends a message *Satisfy*( $x$ ) to node  $v$  such that  $(A, v, \cdot)$  is in  $Ptr(x_i)$ , requesting it to send a copy of  $A$  to  $x$ . If  $Ptr(x_i)$  has no such entry, then there are no shared copies of  $A$ .

Otherwise

- Let  $U$  be the set of secondary  $(i, A[i])$ -neighbors of  $x_i$ . Node  $x_i$  requests a copy of  $A$  with associated upper bound at most  $k$  from each node in  $U \cup \{x_{i+1}\}$ .
- Each node  $u$  in  $U \cup \{x_{i+1}\}$  responds to the request message received from  $x_i$  as follows: If there exists an entry  $(A, v, q_v)$  in  $Ptr(u)$  and if  $q'_v = q_v + c(x_i, u) + \sum_{j=0}^{i-1} c(x_j, x_{j+1})$  is at most  $k$ , then  $u$  sends a success message *Success*( $v, q'_v$ ) to  $x_i$ .
- Let  $U'$  be the set of nodes  $u$  from which  $x_i$  receives a response message *Success*( $u, k_u$ ). If  $U'$  is not empty, then  $x_i$  updates  $(k, y)$  to be  $(k_z, z)$ , where  $z$  is a node with minimum  $k_u$  over all  $u$  in  $U'$ .
- If  $k = O(\sum_{j=0}^{i-1} c(x_j, x_{j+1}))$  then  $x_i$  sends a message *Satisfy*( $x$ ) to node  $y$ , requesting  $y$  to send a copy of  $A$  to  $x$ . Otherwise,  $x_i$  forwards a message *Read*( $x, k, y$ ) to  $x_{i+1}$ .

Figure 3: Action on receiving a message *Read* for object  $A$ .

*Action of  $y_i$  on receiving a message  $Insert(y, k)$ .*

If  $(A, \cdot, \cdot)$  is not in  $Ptr(y_i)$ , or  $(A, \cdot, k')$  is in  $Ptr(y_i)$  and  $k' > k$ , then

- Node  $y_i$  accordingly creates or replaces the entry associated with  $A$  in  $Ptr(y_i)$  by inserting  $(A, y, k)$  into this list.
- If  $y_i[i - 1] = A[i - 1]$  then  $y_i$  sends a message *Insert*( $y, k + c(y_i, y_{i+1})$ ) to  $y_{i+1}$ .

Figure 4: Actions on receiving a message *Insert* for object  $A$ .

The correctness of our access scheme follows from the two points below:

1. The insert and delete procedures maintain the following invariants. For any  $A$  in  $\mathcal{A}$  and any  $y$  in  $V$ , there is at most one entry associated with  $A$  in the pointer list of  $y$ . If  $y$  holds a shared copy of  $A$  and  $\langle y \rangle$  is the primary neighbor sequence for  $A$  with  $y_0 = y$ , then (i) there is an entry associated with  $A$  in the pointer list of every node in  $\langle y \rangle$ , and (ii) the nodes that have a pointer list entry associated with the copy of  $A$  at  $y$  form a prefix subsequence of  $\langle y \rangle$ . The preceding claims follow directly from the insert and delete procedures as described.
2. Every read request for any object  $A$  by any node  $x$  is successful. That is, it locates and brings to  $x$  a shared copy of  $A$ , if such a copy is currently available. The read operation proceeds by following the primary neighbor sequence  $\langle x \rangle$  for  $A$  with  $x_0 = x$ , until either a copy of  $A$

*Action of  $y_i$  on receiving a message  $Delete(y)$ .*

If  $(A, y, \cdot)$  is in  $Ptr(y_i)$  then

- Let  $U$  be the set of reverse  $(i - 1, A[i - 1])$ -neighbors of  $y_i$ . Node  $y_i$  removes  $(A, y, \cdot)$  from  $Ptr(y_i)$ , and requests a copy of  $A$  from each  $u$  in  $U$ .
- Each  $u$  in  $U$  responds to the request message from  $y_i$  by sending a message  $Success(v, q_v + c(y_i, u))$  to  $y_i$  if and only if  $(A, v, q_v)$  is in  $Ptr(u)$ .
- Let  $U'$  be the set of nodes  $u$  such that  $y_i$  receives a message  $Success(u, k_u)$  in response to the request message it sent. If  $|U'| > 0$  then  $y_i$  inserts  $(A, w, k_w)$  into  $Ptr(y_i)$ , where  $w$  is the node in  $U'$  such that  $k_w \leq k_u$ , for all  $u$  in  $U'$ .
- If  $y_i[i - 1] = A[i - 1]$  then  $y_i$  sends a message  $Delete(y)$  to  $y_{i+1}$ .

Figure 5: Actions on receiving a message  $Delete$  for object  $A$ .

is located or the root for  $A$  is reached. By point 1 above, there exists a shared copy of  $A$  in the network if and only if the root for  $A$  has a pointer to it.

## 5 Performance Bounds

In this section, we state our main claims regarding the performance of our access scheme. In Theorems 1 through 4 below, we state bounds on the cost of a read, the cost of an insert or delete, the size of auxiliary memory, and the adaptability of our access scheme.

**Theorem 1** *Let  $x$  be any node in  $V$  and let  $A$  be any object in  $\mathcal{A}$ . If  $y$  is the node with minimum  $c(x, y)$  that holds a shared copy of  $A$ , then the expected cost of satisfying a read request for  $A$  by  $x$  is  $O(f(\ell(A))c(x, y))$ .*

Let  $C$  denote  $\max\{c(u, v) : u, v \in V\}$ . If a node  $x$  tries to read an object  $A$  for which there is currently no shared copy in the network, then the expected cost of the read operation is  $O(C)$ .

**Theorem 2** *The expected cost of an insert operation is  $O(C)$ , and that of a delete operation is  $O(C \log n)$ .*

**Theorem 3** *Let  $q$  be the number of objects that can be stored in the main memory of each node. The size of the auxiliary memory at each node is  $O(q \log^2 n)$  words whp.*

**Theorem 4** *The adaptability of our scheme is  $O(\log n)$  expected and  $O(\log^2 n)$  whp.*

## 6 Informal overview of the analysis

In this section, we give an informal overview of the analysis of our access scheme. Section 6.1 gives an overview of the analysis of the read operation. Section 6.2 outlines our analysis for the other performance bounds. The reader interested only in the formal analysis of our access scheme may proceed directly to Section 7 without loss of continuity.

### 6.1 The read operation

Suppose node  $x$  issues a read request for an object  $A$  for which the closest copy lies at node  $y$  (Recall that a closest node  $y$  to  $x$  is the node with minimum  $c(x, y)$  that holds a copy of  $A$ .) Let  $\langle x \rangle$  (resp.,  $\langle y \rangle$ ) denote the primary neighbor sequence associated with object  $A$  and node  $x = x_0$  (resp.,  $y = y_0$ ). Since a copy of  $A$  was inserted at  $y$  prior to the read request, it is guaranteed that

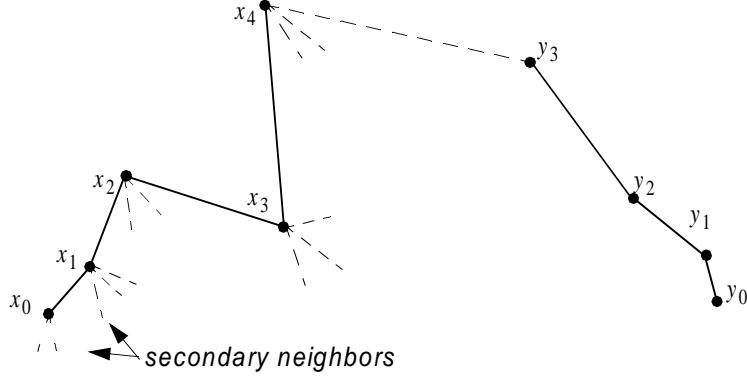


Figure 6: A read request for object  $A$  is forwarded until a pointer to a copy of  $A$  is found.

any node  $y_j$  in the sequence  $\langle y \rangle$  has a pointer to a copy of  $A$  that is at  $y$  or at a node closer to  $y_j$  than  $y$ . The read operation proceeds by forwarding the request along the nodes in the sequence  $\langle x \rangle$  until a pointer to a copy of  $A$  is obtained. In fact, when the read request is forwarded to a node  $x_i$  in the sequence  $\langle x \rangle$ ,  $x_i$  checks both its auxiliary memory as well as its secondary neighbors for a pointer to a copy of  $A$ . If the pointer is present at any of these nodes, then the object is fetched from the appropriate node. Otherwise, the request is forwarded to the next “level”. Figure 6 illustrates the neighbor sequences and the forwarding process. In the figure, a read request for object  $A$  generated at node  $x = x_0$  is forwarded along  $\langle x \rangle$  until a pointer to the copy of  $A$  residing at node  $y = y_0$  is found at node  $y_3$ , which is also a secondary neighbor of  $x_4$ .

An important part of our analysis is to place an upper bound  $\tau$  on the smallest level in the neighbor sequence of  $x$  where a pointer to the object can be found. Of course, owing to the randomization used by our access scheme, any such upper bound is probabilistic. In other words, we can only characterize the probability distribution of  $\tau$ . Moreover, the probability distribution of  $\tau$  does not directly yield the cost incurred for the read operation since the cost incurred at any given level is itself a random variable. While we expect the cost incurred at the highest level to be the dominating term in the overall cost, we need to formally incorporate this intuition.

**Balls defined by the primary neighbor sequences.** We characterize the probability distribution of  $\tau$  by analyzing certain balls that are defined by the sequences  $\langle x \rangle$  and  $\langle y \rangle$ . For any nonnegative integer  $i$ , let  $A_i$  (resp.,  $D_i$ ) denote the ball of smallest radius around  $x_i$  (resp.,  $y_i$ ) that contains  $x_{i+1}$  (resp.,  $y_{i+1}$ ). Let  $B_i$  (resp.,  $E_i$ ) denote the set  $\cup_{0 \leq j \leq i} A_j$  (resp.,  $\cup_{0 \leq j \leq i} D_j$ ). Let  $C_i$  denote the ball of smallest radius around  $x_i$  that contains all of the secondary  $(i, A[i])$ -neighbors of  $x_i$ . If we select  $\tau$  such that  $C_\tau$  is a superset of  $D_\tau$ , then the node  $y_{\tau+1}$  is in  $C_\tau$ , implying that  $\tau$  is an upper bound on the smallest level at which a pointer to  $A$  is guaranteed to be found. Therefore, the cost incurred by the forwarding and request messages is within a constant factor (that depends on the constant  $d$ ) of the sum of the radii of the  $C_i$ ’s,  $0 \leq i \leq \tau$ . Moreover, if  $y_{\tau+1}$  has a pointer to a copy of  $A$  at node  $z$ , then the insert operation ensures that  $c(y_\tau, z)$  is at most the sum of the radii of the  $D_i$ ’s,  $0 \leq i \leq \tau$ . Therefore, the total cost of the read operation is within a constant factor of the sum of the radii of the  $C_i$ ’s and the  $D_i$ ’s,  $0 \leq i \leq \tau$ .

It follows from the discussion in the preceding paragraph that we can estimate  $\tau$  and the total cost of the read operation by determining bounds on the sizes of the balls  $C_i$  and  $D_i$ . We proceed by identifying balls centered around  $x$  (resp.,  $y$ ) that are “expected” to contain the balls  $A_i$  (resp.,  $D_i$ ) and  $C_i$ , and the set  $B_i$  (resp.,  $E_i$ ) for a given  $i$ . Consider the balls  $C_i$  and  $D_i$ . The expected

size of each ball is  $\Theta(2^{(i+1)b})$ . By choosing  $b$  to be sufficiently large, we ensure that the expected size of  $C_i$  (resp.,  $D_i$ ) grows rapidly enough with  $i$  that the expected radius of  $C_i$  (resp.,  $D_i$ ) is much greater than the expected distance from  $x$  to  $x_i$  (resp.,  $y$  to  $y_i$ ). As a result, if the sizes of the balls at each level are close to their respective expected values, then we see that  $C_i$  is a superset of a ball  $X_i$  and a subset of a ball  $X'_i$ , where both  $X_i$  as well as  $X'_i$  consist of  $\Theta(2^{(i+1)b})$  closest neighbors of  $x$ . Similarly, we also expect that  $D_i$  is a subset of a ball  $Y_i$  which consists of  $\Theta(2^{(i+1)b})$  closest neighbors of  $y$ . Moreover, since the ball  $C_i$  includes  $d$  secondary neighbors of  $x_i$  while  $D_i$  is defined only by the primary neighbor of  $y_i$ , we can choose  $d$  sufficiently large such that  $X_i$  is a constant factor bigger than  $Y_i$ . Let  $i^*$  be the smallest  $i$  such that the radius of  $X_i$  is  $\Theta(c(x, y))$ . By Equation 1, an appropriate choice of constants (depending on  $\Delta$ ) in the definitions of  $X_i$  and  $Y_i$  implies that for all  $i$  greater than or equal to  $i^*$ ,  $X_i$  is a superset of  $Y_i$ . Thus, we can now define  $\tau$  as the smallest  $i \geq i^*$  for which both  $C_i \supseteq X_i$  as well as  $D_i \subseteq Y_i$  hold.

Loosely speaking,  $X_i$  and  $Y_i$  may be thought of as the expected values of the balls that are obtained during the processing of a read request and an insert request, respectively, in the simplified access scheme described in Section 3. Recall that in the simplified scheme, node  $x$  processes a read request by repeatedly querying appropriately chosen neighbors of  $x$  that lie in increasingly larger balls around  $x$ . Similarly, node  $y$  processes an insert request for a given object by repeatedly passing a pointer to the new object copy to appropriately chosen neighbors of  $y$  in increasingly larger balls around  $y$ . As mentioned in Section 3, while the simplified access scheme is relatively easy to analyze, it suffers from the drawback that the memory required at the individual nodes is high.

**Analyzing the drift of balls.** In the foregoing discussion, we argued that if the behavior of the access scheme at each level is close to the expected behavior, then  $C_i \supseteq X_i$  and  $D_i \subseteq Y_i$  hold for all  $i$ . It is quite likely, however, that the ball sizes may deviate from their expected values at some level. An important part of our analysis is to quantify the “drift” that the balls  $C_i$  and  $D_i$  have with respect to the balls  $X_i$  and  $Y_i$ , respectively, and thus bound the increase in the expected cost due to forwarding, a technical challenge referred to as Problem 3 in Section 3. In fact, it helps us in our analysis to quantify the drift of the sets  $A_i$ ,  $B_i$ , and  $E_i$  as well and thus provide a more accurate characterization of our access scheme. The main difficulty is that it is not possible to place useful “high probability” bounds on the amount of drift for each value of  $i$ . We instead obtain an approximate probability distribution on the amount of drift. More precisely, we define a nonnegative state variable  $s_i$  (resp.,  $t_i$ ) for the sequence  $\langle x \rangle$  (resp.,  $\langle y \rangle$ ) that captures the amount of drift in the balls  $A_i$ ,  $B_i$  and  $C_i$  (resp.,  $D_i$  and  $E_i$ ). The larger the values of the state variables  $s_i$  and  $t_i$ , the larger the drift. Thus, for example, when  $s_i$  is 0,  $C_i$  is a superset of  $X_i$  and  $C_i$  is a subset of  $X'_i$ . Similarly, if  $t_i$  is 0, then the set  $D_i$  is a subset of  $Y_i$ . Thus  $\tau$  can be now defined as the smallest  $i \geq i^*$  such that  $s_i = t_i = 0$ .

The sequence  $(s_i, t_i)$  can be naturally viewed as a random walk on the two-dimensional integer lattice. One of our key claims concerns the characterization of the random walk defined by  $(s_i, t_i)$ . We show in Lemmas 8.6 and 8.7 that this random walk is biased towards  $(0, 0)$ . More precisely, we show that if  $s_i$  (resp.,  $t_i$ ) is positive then it decreases with a probability that can be made arbitrarily close to 1 by selecting the constants  $b$  and  $d$  in the algorithm to be sufficiently large. We note that a sufficiently large value for  $b$  ensures that the probability that  $C_i$  is a subset of  $X'_i$  and  $D_i$  is a subset of  $Y_i$  is close to 1 (thus addressing Problem 3 of Section 3). Similarly, a sufficiently large value for  $d$  ensures that the probability that  $C_i$  is a superset of  $X_i$  is close to 1 (thus addressing Problem 2 of Section 3).

**Bounding the expected total cost.** Lemmas 8.6 and 8.7 together characterize a single step of the random walk defined by  $(s_i, t_i)$ . Using the two lemmas and standard combinatorial techniques

for analyzing random walks on a line, we then establish the following claim (Lemma 8.17): For any  $i$ , the probability that  $j$  is the smallest index such that  $(s_{i+j}, t_{i+j})$  is  $(0, 0)$  decreases exponentially with  $j$ . One obvious corollary of the preceding claim is that the expected value of  $\tau$  is  $i^* + O(1)$ . More importantly, however, it allows us to bound the tail of the probability distribution of  $\tau$ . That is, Lemma 8.17 implies that the probability that the read request terminates at level  $i \geq i^*$  decreases exponentially with  $i - i^*$ . On the other hand, the expected cost incurred at a level  $i$  increases exponentially with  $i$ . By choosing the constants involved judiciously, we ensure that the product of the probability of reaching a level  $i > i^*$  to the expected cost incurred at level  $i$  decreases exponentially with  $i - i^*$ , thus yielding the desired upper bound on the expected total cost.

## 6.2 Other performance bounds

We now present an informal overview of the analysis for the other performance bounds, namely, that of the insert and delete operations, the auxiliary memory size, and the adaptability of our access scheme. We first consider the insert and delete operations. When an insert operation is issued at a node  $y$ , the node updates the pointer lists of the nodes in some prefix of the primary neighbor sequence  $\langle y \rangle$ . The desired upper bound on the expected cost of an insert operation thus follows directly from the bounds on the expected cost incurred at each level of the primary neighbor sequence that we establish in the analysis of the read operation. For a delete operation issued for a copy of object  $A$  at node  $y$ , an additional communication cost needs to be taken into account since a node in the primary neighbor sequence  $\langle y \rangle$  may also send a message to its reverse neighbors to check for another copy of  $A$ . We show by standard Chernoff-type arguments that the number of reverse neighbors for any node is  $O(\log n)$  expected and  $O(\log^2 n)$  whp (Corollary 9.2.1), thus leading to the desired bound for the delete operation.

As mentioned in Section 4, the auxiliary memory at any node consists of two parts. The first part, the neighbor table, consists of the primary, secondary, and reverse neighbors. Since the number of primary and secondary neighbors of a node is  $O(\log n)$ , the discussion in the preceding paragraph about the number of reverse neighbors indicates that the size of the neighbor table is  $O(\log n)$  expected and  $O(\log^2 n)$  whp. For bounding the size of the second part, the pointer list, we first observe that if node  $u$  has a pointer to an object  $A$  lying at another node  $v$ , there is an  $i$  such that (i)  $u$  is the primary  $(i, A[i])$ -neighbor of  $v$ , and (ii) the label of node  $u$  matches the ID of  $A$  in the rightmost  $ib$  bits. We show by means of a Chernoff-type argument that the number of nodes  $v$  that satisfy condition (i) is  $O(2^{ib} \log n)$  whp (see Lemma 9.3). Moreover, the probability of condition (ii) occurring is at most  $1/2^{ib}$ . Since the number of objects lying at a node is at most  $q$  and there are  $O(\log n)$  possible values of  $i$ , we obtain that the number of pointers at a node is  $O(q \log^2 n)$  whp. This yields the desired upper bound on the auxiliary memory size.

Finally, since the number of nodes that need to be updated on the removal and the addition of a node  $u$  is equal to the total number of neighbors of  $u$ , the desired bound on the adaptability of the access scheme follows directly from the bound on the size of the neighbor table.

## 7 Preliminaries for the Analysis

In this section, we present some preliminary definitions in Section 7 that are used in our analysis. We first define the relationship between the several constants that appear in the model, the access scheme, and the analysis. The constants  $\delta$  and  $\Delta$  appear in the model,  $b$  and  $d$  appear in the access scheme, and  $\gamma$  and  $\varepsilon$  appear in the analysis. We choose  $b$ ,  $d$ ,  $\gamma$ , and  $\varepsilon$  such that the following inequalities hold.

$$2^b \geq \Delta^2 \gamma^3 \tag{2}$$

$$d \geq \gamma^2 \tag{3}$$



$$\gamma \geq \Delta^2 \quad (4)$$

$$\varepsilon \geq \max\{6\Delta/\gamma, 4e^{-\gamma/4\Delta}, 6(d+1)/2^b, 6(2e/2^b)^{d/2}, 6(e\Delta\gamma^2/d)^{d/2}\} \quad (5)$$

$$\varepsilon < (10 \cdot 2^{\lceil b \log_\delta 2 \rceil})^{-1} \quad (6)$$

An assignment of values to the constants  $\gamma$ ,  $b$ ,  $d$ , and  $\varepsilon$  that satisfies the above inequalities may be obtained as follows: Set  $\gamma$  equal to  $2^{b/3}/\Delta^{2/3}$ ,  $d$  equal to  $e2^{2b/3+1}/\Delta^{1/3}$ , and  $\varepsilon$  equal to  $6e\Delta^{5/3}/2^{b/3}$ . The preceding assignment satisfies Equations 2 and 5 if  $b$  is set sufficiently large. Equations 4 and 6 can be satisfied by setting  $b$  large enough so that  $2^b \geq \Delta^8$  and  $2^{b/3-\lceil b \log_\delta 2 \rceil} > 60e\Delta^{5/3}$ . (Note that since we assume  $\delta > 8$  for our model,  $b/3 - \lceil b \log_\delta 2 \rceil$  can be set to an arbitrarily large constant by choosing  $b$  sufficiently large.)

During the course of our analysis, we frequently study certain “local neighborhoods” of a given node with respect to the cost function  $c$ . In Section 2, we defined the ball of radius  $r$  around a node  $u$ ,  $M(u, r)$ . We now define the *ball of size  $k$*  around node  $u$ ,  $N(u, k)$ , for any  $u$  in  $V$  and any integer  $k$  in  $[1, n]$  as follows.<sup>3</sup> Let  $N(u, k)$  denote the unique set of  $k$  nodes such that for any  $v$  in  $N(u, k)$  and  $w$  not in  $N(u, k)$ ,  $c(u, v)$  is less than  $c(u, w)$ . For convenience, if  $k$  is greater than  $n$ , we let  $N(u, k)$  be  $V$ . As in the case of the balls  $M(u, r)$ , we define the *radius* of  $N(u, k)$  to be the maximum value of  $c(u, v)$  over all  $v$  in  $N(u, k)$ .

## 8 Analysis of the Read Operation

We begin our analysis of the read operation by establishing some useful properties of balls in Section 8.1. The desired upper bound on the expected cost of an individual read operation is shown in Section 8.2.

### 8.1 Properties of Balls

In the proofs of the lemmas in this section, we make extensive use of Equation 1 as well as the fact that the cost function is symmetric and satisfies the triangle inequality. We first consider the smallest (resp., the largest) ball centered at a node  $v$  that contains (resp., is contained in) some given subset of nodes. Given any subset  $S$  of  $V$  and some node  $u$  in  $S$ , let  $n_{\subseteq}(u, S)$  (resp.,  $n_{\supseteq}(u, S)$ ) denote the largest (resp., smallest) integer  $k$  such that  $N(u, k)$  is a subset (resp., superset) of  $S$ . Let  $N_{\subseteq}(u, S)$  and  $N_{\supseteq}(u, S)$  denote  $N(u, n_{\subseteq}(u, S))$  and  $N(u, n_{\supseteq}(u, S))$ , respectively.

**Lemma 8.1** *Let  $u$  belong to  $V$ , and let  $k_0$  and  $k_1$  denote positive integers such that  $k_1 \geq \Delta^2 k_0$ . For any  $v$  in  $N(u, k_0)$ ,  $n_{\subseteq}(v, N(u, k_1))$  is at least  $k_1/\Delta$  and  $N_{\supseteq}(v, N(u, k_1))$  is a subset of  $N(u, \Delta k_1)$ .*

**Proof:** We first obtain a lower bound on  $n_{\subseteq}(v, N(u, k_1))$ . Let  $r_0$  and  $r_1$  denote the radii of  $N(u, k_0)$  and  $N(u, k_1)$ , respectively. Since  $k_1 \geq \Delta^2 k_0$ , Equation 1 implies that  $r_1 - r_0 \geq (r_1 + r_0)/2$ . Let  $w$  be the node in  $N_{\subseteq}(v, N(u, k_1))$  such that  $c(v, w)$  is maximum. A ball of radius  $(r_1 - r_0)$  around  $v$  is contained in  $N(u, k_1)$  (since  $v$  is contained in  $N(u, k_0)$ ). Thus  $r_1 - r_0 \leq c(v, w)$ . It follows that  $2c(v, w)$  is at least  $2(r_1 - r_0) \geq r_1 + r_0$  and  $M(v, 2c(v, w))$  is a superset of  $N(u, k_1)$ . We now obtain a lower bound on  $n_{\subseteq}(v, N(u, k_1))$  as follows:

$$\begin{aligned} n_{\subseteq}(v, N(u, k_1)) &= |M(v, c(v, w))| \\ &\geq |M(v, 2c(v, w))|/\Delta \\ &\geq k_1/\Delta. \end{aligned}$$

We now place an upper bound on  $n_{\supseteq}(v, N(u, k_1))$ . Let  $w$  be the node in  $N_{\supseteq}(v, N(u, k_1))$  such that  $c(v, w)$  is maximum. We have  $r_1 - r_0 \leq c(v, w) \leq r_1 + r_0$ . (We showed that  $r_1 - r_0 \leq c(v, w)$  in

---

<sup>3</sup>For integers  $a$  and  $b$ , we let  $[a, b]$  denote the set  $\{k \in \mathbf{Z} : a \leq k \leq b\}$ .

the preceding paragraph, and the other inequality follows from the triangle inequality.) It follows that  $2(r_1 - r_0)$  is at least  $c(v, w)$  and  $M(v, c(v, w))$  is a subset of  $M(u, 2r_1)$ . Therefore,

$$\begin{aligned} n_{\supseteq}(v, N(u, k_1)) &= |M(v, c(v, w))| \\ &\leq |M(u, 2r_1)| \\ &\leq \Delta k_1. \end{aligned}$$

■

Lemma 8.2 and Corollary 8.2.1 are used in Section 8.2. We refer to any predicate on  $V$  that only depends on the label of  $v$  as a *label predicate*. Given any node  $u$  in  $V$  and a label predicate  $\mathcal{P}$  on  $V$ , let  $p(u, \mathcal{P})$  denote the node  $v$  such that (i)  $\mathcal{P}(v)$  holds, and (ii) for any node  $w$  such that  $\mathcal{P}(w)$  holds,  $c(u, v)$  is at most  $c(u, w)$ . (We let  $p(u, \mathcal{P})$  be null if such a  $v$  is not defined.) Let  $P(u, \mathcal{P})$  be  $M(u, c(u, p(u, \mathcal{P})))$  if  $p(u, \mathcal{P})$  is not null, and  $V$  otherwise.

Lemma 8.2 examines the relationship between the set  $P(u, \mathcal{P})$  and the probability distribution of the labels of the nodes, for any given node  $u$  and label predicate  $\mathcal{P}$ . For  $u$  in  $V$ , and  $i$  in  $[(\log n)/b]$ , let  $\lambda_{\geq i}(u)$  denote the string of  $(\log n - ib)$  bits given by  $u[(\log n)/b - 1] \cdots u[i + 1]u[i]$ . For convenience, we let  $\lambda_{> i}(u)$  denote  $\lambda_{\geq i+1}(u)$ . For all  $i$  and all  $u$  in  $V$ , let  $\mathcal{P}_i(u)$  hold if and only if  $u[i] = A[i]$ . For all  $i$  and all  $u$  in  $V$ , let  $\mathcal{P}_{< i}(u)$  denote  $\bigwedge_{j \in [i]} \mathcal{P}_j(u)$ . Let  $\mathcal{P}_{\leq i}(u)$ ,  $\mathcal{P}_{> i}(u)$ , and  $\mathcal{P}_{\geq i}(u)$  be defined similarly. We note that for  $u$  and  $v$  in  $V$ , and nonnegative integers  $i$  and  $j$ , if  $(u \neq v) \vee ((u = v) \wedge (i \neq j))$ , then  $\mathcal{P}_i(u)$  and  $\mathcal{P}_j(v)$  are independent random variables. Also, each of the predicates defined above is a label predicate.

**Lemma 8.2** *Let  $S$  and  $S'$  be subsets of  $V$ , and let  $u$  belong to  $S$ . Let  $\mathcal{P}$  be a label predicate on  $V$  and for each  $v$  in  $S'$ , let  $\lambda_{\geq 0}(v)$  be chosen independently and uniformly at random.*

1. *Given that  $P(u, \mathcal{P}) \subseteq S$ , we have (i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  in  $S' \setminus P(u, \mathcal{P})$ , are independent and uniformly random, and (ii) for each node  $v$  in  $P(u, \mathcal{P}) \setminus \{p(u, \mathcal{P})\}$ ,  $\mathcal{P}(v)$  is false.*
2. *Given that  $P(u, \mathcal{P}) \not\subseteq S$ , we have (i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  in  $S' \setminus N_{\subseteq}(u, S)$ , are independent and uniformly random, and (ii) for each node  $v$  in  $N_{\subseteq}(u, S)$ ,  $\mathcal{P}(v)$  is false.*
3. *Given that  $P(u, \mathcal{P}) \supseteq S$ , we have (i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  in  $S' \setminus N_{\supseteq}(u, S)$ , are independent and uniformly random, and (ii) for each node  $v$  in  $N_{\supseteq}(u, S) \setminus \{p(u, \mathcal{P})\}$ ,  $\mathcal{P}(v)$  is false.*

**Proof:** We first consider Part 1 of the lemma. Part 1(i) follows from the independence of  $\mathcal{P}(v)$  and  $\mathcal{P}(w)$ , for any two distinct nodes  $v$  and  $w$ . By the definition of  $P$ ,  $\mathcal{P}(p(u, \mathcal{P}))$  holds and for each node  $v$  in  $P(u, \mathcal{P}) \setminus \{p(u, \mathcal{P})\}$ ,  $\mathcal{P}(v)$  is false. This proves Part 1(ii). Parts 2 and 3 follow similarly. For Part 2, we note that the event  $P(u, \mathcal{P}) \not\subseteq S$  is equivalent to the event that for each node  $v$  in  $N_{\subseteq}(u, S)$ ,  $\mathcal{P}(v)$  is false. For Part 3, we note that the event  $P(u, \mathcal{P}) \supseteq S$  is equivalent to the event that for each node in  $N_{\supseteq}(u, S) \setminus \{p(u, \mathcal{P})\}$ ,  $\mathcal{P}(v)$  is false. ■

The following claim follows from repeated application of Part 1 of Lemma 8.2.

**Corollary 8.2.1** *Let  $S$  be an arbitrary subset of  $V$ , let  $i$  be in  $[(\log n)/b - 1]$ , and let  $S'$  be a subset of  $V$  such that  $\lambda_{\geq 0}(u)$  is independently and uniformly random for each  $u$  in  $S'$ . Given a sequence of nodes  $u_0, \dots, u_i$  such that for all  $j$  in  $[i]$ ,  $u_{j+1} = p(u_j, \mathcal{P}_{\leq j})$  and  $P(u_j, \mathcal{P}_{\leq j}) \subseteq S$ , we have*

1. The variables  $\lambda_{\geq 0}(u)$ , for all  $u$  in  $S' \setminus \cup_{j \in [i]} P(u, \mathcal{P}_{\leq j})$ , are independent and uniformly random.
2. The variable  $\lambda_{> i}(u_i)$  is uniformly random and for each node  $u$  in  $\cup_{j \in [i]} P(u_j, \mathcal{P}_{\leq j}) \setminus \{u_i\}$ ,  $\mathcal{P}_{\leq i}(u)$  is false. ■

## 8.2 Cost of the read operation

In this section, we place upper bounds on the cost of the read operation by establishing Theorem 1. We first introduce some notation and prove a few elementary lemmas in Section 8.2.1. The bulk of the analysis appears in Sections 8.2.2 and 8.2.3. Using the tools developed in these two sections, we prove Theorem 1 in Section 8.2.4.

### 8.2.1 Preliminaries

Consider a read request originating at node  $x$  for an object  $A$ . Let  $y$  denote a node that has a copy of  $A$ . In the following, we show that the expected cost of a read operation is  $O(f(\ell(A))c(x, y))$ . If  $y$  denotes the node with minimum  $c(x, y)$  among the set of nodes that have a copy of  $A$ , then the preceding bound implies that the expected cost is asymptotically optimal.

Let  $\langle x \rangle$  and  $\langle y \rangle$  be the primary neighbor sequences for  $A$  with  $x_0 = x$  and  $y_0 = y$ , respectively. For any nonnegative integer  $i$ , let  $A_i$  (resp.,  $D_i$ ) denote the ball of smallest radius around  $x_i$  (resp.,  $y_i$ ) that contains  $x_{i+1}$  (resp.,  $y_{i+1}$ ). Let  $B_i$  (resp.,  $E_i$ ) denote the set  $\cup_{0 \leq j \leq i} A_j$  (resp.,  $\cup_{0 \leq j \leq i} D_j$ ). Let  $C_i$  denote the ball of smallest radius around  $x_i$  that contains all of the secondary  $(i, A[i])$ -neighbors of  $x_i$ . For convenience, we define  $B_{-1} = E_{-1} = \emptyset$ .

It is useful to consider an alternative view of  $x_i$ ,  $y_i$ ,  $A_i$ , and  $D_i$ . For any nonnegative  $i$ , if  $x_{i+1}$  (resp.,  $y_{i+1}$ ) is not the root node for  $A$ , then  $x_{i+1}$  (resp.,  $y_{i+1}$ ) is  $p(x_i, \mathcal{P}_{\leq i})$  (resp.,  $p(y_i, \mathcal{P}_{\leq i})$ ) and  $A_i$  (resp.,  $D_i$ ) is  $P(x_i, \mathcal{P}_{\leq i})$  (resp.,  $P(y_i, \mathcal{P}_{\leq i})$ ).

Let  $\gamma$  be an integer constant satisfying Equations 2 through 6. For any nonnegative integer  $i$  and any integer  $j$ , let  $X_i^j$  (resp.,  $Y_i^j$ ) denote the ball  $N(x, \gamma^j 2^{(i+1)b})$  (resp.,  $N(y, \gamma^j 2^{(i+1)b})$ ). Let  $i^*$  denote the least integer such that the radius of  $X_{i^*}^1$  is at least  $c(x, y)$ . Let  $a_i$  (resp.,  $b_i$ ) denote the radius of  $X_i^1$  (resp.,  $Y_i^1$ ).

**Lemma 8.3** *For all  $i$  such that  $i \geq i^*$ ,  $X_i^2$  is a superset of  $Y_i^1$ .*

**Proof:** By the definition of  $i^*$ ,  $a_i$  is at least  $c(x, y)$ . Therefore,  $M(y, 2a_i)$  is a superset of  $X_i^1$ . Hence,  $M(y, 2a_i)$  contains at least  $\gamma 2^{(i+1)b}$  nodes and is a superset of  $Y_i^1$ .

By Equation 1,  $|M(x, 3a_i)|$  is at most  $\Delta^2 |M(x, a_i)| \leq \Delta^2 |X_i^1|$ . By Equation 4,  $\Delta^2 |X_i^1| \leq \Delta^2 \gamma 2^{(i+1)b} \leq \gamma^2 2^{(i+1)b}$ . Thus,  $M(x, 3a_i)$  is a subset of  $X_i^2$ . Since  $M(x, 3a_i)$  is a superset of  $M(y, 2a_i)$ , which is a superset of  $Y_i^1$ , the claim holds. ■

**Lemma 8.4** *For all  $i$  in  $[(\log n)/b - 2]$  we have  $2^{\lceil b \log_{\Delta} 2 \rceil} a_i \leq a_{i+1} \leq 2^{\lceil b \log_{\delta} 2 \rceil} a_i$  and  $2^{\lceil b \log_{\Delta} 2 \rceil} b_i \leq b_{i+1} \leq 2^{\lceil b \log_{\delta} 2 \rceil} b_i$ . For  $i = (\log n)/b - 2$ , we have  $a_{i+1} \leq 2^{\lceil b \log_{\delta} 2 \rceil} a_i$  and  $b_{i+1} \leq 2^{\lceil b \log_{\delta} 2 \rceil} b_i$ . Also,  $a_{i^*}$  and  $b_{i^*}$  are both  $O(c(x, y))$ .*

**Proof:** We prove the bounds for  $a_{i+1}$  and  $a_{i^*}$ . The bounds for  $b_{i+1}$  and  $b_{i^*}$  follow the same lines. Since  $\gamma \leq 2^b$  by Equation 4, for all  $i$  in  $[(\log n)/b - 2]$ , we have  $|X_{i+1}^1| = 2^b |X_i^1|$ . Therefore, for all  $i$  in  $[(\log n)/b - 2]$ , it follows from Equation 1 that  $2^{\lceil b \log_{\Delta} 2 \rceil} a_i \leq a_{i+1} \leq 2^{\lceil b \log_{\delta} 2 \rceil} a_i$ . For  $i = (\log n)/b - 2$ ,  $|X_{i+1}^1| \leq 2^b |X_i^1|$ , and hence,  $a_{i+1} \leq 2^{\lceil b \log_{\delta} 2 \rceil} a_i$ .

If  $i^* > 0$ , then  $a_{i^*}$  (resp.,  $b_{i^*}$ ) is at most  $2^{\lceil b \log_{\delta} 2 \rceil} c(x, y) = O(c(x, y))$ . Otherwise,  $a_{i^*}$  is  $O(2^{\lceil \log_{\delta} \gamma \rceil}) = O(c(x, y))$ , since  $\delta$  and  $\gamma$  are constants. ■

We define two sequences  $\langle s_i \rangle$  and  $\langle t_i \rangle$  of nonnegative integers as follows:

$$s_i = \begin{cases} 0 & \text{if } B_i \subseteq X_i^1, A_i \supseteq X_i^{-1}, C_i \supseteq X_i^2, \\ 1 & \text{if } B_i \subseteq X_i^1, A_i \supseteq X_i^{-1}, C_i \not\supseteq X_i^2, \\ 2 & \text{if } B_i \subseteq X_i^1, A_i \not\supseteq X_i^{-1}, \text{ and} \\ 3+j & \text{if } 0 \leq j \leq i, B_{i-j} \not\subseteq X_i^1, B_{i-j-1} \subseteq X_i^1. \end{cases} \quad (7)$$

$$t_i = \begin{cases} 0 & \text{if } E_i \subseteq Y_i^1, \text{ and} \\ 1+j & \text{if } 0 \leq j \leq i, E_{i-j} \not\subseteq Y_i^1, E_{i-j-1} \subseteq Y_i^1. \end{cases} \quad (8)$$

The following intuition underlies the above definitions of  $s_i$  and  $t_i$ . For any  $j$ , the expected sizes of the balls  $A_i$  and  $D_i$  are both  $2^{(i+1)b}$ . Thus, the expected sizes of the balls  $B_i$  and  $E_i$  are both at most  $2^{(i+1)b+1}$ . Moreover, the expected size of  $C_i$  is at least  $c_1 2^{(i+1)b}$ , where  $c_1$  is a constant that depends on  $d$ . The constant  $\gamma$  is chosen sufficiently large and  $d$  is chosen sufficiently larger than  $\gamma$  such that the “expected behavior” of the balls  $A_i$ ,  $B_i$ ,  $C_i$ , and  $E_i$  is as follows:  $B_i \subseteq X_i^1$ ,  $A_i \supseteq X_i^{-1}$ ,  $C_i \supseteq X_i^2$ , and  $E_i \subseteq Y_i^1$ . The value of  $s_i$  (resp.,  $t_i$ ) indicates the degree to which the sizes of the balls  $A_i$ ,  $B_i$ , and  $C_i$  (resp.,  $D_i$  and  $E_i$ ) deviate from this expected behavior. The larger the value of  $s_i$ , the greater the deviation from the expected behavior.

**Lemma 8.5** *If  $s_i$  is in  $\{0, 1, 2\}$ , then  $c(x_i, x_{i+1})$  is  $O(a_i)$ . If  $t_i$  is 0, then  $c(y_i, y_{i+1})$  is  $O(b_i)$ .*

**Proof:** The proof of the first claim follows from the observation that if  $s_i$  is in  $\{0, 1, 2\}$  then  $A_i \subseteq B_i \subseteq X_i^1$ . The proof of the second claim follows from the observation that if  $t_i$  is 0 then  $D_i \subseteq E_i \subseteq Y_i^1$ . ■

### 8.2.2 Properties of $\langle s_i \rangle$ and $\langle t_i \rangle$

Our plan for determining an upper bound on the cost of the given read operation for object  $A$  is as follows. Let  $\tau$  be the smallest integer  $i \geq i^*$  such that  $(s_i, t_i) = (0, 0)$ . By the definitions of  $s_\tau$  and  $t_\tau$ ,  $C_\tau \supseteq X_\tau^2$  and  $Y_\tau^1 \supseteq E_\tau \supseteq D_\tau$ . By Lemma 8.3,  $X_\tau^2 \supseteq Y_\tau^1$ , thus implying that  $C_\tau$  is a superset of  $D_\tau$ . Thus, a copy of  $A$  is located within  $\tau$  forwarding steps along  $\langle x \rangle$ . By the definition of the primary and secondary neighbors, the total cost of all messages sent by node  $x_i$  is  $O(c(x_i, x_{i+1}))$ . Since a copy of  $A$  is located within  $\tau$  forwarding steps, the cost of all messages needed in locating the particular copy of  $A$  that is read is  $O(\sum_{0 \leq j < \tau} (dc(x_j, x_{j+1}) + c(y_j, y_{j+1})))$ . The cost of reading the copy is at most  $f(\ell(A))$  times the preceding cost. Since  $d$  is a constant, the cost of reading  $A$  is at most

$$\sum_{0 \leq j < \tau} O(f(\ell(A))(c(x_j, x_{j+1}) + c(y_j, y_{j+1}))). \quad (9)$$

The remainder of the proof concerns the task of showing that  $E[\sum_{0 \leq j < \tau} (c(x_j, x_{j+1}) + c(y_j, y_{j+1}))]$  is  $O(c(x, y))$ . A key idea is to establish that the sequence  $\langle s_i, t_i \rangle$  corresponds to a two-dimensional random walk that is biased towards  $(0, 0)$ . Lemmas 8.6 and 8.7 below provide a first step towards formalizing this notion.

**Lemma 8.6** *Let  $i$  be in  $[(\log n)/b - 1]$ . Given  $s_j$  and  $t_j$  for all  $j$  in  $[i]$  such that  $s_{i-1}$  is at least 3, the probability that  $s_i$  is less than  $s_{i-1}$  is at least  $1 - \varepsilon^2$ . Given  $s_j$  and  $t_j$  for all  $j$  in  $[i]$  such that  $t_{i-1}$  is at least 1, the probability that  $t_i$  is less than  $t_{i-1}$  is at least  $1 - \varepsilon^2$ .*

**Lemma 8.7** *Let  $i$  be in  $[(\log n)/b - 1]$ . Given  $s_j$  and  $t_j$  for all  $j$  in  $[i]$  such that  $s_{i-1}$  is at most 3, the probability that  $s_i$  is 0 is at least  $1 - \varepsilon$ . Given  $s_j$  and  $t_j$  for all  $j$  in  $[i]$  such that  $t_{i-1}$  is at most 1, the probability that  $t_i$  is 0 is at least  $1 - \varepsilon$ .*

In order to establish the above lemmas, we first introduce some additional notation. For each  $i \geq -1$ , we define  $S_i$  and  $T_i$  as follows. Let  $S_{-1} = T_{-1} = \emptyset$  and for  $i \geq 0$  let

$$S_i = \begin{cases} S_{i-1} \cup B_i \cup (C_i \cap N_{\supseteq}(x_i, X_i^2)) & \text{if } s_i \in \{0, 1\}, \\ S_{i-1} \cup B_i & \text{if } s_i = 2, \\ S_{i-1} \cup B_{i-s_i+2} \cup N_{\subseteq}(x_{i-s_i+3}, X_i^1) & \text{otherwise.} \end{cases}$$

$$T_i = \begin{cases} T_{i-1} \cup E_i & \text{if } t_i = 0, \\ T_{i-1} \cup E_{i-t_i} \cup N_{\subseteq}(y_{i-t_i+1}, Y_i^1) & \text{otherwise.} \end{cases}$$

The set  $S_i$  (resp.,  $T_i$ ) contains all of the nodes whose labels need to be examined to determine the values of  $s_0$  through  $s_i$  (resp.,  $t_0$  through  $t_i$ ). Moreover, as we show in Lemma 8.8, the particular values of  $s_0$  through  $s_i$  and  $t_0$  through  $t_i$  bias the distribution of only a suffix of the labels of the nodes in  $S_i \cup T_i$ .

**Lemma 8.8** *Let  $i$  be in  $[(\log n)/b - 1]$ . Given  $s_j$  and  $t_j$  for all  $j$  in  $[i]$ , we have that*

1. *The variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_i \cup T_i$ , are independent and uniformly random.*
2. *There exists a subset  $S'_i$  of  $S_i$  of size at most  $d + 1$  such that (i) the variables  $\lambda_{> i}(u)$ , for all  $u$  in  $S'_i$ , are independent and uniformly random, and (ii) for each node  $u$  in  $S_i \setminus S'_i$ ,  $\mathcal{P}_{\leq i}(u)$  is false.*
3. *There exists at most one node  $v$  in  $T_i$  such that (i) the variable  $\lambda_{> i}(v)$  is uniformly random, and (ii) for each node  $u$  in  $T_i \setminus \{v\}$ ,  $\mathcal{P}_{< i}(u)$  is false.*

**Proof:** We prove Parts 1, 2, and 3 for all  $i \geq -1$ . The proof is by induction. For the induction base we set  $i = -1$ . Part 1 follows directly from the random assignment of labels. For Part 2, we set  $S'_{-1}$  to  $\emptyset$ , and the desired claim holds since  $S_{-1}$  is  $\emptyset$ . The claim of Part 3 holds vacuously since  $T_{-1}$  is  $\emptyset$ .

For the induction hypothesis, we assume that Parts 1, 2, and 3 of the lemma hold for  $i - 1$ . We first consider different cases depending on the value of  $s_i$ .

- (a)  $s_i = 3 + j, j \in [i]$ : The event  $s_i = 3 + j$  is equivalent to the event  $(B_{i-j-1} \subseteq X_i^1) \wedge (A_{i-j} \not\subseteq X_i^1)$ . We first condition on the event  $B_{i-j-1} \subseteq X_i^1$  by invoking Corollary 8.2.1 with the substitution  $(X_i^1, V \setminus (S_{i-1} \cup T_{i-1}), i - j)$  for  $(S, S', i)$ . We next condition on the event  $A_{i-j} \not\subseteq X_i^1$  by invoking Part 2 of Lemma 8.2 with the substitution  $(x_{i-j}, X_i^1, V \setminus (S_{i-1} \cup T_{i-1} \cup B_{i-j-1}), \mathcal{P}_{\leq i})$  for  $(u, S, S', \mathcal{P})$ . By combining Part (i) of both invocations, we have (a.i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  not in  $S_{i-1} \cup T_{i-1} \cup B_{i-j-1} \cup N_{\subseteq}(x_{i-j}, X_i^1)$ , are independent and uniformly random. By combining Part (ii) of both invocations, we have (a.ii) for each node  $v$  in  $B_{i-j-1} \cup N_{\subseteq}(x_{i-j}, X_i^1)$ ,  $\mathcal{P}_{\leq i}(v)$  is false.

We set  $S'_i$  to  $S'_{i-1} \setminus (B_{i-j-1} \cup N_{\subseteq}(x_{i-j}, X_i^1))$ .

- (b)  $s_i = 2$ : The event  $s_i = 2$  is equivalent to the event  $(B_i \subseteq X_i^1) \wedge (A_i \not\subseteq X_i^1)$ . We first condition on the event  $B_i \subseteq X_i^1$  by invoking Corollary 8.2.1 with the substitution  $(X_i^1, V \setminus (S_{i-1} \cup T_{i-1}), i)$  for  $(S, S', i)$ . It follows from the preceding invocation and the definition of  $B_i$  that (b.i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  not in  $S_{i-1} \cup T_{i-1} \cup B_i$ , are independent and uniformly random, and (b.ii) for each node  $v$  in  $B_i \setminus \{x_{i+1}\}$ ,  $\mathcal{P}_{\leq i}(v)$  is false.

We set  $S'_i$  to  $S'_{i-1} \setminus (B_i \setminus \{x_{i+1}\})$ .

- (c)  $s_i \in \{0, 1\}$ : The event  $s_i \in \{0, 1\}$  is equivalent to the event  $(B_i \subseteq X_i^1) \wedge (A_i \supseteq X_i^{-1})$ . We condition on the event  $B_i \subseteq X_i^1$  by invoking Corollary 8.2 with the substitution  $(X_i^1, V \setminus (S_{i-1} \cup T_{i-1}), i)$  for  $(S, S', i)$ . It follows from the preceding invocation and the definition of  $B_i$  that (i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  not in  $S_{i-1} \cup T_{i-1} \cup B_i$ , are independent and uniformly random, and (ii) for each node  $v$  in  $B_i \setminus \{x_{i+1}\}$ ,  $\mathcal{P}_{\leq i}(v)$  is false.

Let  $S'_i$  equal the set  $\{v \in C_i \cap N_{\supseteq}(x_i, X_i^2) : \mathcal{P}_{\leq i}(v)\}$ . By the definition of  $C_i$ ,  $|S'_i|$  is at most  $d+1$ . If  $C_i \not\supseteq X_i^2$ , then  $C_i \subseteq N_{\supseteq}(x_i, X_i^2)$  and it follows from the definition of  $C_i$  that (c.i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  not in  $S_{i-1} \cup T_{i-1} \cup B_i \cup C_i$ , are independent and uniformly random, and (c.ii) the variables  $\lambda_{> i}(v)$ , for all  $v$  in  $S'_i$ , are independent and uniformly random, and for each node  $v$  in  $(B_i \cup C_i) \setminus S'_i$ ,  $\mathcal{P}_{\leq i}(v)$  is false. If  $C_i \supseteq X_i^2$  then  $C_i \supseteq N_{\supseteq}(x_i, X_i^2)$  and it follows from Part 3 of Lemma 8.2 that (c.i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  not in  $S_{i-1} \cup T_{i-1} \cup B_i \cup N_{\supseteq}(x_i, X_i^2)$ , are independent and uniformly random, and (c.ii) the variables  $\lambda_{> i}(v)$ , for all  $v$  in  $S'_i$ , are independent and uniformly random, and for each node  $v$  in  $(B_i \cup N_{\supseteq}(x_i, X_i^2)) \setminus S'_i$ ,  $\mathcal{P}_{\leq i}(v)$  is false.

We thus obtain from (a.i), (b.i), and (c.i) and the definition of  $S_i$  that (i) the variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_i \cup T_{i-1}$ , are independent and uniformly random. By the definitions of  $s_i$  and  $t_i$ , the particular values of  $s_i$  and  $t_i$  are independent of the suffix  $\lambda_{> i}(u)$  of any node  $u$ . In particular, the variables  $\lambda_{> i}(u)$ , for all  $u$  in  $S'_i$ , are independent and uniformly random. It follows from the preceding observation and claims (a.ii), (b.ii), and (c.ii) that (ii) the bits of  $\lambda_{> i}(u)$ , for all  $u$  in  $S'_i$ , are independent and uniformly random, and for each node in  $S_i \setminus S'_i$ ,  $\mathcal{P}_{\leq i}(u)$  is false. We next consider two cases depending on the value of  $t_i$ .

- (d)  $t_i = 1 + j$ ,  $j \in [i]$ : This case is similar to Case (a). The event  $t_i = 1 + j$  is equivalent to the event  $(E_{i-j-1} \subseteq Y_i^1) \wedge (D_{i-j} \not\subseteq Y_i^1)$ . We first condition on the event  $E_{i-j-1} \subseteq Y_i^1$  by invoking Corollary 8.2.1 with the substitution  $(Y_i^1, V \setminus (S_i \cup T_{i-1}), i - j)$  for  $(S, S', i)$ . We next condition on the event  $D_{i-j} \not\subseteq Y_i^1$  by invoking Part 2 of Lemma 8.2 with the substitution  $(y_{i-j}, Y_i^1, V \setminus (S_i \cup T_{i-1} \cup E_{i-j-1}), \mathcal{P}_{\leq i})$  for  $(u, S, S', \mathcal{P})$ .

By combining Part (i) of both invocations, we have (d.i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  not in  $S_i \cup T_{i-1} \cup E_{i-j-1} \cup N_{\subseteq}(y_{i-j}, Y_i^1)$ , are independent and uniformly random. By combining Part (ii) of both invocations, we have (d.ii) for each node  $v$  in  $E_{i-j-1} \cup N_{\subseteq}(y_{i-j}, Y_i^1)$ ,  $\mathcal{P}_{\leq i}(v)$  is false.

- (e)  $t_i = 0$ : This case is similar to Case (b). The event  $t_i = 2$  is equivalent to the event  $E_i \subseteq Y_i^1$ . We invoke Corollary 8.2.1 with the substitution  $(Y_i^1, V \setminus (S_i \cup T_{i-1}), i)$  for  $(S, S', i)$  to obtain: (e.i) the variables  $\lambda_{\geq 0}(v)$ , for all  $v$  not in  $S_i \cup T_{i-1} \cup E_i$ , are independent and uniformly random, and (e.ii) for each node  $v$  in  $E_i \setminus \{y_{i+1}\}$ ,  $\mathcal{P}_{\leq i}(v)$  is false.

To complete the induction step, we consider each part of the statement of the lemma separately:

1. By (i), (d.i), (e.i), and the definition of  $T_i$ , it follows that given  $s_j$  and  $t_j$ ,  $j \in [i]$ , the variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_i \cup T_i$ , are independent and uniformly random.
2. This part follows directly from (ii) above.
3. By (d.ii) and (e.ii), it follows that given arbitrary values for  $s_j$  and  $t_j$ ,  $j \in [i]$ : (i) the variable  $\lambda_{> i}(y_{i+1})$  is uniformly random, and (ii) for each node  $u$  in  $T_i \setminus \{y_{i+1}\}$ ,  $\mathcal{P}_{\leq i}(u)$  is false.

■

Lemma 8.9 places upper bounds on the sizes of  $S_i$  and  $T_i$ .

**Lemma 8.9** *Let  $i$  be a nonnegative integer. If  $s_i$  is in  $\{0, 1\}$ ,  $S_i$  is a subset of  $X_i^3$ ; otherwise,  $S_i$  is a subset of  $X_i^1$ . The set  $T_i$  is a subset of  $Y_i^1$ .*

**Proof:** The proof is by induction on  $i$ . For convenience, we set  $i = -1$  for the induction base. Since  $S_{-1} = T_{-1} = \emptyset$ , the claims follow trivially. Let the claims of the lemma hold for  $S_{i-1}$  and  $T_{i-1}$ . We will show that  $S_i \subseteq X_i^1$ . The proof for  $T_i$  is along the same lines.

By the induction hypothesis,  $S_{i-1} \subseteq X_{i-1}^3$ . Since  $\gamma^2 \leq 2^b$  by Equation 2,  $X_{i-1}^3 \subseteq X_i^1$ , hence implying that  $S_{i-1} \subseteq X_i^1$ . We now consider three cases depending on the value of  $s_i$ .

If  $s_i$  is in  $\{0, 1\}$ , then  $B_i \subseteq X_i^1$ . Moreover, by Lemma 8.1,  $N_{\supseteq}(x_i, X_i^2) \subseteq N(x, \Delta \gamma^2 2^{(i+1)b})$ . Since  $\Delta \leq \gamma$  by Equation 4,  $N(x, \Delta \gamma^2 2^{(i+1)b}) \subseteq X_i^3$ . It thus follows that  $S_{i-1}$ ,  $B_i$ , and  $N_{\supseteq}(x_i, X_i^2)$  are all subsets of  $X_i^3$ . Thus,  $S_i$  is a subset of  $X_i^3$ .

If  $s_i$  is 2, then  $B_i \subseteq X_i^1$ . It thus follows that  $S_{i-1}$  and  $B_i$  are both subsets of  $X_i^3$ . Thus,  $S_i$  is a subset of  $X_i^3$ .

If  $s_i$  is greater than 2, then  $B_{i-s_i+2} \subseteq X_i^1$ . Moreover,  $N_{\subseteq}(x_{i-s_i+3}, X_i^1)$  is a subset of  $X_i^1$ . Thus,  $S_i$  is a subset of  $X_i^1$ . ■

The following lemma is used in the proof of Lemma 8.6.

**Lemma 8.10** *Let  $i$  be in  $[(\log n/b) - 1]$ . Given  $s_k$  and  $t_k$  for all  $k$  in  $[i]$  such that  $s_{i-1}$  is  $3 + j$  for some  $j$  in  $[i + 1]$ , the probability that  $B_{i-j-1}$  is a subset of  $X_{i-1}^2$  is at least  $1 - \varepsilon^2/2$ . Given  $s_k$  and  $t_k$  for all  $k$  in  $[i]$  such that  $t_{i-1}$  is  $1 + j$  for some  $j$  in  $[i + 1]$ , the probability that  $E_{i-j-1}$  is a subset of  $Y_{i-1}^2$  is at least  $1 - \varepsilon^2/2$ .*

**Proof:** Let  $\mathcal{E}$  denote the event that the  $2i$  random variables  $s_k$  and  $t_k$ ,  $k \in [i]$ , take the given values. Let us assume that  $\mathcal{E}$  holds. We begin with the proof of the first claim. Since  $s_{i-1}$  is  $3 + j$ ,  $B_{i-j-1}$  is not a subset of  $X_{i-1}^1$ , and  $B_{i-j-2}$  is a subset of  $X_{i-1}^1$ .

By Part 1 of Lemma 8.8, it follows that given  $\mathcal{E}$ , the variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_{i-1} \cup T_{i-1}$ , are independent and uniformly random. By Lemma 8.9,  $|S_{i-1} \cup T_{i-1}|$  is at most  $\gamma 2^{ib+1}$ . By Lemma 8.1, since  $\gamma \geq \Delta^2$  by Equation 4,  $n_{\subseteq}(x_{i-j-1}, X_{i-1}^2)$  is at least  $\gamma^2 2^{ib}/\Delta$ . Therefore, the probability that  $A_{i-j-1}$  is not a subset of  $N_{\subseteq}(x_{i-j-1}, X_{i-1}^2)$  is at most

$$\begin{aligned} (1 - 1/2^{(i-j)b})(\gamma^2/\Delta - 2\gamma)2^{ib} &\leq e^{-(\gamma^2/\Delta - 2\gamma)2^{ib}} \\ &\leq \varepsilon^2/2. \end{aligned}$$

(The second step makes use of the following inequalities: (i)  $\gamma \geq 4\Delta$ , which is obtained from Equation 4, and (ii)  $e^{-\gamma/\Delta} \leq (4e^{-\gamma/4\Delta})^2/2 \leq \varepsilon^2/2$ , which is obtained from Equation 5.) Since  $N_{\subseteq}(x_{i-j-1}, X_{i-1}^2)$  is a subset of  $X_{i-1}^2$ , the probability that  $A_{i-j-1}$  is not a subset of  $X_{i-1}^2$  is at least  $1 - \varepsilon^2/2$ . Since  $B_{i-j-2}$  is a subset of  $X_{i-1}^1 \subseteq X_{i-1}^2$  and  $B_{i-j-1} = B_{i-j-2} \cup A_{i-j-1}$ , we obtain that  $B_{i-j-1}$  is a subset of  $X_{i-1}^2$  with probability at least  $1 - \varepsilon^2/2$ .

The proof of the second claim is analogous to the above proof and is obtained by substituting  $(t, D, E, y, Y)$  for  $(s, A, B, x, X)$ . ■

We are now ready to prove Lemmas 8.6 and 8.7.

**Proof of Lemma 8.6:** Let  $\mathcal{E}$  denote the event that the  $2i$  random variables  $s_j$  and  $t_j$ ,  $j \in [i]$ , take the given values. Let us assume that  $\mathcal{E}$  holds. We begin with the proof of the first claim. Let  $s_{i-1}$  be  $3 + j$  for some  $j$  in  $[i]$ . Thus,  $B_{i-j-1}$  is not a subset of  $X_{i-1}^1$ , and  $B_{i-j-2}$  is a subset of  $X_{i-1}^1$ .

We show that  $B_{i-j}$  is a subset of  $X_i^1$  with probability at least  $1 - \varepsilon^2$ . We first invoke Lemma 8.10 to obtain: (a)  $B_{i-j-1}$  is a subset of  $X_{i-1}^2$  with probability at least  $1 - \varepsilon^2/2$ . Let us now assume that  $\mathcal{E}$  and the event that  $B_{i-j-1}$  is a subset of  $X_{i-1}^2$  hold.

We now show that (b) the probability that  $B_{i-j}$  is a subset of  $X_i^1$  is at least  $1 - \varepsilon^2/2$ . It follows from Part 1 of Lemma 8.8 that given  $\mathcal{E}$  the variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_{i-1} \cup T_{i-1}$ , are independent and uniformly random. Thus, given  $\mathcal{E}$  and the event that  $B_{i-j-1}$  is a subset of  $X_{i-1}^2$ , the variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_{i-1} \cup T_{i-1} \cup X_{i-1}^2$ , are independent and uniformly random. By Lemma 8.9,  $S_{i-1} \subseteq X_{i-1}^1$  and  $T_{i-1} \subseteq Y_{i-1}^1$ . Therefore, the size of the set  $S_{i-1} \cup T_{i-1} \cup X_{i-1}^2$  is at most  $\gamma(\gamma+1)2^{ib}$ . By Lemma 8.1, since  $\gamma \geq \Delta^2$  by Equation 4,  $n_{\subseteq}(x_{i-j}, X_i^1)$  is at least  $\gamma 2^{(i+1)b}/\Delta$ . Therefore, the probability that  $A_{i-j}$  is not a subset of  $N_{\subseteq}(x_{i-j}, X_i^2)$  is at most

$$\begin{aligned} (1 - 1/2^{(i-j+1)b})^{(\gamma/\Delta - \gamma^2/2^b - \gamma/2^b)2^{(i+1)b}} &\leq e^{-(\gamma/\Delta - \gamma^2/2^b - \gamma/2^b)2^{jb}} \\ &\leq \varepsilon^2/2. \end{aligned}$$

(The second step makes use of the following inequalities: (i)  $2\Delta(\gamma+1) \leq \Delta^2\gamma^3 \leq 2^b$ , which is obtained from Equation 2, and (ii)  $e^{-\gamma/2\Delta} \leq (4e^{-\gamma/4\Delta})^2/2 \leq \varepsilon^2/2$ , which is obtained from Equation 5.) Thus, the probability that  $B_{i-j}$  is not a subset of  $X_i^1$  is at most  $\varepsilon^2/2$ .

It follows from (a) and (b) above that with probability at least  $(1 - \varepsilon^2)$ ,  $s_i$  is less than  $s_{i-1}$ , thus establishing the first claim of the lemma. The proof of the second claim is analogous to the above proof and is obtained by substituting  $(t, D, E, y, Y)$  for  $(s, A, B, x, X)$ .  $\blacksquare$

**Proof of Lemma 8.7:** Let  $\mathcal{E}$  denote the event that the random variables  $s_j, t_j, j \in [i]$ , take the given values. Let us assume that  $\mathcal{E}$  holds. We begin with the proof of the first claim. If  $s_{i-1}$  is in  $\{0, 1, 2\}$ ,  $B_{i-1}$  is a subset of  $X_{i-1}^1$ . If  $s_{i-1}$  is 3, then by Lemma 8.10,  $B_{i-1}$  is a subset of  $X_{i-1}^2$  with probability at least  $1 - \varepsilon^2/2$ . We now assume that  $B_{i-1}$  is a subset of  $X_{i-1}^2$ .

We first show that (a) the probability that  $B_i$  is a subset of  $X_i^1$  is at least  $1 - \varepsilon/3 + \varepsilon^2/2$ . By Part 1 of Lemma 8.8, it follows that given  $\mathcal{E}$  the variables  $\lambda_{\geq 0}(u)$ , for all nodes  $u$  not in  $S_{i-1} \cup T_{i-1}$ , are independent and uniformly random. By Lemma 8.9,  $|S_{i-1} \cup T_{i-1}|$  is at most  $\gamma^3 2^{ib+1}$ . By Lemma 8.1, since  $x_i$  is in  $X_{i-1}^2$  and  $2^b \geq \Delta^2\gamma$  (by Equation 2),  $n_{\subseteq}(x_i, X_i^1)$  is at least  $\gamma 2^{(i+1)b}/\Delta$ . Therefore, the probability that  $A_i$  is not a subset of  $N_{\subseteq}(x_i, X_i^1)$  is at most

$$\begin{aligned} (1 - 1/2^{(i+1)b})^{2^{(i+1)b}(\gamma/\Delta - 2\gamma^3/2^b)} &\leq e^{-(\gamma/\Delta - 2\gamma^3/2^b)2^{ib}} \\ &\leq e^{-\gamma/2\Delta} \\ &\leq \varepsilon/3 - \varepsilon^2/2. \end{aligned}$$

(The second inequality follows from the inequality  $4\gamma^2\Delta \leq 2^b$ , which is obtained from Equation 2. The last inequality follows from the inequalities: (i)  $e^{-\gamma/2\Delta} \leq \varepsilon/4$ , which is obtained from Equation 5, and (ii)  $\varepsilon/4 \leq \varepsilon/3 - \varepsilon^2/2$ , which holds since  $\varepsilon \leq 1/10$  by Equation 6.) This implies that the probability that  $A_i$  is not a subset of  $X_i^1$  is at most  $\varepsilon/3 - \varepsilon^2/2$ .

We next show that (b) the probability that  $A_i$  is a superset of  $X_i^{-1}$  is at least  $1 - \varepsilon/3$ . Since  $\Delta^2\gamma^3 \leq 2^b$  by Equation 2, Lemma 8.1 implies that  $n_{\supseteq}(x_i, X_i^{-1}) \leq \Delta 2^{(i+1)b}/\gamma$ . By Lemma 8.8, we have (i) the variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_{i-1} \cup T_{i-1}$ , are independent and uniformly random, and (ii) there are at most  $d+1$  nodes in  $S_{i-1} \cup T_{i-1}$  for which the predicate  $\mathcal{P}_{< i}$  holds. Therefore, the probability that  $A_i$  is a subset of  $N_{\supseteq}(x_i, X_i^{-1})$  is at most  $(d+1)/2^b + \Delta/\gamma$ , which is at most  $\varepsilon/3$ . It follows that the probability that  $A_i$  is not a superset of  $X_i^{-1}$  is at most  $\varepsilon/3$ .



We finally show that (c) given that  $B_i$  is a subset of  $X_i^1$  and  $A_i$  is a superset of  $X_i^{-1}$ , the probability that  $C_i$  is a superset of  $X_i^2$  is at least  $1 - \varepsilon/3$ . We note that given  $\mathcal{E}$  and the two events that  $B_i$  is a subset of  $X_i^1$  and  $A_i$  is a superset of  $X_i^{-1}$ : (i) the variables  $\lambda_{\geq 0}(u)$ , for all  $u$  not in  $S_{i-1} \cup T_{i-1} \cup X_i^1$ , are independent and uniformly random, and (ii) there exist at most  $d + 1$  nodes in  $S_{i-1} \cup T_{i-1}$  for which the predicate  $\mathcal{P}_{< i}$  holds.

We will place an upper bound on the probability that  $C_i$  is not a superset of  $X_i^2$  by placing a lower bound on the probability that  $C_i$  is not a superset of  $N_{\supseteq}(x_i, X_i^2)$ , which is a superset of  $X_i^2$ . Let  $r_0$  (resp.,  $r_1$ ) denote  $n_{\supseteq}(x_i, X_i^{-1})$  (resp.,  $n_{\supseteq}(x_i, X_i^2)$ ). By definition,  $n_{\supseteq}(x_i, X_i^{-1})$  is at least  $2^{(i+1)b}/\gamma$ . By Lemma 8.1,  $n_{\supseteq}(x_i, X_i^2)$  is at most  $\Delta\gamma^2 2^{(i+1)b}$ .

We first show that the nodes in  $N_{\supseteq}(x_i, X_i^2)$  are within a cost of  $d \cdot c(x_i, x_{i+1})$ . We note that  $c(x_i, x_{i+1})$  is at least the difference of the radii of  $X_i^{-1}$  and  $X_{i-1}^2$ . Moreover, since  $N_{\supseteq}(x_i, X_i^2)$  is a subset of  $X_i^3$ ,  $n_{\supseteq}(x_i, X_i^2)$  is at most the sum of the radii of  $X_i^3$  and  $X_{i-1}^2$ . Since  $(4\gamma^4)^{\log_2 2} \leq \gamma^2 \leq d$  by Equation 3, all of the nodes in  $N_{\supseteq}(x_i, X_i^2)$  are within a cost of  $d \cdot c(x_i, x_{i+1})$  from  $x_i$ .

It now follows that the probability  $C_i$  is not a superset of  $X_i^2$  is at most the probability that there exist  $d$  nodes in  $N_{\supseteq}(x_i, X_i^2)$  whose  $(i+1)b$  rightmost bits match a certain bit-sequence. This probability is at most

$$\begin{aligned} \binom{d+1}{d/2} (1/2^b)^{d/2} + \binom{\Delta\gamma^2 2^{(i+1)b}}{d/2} (1/2^{(i+1)b})^{d/2} &\leq (2e/2^b)^{d/2} + (e\Delta\gamma^2/d)^{d/2} \\ &\leq \varepsilon/3. \end{aligned}$$

(The second step follows from the inequalities  $(2e/2^b)^{d/2} \leq \varepsilon/6$  and  $(e\Delta\gamma^2/d)^d \leq \varepsilon/6$ , both of which are derived from Equation 5.)

It follows from (a), (b), and (c) above that with probability at least  $1 - \varepsilon$ ,  $s_i$  is 0, thus establishing the first claim of the lemma. The proof of the second claim is analogous to the proof of (a) and is obtained by substituting  $(t, D, E, y, Y)$  for  $(s, A, B, x, X)$ .  $\blacksquare$

By the definitions of  $s_i$  and  $t_i$ , it follows that  $0 \leq s_{i+1} \leq 3$  if  $s_i \leq 2$ , and  $0 \leq s_{i+1} \leq s_i + 1$  otherwise. In addition,  $0 \leq t_{i+1} \leq t_i + 1$ , for all  $i$ . Let  $s'_i$  equal 0 if  $s_i = 0$ , 1 if  $s_i \in \{1, 2, 3\}$ , and  $s_i - 2$  otherwise. Hence  $0 \leq \max\{s'_{i+1}, t_{i+1}\} \leq \max\{s'_i, t_i\} + 1$ , for all  $i$ . In Section 8.2.3 below, we analyze the random walk corresponding to the sequence  $\langle \max\{s', t\} \rangle$ .

### 8.2.3 Random walks

We begin the analysis of the random walk corresponding to the sequence  $\langle \max\{s', t\} \rangle$  by proving several useful properties of certain random walks on a line. These properties are stated in Lemmas 8.11 through 8.15. The main technical claim of this section is Lemma 8.17.

Let  $W(U, F)$  be a directed graph in which  $U$  is the set of nodes and  $F$  is the set of edges. For all  $u$  in  $U$ , let  $\mathcal{D}_u$  be a probability distribution over the set  $\{(u, v) \in F\}$ . It is convenient to define  $\Pr_{\mathcal{D}_u}[(u, v) : (u, v) \notin F] = 0$ . A *random walk* on  $W$  starting at  $v_0$  and according to  $\{\mathcal{D}_u : u \in U\}$  is a random sequence  $\langle v \rangle$  such that (i)  $v_i$  is in  $U$  and  $(v_i, v_{i+1})$  is in  $F$ , for all  $i$ , and (ii) given any fixed (not necessarily simple) path  $u_0, \dots, u_i$  in  $W$  and any fixed  $u_{i+1}$  in  $U$ ,  $\Pr[v_{i+1} = u_{i+1} \mid (v_0, \dots, v_i) = (u_0, \dots, u_i)] = \Pr[v_{i+1} = u_{i+1} \mid v_i = u_i] = \Pr_{\mathcal{D}_{u_i}}[(u_i, u_{i+1})]$ .

Let  $H$  be the directed graph with node set  $\mathbf{N}$  (the set of nonnegative integers) and edge set  $\{(i, j) : i \in \mathbf{N}, 0 \leq j \leq i + 1\}$ . Let  $H'$  be the subgraph of  $H$  induced by the edges  $\{(i + 1, i), (i, i + 1) : i \in \mathbf{N}\} \cup \{(0, 0), (1, 1)\}$ .

Let  $p$  and  $q$  be reals in  $(0, 1]$  such that  $p \geq q$ . We now define two random walks,  $\omega_{p,q}$  and  $\omega'_{p,q}$ , on graphs  $H$  and  $H'$ , respectively. Both  $\omega_{p,q}$  and  $\omega'_{p,q}$  begin at the node 0. The walk

$\omega_{p,q} = \langle w \rangle$  is characterized by: (i)  $\Pr[w_{i+1} \leq j-1 \mid w_i = j] \geq p$ , for any integer  $j > 1$ , (ii)  $\Pr[w_{i+1} = 0 \mid w_i = j] \geq q$ , for  $j$  equal 0 or 1, and (iii)  $\Pr[w_{i+1} = 2 \mid w_i = 1] \leq 1-p$ . The walk  $\omega'_{p,q} = \langle w' \rangle$  is characterized by: (i)  $\Pr[w'_{i+1} = j-1 \mid w'_i = j] = p$ , for all integer  $j > 1$ , (ii)  $\Pr[w'_{i+1} = 0 \mid w'_i = j] = q$ , for  $j$  equal 0 or 1, and (iii)  $\Pr[w'_{i+1} = 2 \mid w'_i = 1] = 1-p$ . We note that Lemmas 8.6 and 8.7 imply that the sequence  $\langle \max\{s', t\} \rangle$  can be represented by the random walk  $\omega_{p,q}$  with  $p = 1 - 2\varepsilon^2$  and  $q = 1 - 2\varepsilon$ .

We analyze the random walk  $\omega_{p,q}$  by first showing that  $\omega_{p,q}$  is more “favorable” than  $\omega'_{p,q}$  with respect to the properties of interest. The random walk  $\omega'_{p,q}$  is easier to analyze as it is exactly characterized by  $p$  and  $q$ . Lemmas 8.11 and 8.13 show that the bias of  $\omega_{p,q}$  towards 0 is more than that of  $\omega'_{p,q}$ . Since the values of  $p$  and  $q$  are fixed throughout the following discussion, we omit the subscript  $p, q$  in the terms  $\omega_{p,q}$  and  $\omega'_{p,q}$  for convenience.

**Lemma 8.11** *For all  $i$  and  $k$  in  $\mathbb{N}$ , for random walks  $\omega$  and  $\omega'$ , we have  $\Pr[w_i \leq k] \geq \Pr[w'_i \leq k]$ .*

**Proof:** We prove the claim by induction on  $i$ . The base case  $i = 0$  is trivial since  $\omega_0 = \omega'_0 = 0$ . Assume the claim holds for  $i$  and any  $k$ . If  $k \geq 1$ , then we have

$$\begin{aligned} \Pr[w'_{i+1} \leq k] &= \Pr[w'_i \leq k-1] + p \Pr[k \leq w'_i \leq k+1] \\ &= (1-p) \Pr[w'_i \leq k-1] + p \Pr[w'_i \leq k+1] \end{aligned}$$

and

$$\begin{aligned} \Pr[w_{i+1} \leq k] &\geq \Pr[w_i \leq k-1] + p \Pr[k \leq w_i \leq k+1] \\ &= (1-p) \Pr[w_i \leq k-1] + p \Pr[w_i \leq k+1]. \end{aligned}$$

If  $k = 1$ , then we have

$$\begin{aligned} \Pr[w'_{i+1} \leq 0] &= q \Pr[w'_i \leq 0] + q \Pr[w'_i = 1] \\ &= q \Pr[w'_i \leq 1] \end{aligned}$$

and

$$\begin{aligned} \Pr[w_{i+1} \leq 0] &\geq q \Pr[w_i \leq 0] + q \Pr[w_i = 1] \\ &= q \Pr[w_i \leq 1]. \end{aligned}$$

The lemma now follows by induction. ■

We now establish a probabilistic comparison of the number of steps it takes for the random walks  $\omega$  and  $\omega'$  to reach node 0 starting from a given node  $i$ . Let  $z_i(\sigma)$  be the random variable denoting the number of steps taken to reach node 0 starting from node  $i$ , for a random walk  $\sigma$ .

**Lemma 8.12** *For all  $\ell$  and all  $i > 0$ , we have  $\Pr[z_i(\omega') \leq \ell] \leq \Pr[z_{i-1}(\omega') \leq \ell]$ ,*

**Proof:** We use induction on  $\ell$ . The base case  $\ell = 0$  is trivial. Let  $\ell \geq 1$ . If  $i > 2$  then

$$\begin{aligned} \Pr[z_{i-1}(\omega') \leq \ell] &= p \Pr[z_{i-2}(\omega') \leq \ell-1] + (1-p) \Pr[z_i(\omega') \leq \ell-1] \\ &\geq p \Pr[z_{i-1}(\omega') \leq \ell-1] + (1-p) \Pr[z_{i+1}(\omega') \leq \ell-1] \\ &= \Pr[z_i(\omega') \leq \ell], \end{aligned}$$

where the second step follows from the induction hypothesis. If  $i = 2$ , then we have

$$\begin{aligned}
\Pr[z_1(\omega') \leq \ell] &= q + (1 - q - (1 - p)) \Pr[z_1(\omega') \leq \ell - 1] + (1 - p) \Pr[z_2(\omega') \leq \ell - 1] \\
&\geq p \Pr[z_1(\omega') \leq \ell - 1] + (1 - p) \Pr[z_2(\omega') \leq \ell - 1] \\
&\geq p \Pr[z_1(\omega') \leq \ell - 1] + (1 - p) \Pr[z_3(\omega') \leq \ell - 1] \\
&= \Pr[z_2(\omega') \leq \ell],
\end{aligned}$$

where the second step follows from the induction hypothesis. If  $i = 1$  then we have

$$\begin{aligned}
\Pr[z_0(\omega') \leq \ell] &= q + (1 - q) \Pr[z_1(\omega') \leq \ell - 1] \\
&\geq q \Pr[z_0(\omega') \leq \ell - 1] + (p - q) \Pr[z_1(\omega') \leq \ell - 1] + (1 - p) \Pr[z_2(\omega') \leq \ell - 1] \\
&= \Pr[z_1(\omega') \leq \ell],
\end{aligned}$$

where the second step follows from the induction hypothesis. ■

We now use Lemma 8.12 to argue that the random variable  $z(\omega)$  is stochastically dominated by the random variable  $z(\omega')$ .

**Lemma 8.13** *For all  $i$  and  $\ell$  in  $\mathbf{N}$ , we have  $\Pr[z_i(\omega) \leq \ell] \geq \Pr[z_i(\omega') \leq \ell]$ .*

**Proof:** The proof is by induction on  $\ell$ . The base case  $\ell = 0$  is trivial. Let  $p_j = \Pr[w_{i+1} \leq j - 1 \mid w_i = j]$ , for  $j > 1$ , and  $q_j = \Pr[w_{i+1} = j \mid w_i = j]$ , for all  $j$  in  $\mathbf{N}$ . Note that the following inequalities hold: (i)  $p \leq p_j$ , for all  $j > 1$ , (ii)  $q \leq \min\{p_1, q_0\}$ , and (iii)  $p \geq q$ .

If  $i \geq 2$ , then we have

$$\begin{aligned}
\Pr[z_i(\omega') \leq \ell] &= p \Pr[z_{i-1}(\omega') \leq \ell - 1] + (1 - p) \Pr[z_{i+1}(\omega') \leq \ell - 1] \\
&\leq p_i \Pr[z_{i-1}(\omega') \leq \ell - 1] + (1 - p_i) \Pr[z_{i+1}(\omega') \leq \ell - 1] \\
&\leq p_i \Pr[z_{i-1}(\omega') \leq \ell - 1] + q_i \Pr[z_i(\omega') \leq \ell - 1] + (1 - p_i - q_i) \Pr[z_{i+1}(\omega') \leq \ell - 1] \\
&\leq p_i \Pr[z_{i-1}(\omega) \leq \ell - 1] + q_i \Pr[z_i(\omega) \leq \ell - 1] + (1 - p_i - q_i) \Pr[z_{i+1}(\omega) \leq \ell - 1] \\
&= \Pr[z_i(\omega) \leq \ell].
\end{aligned}$$

(The second step holds because: (i)  $p \leq p_i$ , and (ii)  $\Pr[z_{i-1}(\omega') \leq \ell - 1] \geq \Pr[z_{i+1}(\omega') \leq \ell - 1]$ , which follows from Lemma 8.12. The third step holds since  $\Pr[z_i(\omega') \leq \ell - 1] \geq \Pr[z_{i+1}(\omega') \leq \ell - 1]$  by Lemma 8.12. The fourth equation follows from the induction hypothesis.) For  $i = 1$ , we have

$$\begin{aligned}
\Pr[z_1(\omega') \leq \ell] &= q + (p - q) \Pr[z_1(\omega') \leq \ell - 1] + (1 - p) \Pr[z_2(\omega') \leq \ell - 1] \\
&\leq p_1 + q_1 \Pr[z_1(\omega') \leq \ell - 1] + (1 - p_1 - q_1) \Pr[z_2(\omega') \leq \ell - 1] \\
&\leq p_1 + q_1 \Pr[z_1(\omega) \leq \ell - 1] + (1 - p_1 - q_1) \Pr[z_2(\omega) \leq \ell - 1] \\
&= \Pr[z_1(\omega) \leq \ell].
\end{aligned}$$

(The second step holds because: (i)  $q \leq p_1$ , (ii)  $1 - p \geq 1 - p_1 - q_1$ , and (iii)  $\Pr[z_1(\omega') \leq \ell - 1] \geq \Pr[z_2(\omega') \leq \ell - 1]$ , which follows from Lemma 8.12. The third step follows from the induction hypothesis.)

For  $i = 0$ , we have

$$\begin{aligned}
\Pr[z_0(\omega') \leq \ell] &= q + (1 - q) \Pr[z_1(\omega') \leq \ell - 1] \\
&\leq q_0 + (1 - q_0) \Pr[z_1(\omega') \leq \ell - 1] \\
&\leq q_0 + (1 - q_0) \Pr[z_1(\omega) \leq \ell - 1] \\
&= \Pr[z_0(\omega) \leq \ell].
\end{aligned}$$

(The second step holds because  $q \leq q_0$ . The third step follows from the induction hypothesis.) We have thus established the desired claim.  $\blacksquare$

We now show that, in a probabilistic sense, the time to return to 0 is smaller for  $\omega$  than for  $\omega'$ . For any  $i$ , let  $\tau_i$  (resp.,  $\tau'_i$ ) denote the smallest  $j \geq 0$  such that  $w_{i+j} = 0$  (resp.,  $w'_{i+j} = 0$ ). We note that by letting  $\langle w \rangle$  represent  $\langle \max\{s', t\} \rangle$ , the terminating step  $\tau$  is given by  $i^* + \tau_{i^*}$ . Lemma 8.14 shows that, for any  $i$ , the random variable  $\tau_i$  is stochastically dominated by the random variable  $\tau'_i$ .

**Lemma 8.14** *For any  $i$  and  $j \geq i$ , we have  $\Pr[\tau_i \leq j] \geq \Pr[\tau'_i \leq j]$ .*

**Proof:** We have

$$\begin{aligned} \Pr[\tau_i \leq j] &= \sum_{0 \leq k \leq i} \Pr[w_i = k] \Pr[z_k(\omega) \leq j] \\ &\geq \sum_{0 \leq k \leq i} \Pr[w_i = k] \Pr[z_k(\omega') \leq j] \\ &\geq \sum_{0 \leq k \leq i} \Pr[w'_i = k] \Pr[z_k(\omega') \leq j] \\ &= \Pr[\tau'_i \leq j]. \end{aligned}$$

(The first step follows from the definitions of  $\tau_i$  and  $z_i(\omega)$ . For the second step, we use Lemma 8.13. For the third step, we first invoke Lemma 8.11 and then invoke Lemma A.1 with the substitution  $(i, k, \Pr[w_i = k], \Pr[w'_i = k], \Pr[z_k(\omega') \leq j])$  for  $(m, i, p_i, q_i, n_i)$ . We note that one of the conditions for the latter invocation, namely,  $\Pr[z_k(\omega') \leq j]$  is nonincreasing with  $k$ , follows from Lemma 8.12. The fourth step follows from the definitions of  $\tau'_i$  and  $z_i(\omega')$ .  $\blacksquare$ )

Lemma 8.14 indicates that we can obtain an upper bound on the time taken for the random walk  $\omega$  to return to 0 by deriving a corresponding bound for the random walk  $\omega'$ . Indeed, we will use Lemma 8.14 to obtain an upper bound on the length of any “excursion” in  $\omega$ . An *excursion* of length  $\ell$  in a graph  $W$  with node set  $\mathbf{N}$  is a walk that starts at node 0 and first returns to the start node at time  $\ell$ , for all  $\ell$  in  $\mathbf{N}$ . For all  $i$  such that  $w_i = 0$ , let  $\ell_i(\omega)$  be the random variable that gives the length of the excursion in  $\omega$  starting at time  $i$ . We note that for all  $i$ ,  $\ell_i(\omega)$  equals  $z_0(\omega)$ .

The following lemma, which describes a probabilistic recurrence relation for the length of an excursion in  $\omega'$ , is proved using a classical combinatorial result known as Raney’s lemma [8, 18].

**Lemma 8.15** *Let  $p$  and  $q$  satisfy the inequality  $1 - p \leq (p - q)^2$ . For all  $i$  and  $\ell$  in  $\mathbf{N}$ , we have  $\Pr[\ell_i(\omega') = \ell + 1 \mid w'_i = 0] \leq \max\{1 - q, 5(p - q)\} \Pr[\ell_i(\omega') = \ell \mid w'_i = 0]$ .*

**Proof:** Since  $\omega'$  is a random walk,

$$\Pr[\ell_i(\omega') = \ell \mid (w'_0, \dots, w'_{i-1}, w'_i) = (u_0, \dots, u_{i-1}, 0)] = \Pr[\ell_0(\omega') = \ell \mid w'_0 = 0]$$

for any  $u_0, \dots, u_{i-1}$  in  $\mathbf{N}$ . For the remainder of the proof, we assume without loss of generality that  $i$  is 0.

For  $\ell = 1$ , the desired claim holds since  $\Pr[\ell_0(\omega') = 2] / \Pr[\ell_0(\omega') = 1] = (1 - q)$ . We now consider  $\ell \geq 2$ . Let  $\mathcal{E}_j$  denote the event that the random walk does not reach node 0 in the first  $j$  steps. That is,  $\mathcal{E}_j$  is the event that  $w'_k$  is nonzero for all  $k$  in  $[1, j]$ . For all  $j$ , let  $\alpha_j$  denote the

probability that  $w'_{j+1}$  is 1 and  $\mathcal{E}_{j+1}$  holds, given that  $w'_1$  is 1. For convenience, we assume that  $\alpha_{-1}$  equals  $1/(p-q)$ . We obtain that

$$\Pr[\ell_0(\omega') = \ell] = (1-q) \cdot \alpha_{\ell-2} \cdot q. \quad (10)$$

It thus follows that the ratio of  $\Pr[\ell_0(\omega') = \ell + 1]$  and  $\Pr[\ell_0(\omega') = \ell]$  equals  $\alpha_{\ell-1}/\alpha_{\ell-2}$ . The remainder of the proof is devoted to obtaining an upper bound on  $\alpha_{j+1}/\alpha_j$  for all  $j \geq 0$ .

Let  $\beta_m$  denote the probability that the following conditions hold given that  $w'_1 = 1$ : (i)  $\mathcal{E}_{2m+1}$  holds, (ii)  $w'_{2m+1} = 1$ , and (iii) the edge  $(1, 1)$  is not traversed in any of the first  $2m+1$  steps. Using Raney's lemma [8, 18], we have  $\beta_m = \frac{1}{2m+1} \binom{2m+1}{m} (p(1-p))^m$ . By the definitions of  $\alpha_j$  and  $\beta_m$ , it follows that

$$\begin{aligned} \alpha_j &= \sum_{0 \leq m \leq \lfloor j/2 \rfloor} \beta_m \cdot (p-q) \cdot \alpha_{j-2m-1} \\ &= \sum_{0 \leq m \leq \lfloor j/2 \rfloor} \frac{1}{2m+1} \binom{2m+1}{m} (p(1-p))^m \cdot (p-q) \cdot \alpha_{j-2m-1}. \end{aligned}$$

We now prove by induction on  $j \geq 2$  that  $\alpha_{j+1}/\alpha_j$  is at most  $5(p-q)$ . The induction base holds since  $\alpha_0$  is 1 and  $\alpha_1$  is  $5(p-q)$ . For the induction hypothesis, we assume that  $\alpha_{j+1}/\alpha_j$  is at most  $5(p-q)$  for all  $j \leq k-1$ . If  $k$  is even, then we have

$$\begin{aligned} \alpha_{k+1}/\alpha_k &\leq \max_{0 \leq m \leq k/2} \alpha_{k-2m}/\alpha_{k-2m-1} \\ &\leq 5(p-q), \end{aligned}$$

where the last step follows from the induction hypothesis. If  $k$  is odd, then we have

$$\begin{aligned} \alpha_{k+1}/\alpha_k &\leq \max \left\{ \left( \frac{1}{k} \binom{k}{(k-1)/2} (p-q)^2 + \frac{1}{k+2} \binom{k+2}{(k+1)/2} p \right) / \left( \frac{1}{k} \binom{k}{(k-1)/2} (p-q) \right), \right. \\ &\quad \left. \max_{0 \leq m \leq (k-3)/2} \alpha_{k-2m}/\alpha_{k-2m-1} \right\} \\ &\leq \max\{5(p-q), 5(p-q)\} \\ &= 5(p-q), \end{aligned}$$

where the second step follows from the induction hypothesis along with the inequalities  $1-p \leq (p-q)^2$  and

$$\binom{k+2}{(k+1)/2} \leq 4 \binom{k}{(k-1)/2}.$$

The claim of the lemma follows from the upper bound on  $\alpha_{k+1}/\alpha_k$  and Equation 10. ■

We are now ready to use the properties of the random walks  $\omega$  and  $\omega'$  that are stated in Lemmas 8.14 and 8.15 to analyze the random walk obtained by the sequence  $\langle \max\{s', t\} \rangle$ . We set  $p = 1 - 2\varepsilon^2$  and  $q = 1 - 2\varepsilon$ . Lemmas 8.6 and 8.7 imply that  $\omega$  characterizes the random walk corresponding to the sequence  $\langle \max\{s', t\} \rangle$ . Consider the random walk  $\omega'$ . We define a sequence  $\langle v \rangle$  associated with  $\langle w' \rangle$  as follows: If  $w'_j = 0$  then  $v_j = G$ ; otherwise,  $v_j = B$ .

**Lemma 8.16** *Let  $i$  be in  $[(\log n/b) - 1]$ . Given any fixed sequence  $\langle v \rangle_{i-1}$  of  $B, G$  values, the probability that  $w'_i$  is 0 is at least  $1 - 10\varepsilon$ .*

**Proof:** Assume that  $v_j = G$ . What is the probability that  $v_i = G$ ,  $i > j$ , if we know that  $v_k = B$ , for all integers  $k$  in the interval  $[j + 1, i]$ ? From Lemma 8.15, it follows that this probability is at least  $1 - 10\varepsilon$ . This is because Lemma 8.15 states that  $1 - \max\{2\varepsilon, 10(\varepsilon - \varepsilon^2)\}$  is a lower bound on the probability that there is an excursion of length  $i - j$  starting at  $j$  in  $H'$ , given that there is an excursion of length at least  $i - j$  starting at  $j$  in  $H'$ . (Note that  $1 - p \leq (p - q)^2$  since  $\varepsilon < 1/10$  by Equation 6.)

Given any fixed  $B, G$  sequence  $\langle u \rangle_{j-1}$ ,  $\Pr[v_i = G \mid (v_0, \dots, v_{i-1}) = (u_0, \dots, u_{j-1}, G, B, \dots, B)] = \Pr[v_i = G \mid (v_j, \dots, v_{i-1}) = (G, B, \dots, B)]$ . Since this holds for any  $j > 0$  and since  $w'_i = 0$  if and only if  $v_i = G$ , we have  $\Pr[w'_i \mid (v_0, \dots, v_{i-1}) = (u_0, \dots, u_{i-1})] \geq 1 - 10\varepsilon$ . ■

Our main technical claim concerning the random walk  $\omega$  now follows from Lemmas 8.14 and 8.16.

**Lemma 8.17** *For any  $i$  in  $[(\log n)/b - 1]$  and any nonnegative integer  $j$ , the probability that  $\tau_i$  is at least  $j$  is at most  $(10\varepsilon)^j$ .*

**Proof:** By Lemma 8.16, the probability that  $\tau'_i$  is at least  $j$  is at most  $(10\varepsilon)^j$ . The desired claim then follows from Lemma 8.14. ■

#### 8.2.4 Proof of Theorem 1

We first derive upper bounds on  $E[c(x_i, x_{i+1})]$  and  $E[c(y_i, y_{i+1})]$ , for all  $i$ , using Lemma 8.17. Recall that  $a_i$  and  $b_i$  denote the radii of  $X_i^1$  and  $Y_i^1$ , respectively, and  $i^*$  is the smallest integer  $i$  such that  $a_i$  is at least  $c(x, y)$ .

**Lemma 8.18** *For any  $i$  in  $[(\log n)/b - 1]$ ,  $E[c(x_i, x_{i+1})] = O(a_i)$  and  $E[c(y_i, y_{i+1})] = O(b_i)$ .*

**Proof:** We first observe that  $c(x_i, x_{i+1})$  (resp.,  $c(y_i, y_{i+1})$ ) is at most  $a_k$  (resp.,  $b_k$ ), where  $k$  is the least  $j \geq i$  such that  $s_j$  (resp.,  $t_j$ ) belongs to  $\{0, 1, 2\}$  (resp.,  $\{0\}$ ); if such a  $j$  does not exist, then  $k$  is  $(\log n)/b - 1$ . Thus,  $k$  is at most  $i + \tau_i$ . By Lemma 8.17, it follows that for any  $j \geq i$ , the probability that  $k \geq j$  is at most  $(10\varepsilon)^{j-i}$ . By Lemma 8.4, we thus have

$$E[c(x_i, x_{i+1})] \leq \sum_{j \geq i} a_i (10\varepsilon)^{j-i} 2^{\lceil b \log_\delta 2 \rceil (j-i)} = O(a_i)$$

and

$$E[c(y_i, y_{i+1})] \leq \sum_{j \geq i} b_i (10\varepsilon)^{j-i} 2^{\lceil b \log_\delta 2 \rceil (j-i)} = O(b_i),$$

since  $10\varepsilon 2^{\lceil b \log_\delta 2 \rceil} < 1$  by Equation 6. ■

We now use Lemmas 8.4, 8.17, and 8.18 to establish Theorem 1.

**Proof of Theorem 1:** By Equation 9, the expected cost of the read operation is bounded by the expected value of  $f(\ell(A)) \sum_{0 \leq i < \tau} O(c(x_i, x_{i+1}) + c(y_i, y_{i+1}))$ . (Recall that  $\tau$  is the smallest integer  $i \geq i^*$  such that  $(s_i, t_i) = (0, 0)$ .) We upper bound the two terms  $E[\sum_{0 \leq i < i^*} (c(x_i, x_{i+1}) + c(y_i, y_{i+1}))]$  and  $E[\sum_{i^* \leq i < \tau} (c(x_i, x_{i+1}) + c(y_i, y_{i+1}))]$  separately. By Lemmas 8.4 and 8.18, the first term is

$O(a_{i^*} + b_{i^*})$ . We upper bound the second term as follows. Since  $\tau$  is  $i^* + \tau_{i^*}$ , we obtain from Lemma 8.17 that for any  $j \geq 0$ , the probability that  $\tau \geq i^* + j$  is at most  $(10\varepsilon)^j$ . Therefore,

$$\begin{aligned} E\left[\sum_{i^* \leq i < \tau} (c(x_i, x_{i+1}) + c(y_i, y_{i+1}))\right] &\leq \sum_{j \geq 0} j(10\varepsilon)^j (a_{i^*+j} + b_{i^*+j}) \\ &\leq \sum_{j \geq 0} j(10\varepsilon)^j 2^{j \lceil b \log_\delta 2 \rceil} (a_{i^*} + b_{i^*}) \\ &= O(a_{i^*} + b_{i^*}) \\ &= O(c(x, y)). \end{aligned}$$

(The second step follows from Lemma 8.4. The third step holds since  $10\varepsilon 2^{\lceil b \log_\delta 2 \rceil} < 1$  by Equation 6. The fourth step follows from Lemma 8.4.)  $\blacksquare$

## 9 Analysis for Other Performance Bounds

We begin by establishing some useful properties of neighbors in Section 9.1. These properties are then used to derive bounds on the cost of the insert and delete operations, the size of the auxiliary memory, and the adaptability of our access scheme in Sections 9.2, 9.3, and 9.4, respectively.

### 9.1 Properties of Neighbors

In this section, we establish certain claims concerning the different types of neighbors that are defined in Section 4. We differentiate between root and nonroot primary  $(i, j)$ -neighbors. A *root primary*  $(i, j)$ -neighbor  $w$  of  $v$  is a primary  $(i, j)$ -neighbor  $w$  of  $v$  such that either  $w[i] \neq j$  or  $i = (\log n)/b - 1$ . A primary neighbor that is not a root primary neighbor is a *nonroot primary* neighbor. Note that, for  $i < (\log n)/b - 1$ , if  $u$  is a root primary  $(i, j)$ -neighbor of  $v$ , then  $u[\ell]$  equals  $v[\ell]$ , for each  $\ell$  in  $[i]$ , and there is no node  $w$  in  $V$  such that  $w[i]$  equals  $j$  and  $w[\ell]$  equals  $v[\ell]$ , for each  $\ell$  in  $[i]$ . We illustrate the difference between a root primary neighbor and a nonroot primary neighbor by an example. Consider a network of  $n = 2^6$  nodes. Thus, each node has a 6-bit label. Let  $b$  equal 2. Consider nodes  $u$  and  $v$  with labels 100111, and 110011, respectively. If  $v$  is a primary  $(1, 00)$ -neighbor of  $u$ , then  $v$  is a nonroot primary neighbor. On the other hand, if  $v$  is a primary  $(1, 01)$  neighbor of  $u$ , then  $v$  is a root primary neighbor since  $v[1] = 00 \neq 01$ . Moreover, this case implies that there is no node in the network with a label for which the four rightmost bits are 1101. (Note that the neighbor table is constructed by matching rightmost bits.) Finally, we observe that since  $(\log n)/b - 1 = 2$ , any primary  $(2, \cdot)$  neighbor is a root primary neighbor.

**Lemma 9.1** *Let  $u$  and  $v$  be in  $V$ , and let  $k$  denote  $|M(u, c(u, v))|$ . For any  $j$  in  $[2^b]$ , we have (i) for any  $i$  in  $[(\log n)/b - 1]$ , the probability that  $u$  is a primary  $(i, j)$ -neighbor of  $v$  is at most  $e^{-((k/\Delta)-2)/2^{(i+1)b}}$ , and (ii) for any  $i$  in  $[(\log n)/b]$ , the probability that  $u$  is a root primary  $(i, j)$ -neighbor of  $v$  is at most  $e^{-n/2^{(i+1)b}}$ .*

**Proof:** Consider the ball  $M(v, c(v, u))$ . By Equation 1,  $|M(v, c(v, u))| \geq |M(v, 2c(v, u))|/\Delta$ . Since  $M(v, 2c(v, u))$  is a superset of  $M(u, c(u, v))$ , we have  $|M(v, c(v, u))| \geq k/\Delta$ . The probability that a node  $w$  in  $M(v, c(u, v)) \setminus \{u, v\}$  does not match the label of  $v$  in its  $(i+1)b$  rightmost bits is at most  $1 - 1/2^{(i+1)b}$ . Since  $i$  is less than  $(\log n)/b - 1$ , the probability that  $u$  is a primary  $(i, j)$ -neighbor of  $v$  is at most

$$\begin{aligned} &(1 - 1/2^{(i+1)b})^{(k/\Delta)-2} \\ &\leq e^{-((k/\Delta)-2)/2^{(i+1)b}}. \end{aligned}$$

If  $u$  is a root primary  $(i, j)$ -neighbor of  $v$ , then  $u[\ell]$  equals  $v[\ell]$  for each  $\ell$  in  $[i]$  and there is no node  $w$  in  $V$  such that  $w[i]$  equals  $j$  and  $w[\ell]$  equals  $v[\ell]$  for each  $\ell$  in  $[i]$ . Therefore, the probability that  $u$  is a root primary  $(i, j)$ -neighbor of  $v$  is at most

$$\begin{aligned} & (1/2^{ib})(1 - 1/2^{(i+1)b})^{n-1}(1 - 1/2^b) \\ & \leq (1/2^{ib})(1 - 1/2^{(i+1)b})^n \\ & \leq (1/2^{ib})e^{-n/2^{(i+1)b}}. \end{aligned}$$

■

**Corollary 9.1.1** *Let  $u$  and  $v$  be in  $V$ , let  $i$  be in  $[(\log n)/b]$ , and let  $j$  be in  $[2^b]$ . If  $u$  is a primary  $(i, j)$ -neighbor of  $v$ , then  $v$  is in  $N(u, O(2^{ib} \log n))$  whp.* ■

Lemma 9.2 and Corollary 9.2.1 below establish bounds on the number of nodes  $v$  such that  $u$  is a primary or secondary neighbor of  $v$ , and on the number of nodes  $v$  such that  $v$  is a reverse neighbor of  $u$ , respectively. For any  $u$  in  $V$ , let  $a_u$  denote the total number of triples  $(i, j, v)$  such that  $i$  belongs to  $[(\log n)/b]$ ,  $j$  belongs to  $[2^b]$ ,  $v$  belongs to  $V$ , and  $u$  is a primary or secondary  $(i, j)$ -neighbor of  $v$ . Lemma 9.2 is used in the proof of Theorem 4, while Corollary 9.2.1 is used in the proofs of Theorems 2 and 3.

**Lemma 9.2** *Let  $u$  be in  $V$  and let  $i$  be in  $[(\log n)/b]$ . Then the number of nodes for which  $u$  is an  $i$ th level primary neighbor is  $O(\log n)$  whp. Also,  $E[a_u] = O(\log n)$  and  $a_u$  is  $O(\log^2 n)$  whp.*

**Proof:** Given a node  $v$  in  $V$ ,  $i$  in  $[(\log n)/b - 1]$ , and  $j$  in  $[2^b]$ , it follows from Lemma 9.1 that the probability that  $u$  is a root primary  $(i, j)$ -neighbor of  $v$  is at most  $(1/2^{ib})e^{-n/2^{(i+1)b}}$ . Given a node  $v$  in  $V$  and  $j$  in  $[2^b]$ , the probability that  $u$  is a root  $((\log n)/b, j)$ -primary neighbor of  $v$  is at most  $1/n$ .

Fix  $j$  in  $[2^b]$ . Let  $\ell$  equal  $(\log n - \log \log n)/b - \Omega(1)$ , where the constant in the  $\Omega(1)$  term is chosen sufficiently large. We consider two cases:  $i < \ell$  and  $i \geq \ell$ . If  $i < \ell$ , then the probability that there exists  $v$  in  $V$  such that  $u$  is a root primary  $(i, j)$ -neighbor of  $v$  is at most

$$\begin{aligned} & n(1/2^{ib})e^{-n/2^{(i+1)b}} \\ & \leq ne^{-\Omega(\log n)} \\ & = O(1/\text{poly}(n)). \end{aligned}$$

If  $i \geq \ell$ , then given  $v$  in  $V$ , the probability that  $u$  is a root primary  $(i, j)$ -neighbor of  $v$  is at most  $1/2^{\ell b} = O((\log n)/n)$ . It follows from Chernoff bounds [6] that the number of nodes for which  $u$  is a root primary  $(i, j)$ -neighbor is  $O(\log n)$  whp.

We now consider secondary and nonroot primary neighbors. For any  $i$  in  $[(\log n)/b]$ , if  $u$  is a secondary or nonroot primary  $(i, j)$ -neighbor of  $v$  then  $j$  is  $u[i]$  and  $u$  is one of the  $d + 1$  nodes  $w$  in  $V$  with minimum  $c(v, w)$  whose lowest  $ib$  bits match those of  $v$ . We now fix  $u$  and  $i$ , set  $j$  to  $u[i]$ , and obtain an upper bound on the probability that  $u$  is one of the at most  $d + 1$  nodes  $w$  with minimum  $c(v, w)$  and whose first  $ib$  bits match those of  $v$ .

Consider a node  $v$  in  $N(u, \mu^{k+1}2^{(i+1)b}) \setminus N(u, \mu^k 2^{(i+1)b})$ , where  $\mu$  is a real constant to be specified later. If  $k$  equals zero, then the probability that  $u$  is a primary or secondary  $(i, j)$ -neighbor of  $v$  is at most  $1/2^{ib}$ . Otherwise, consider the ball  $M(v, c(v, u))$ . By the low-expansion condition (i.e.,



the right inequality of Equation 1),  $|M(v, c(v, u))|$  is at least  $|M(v, 2c(v, u))|/\Delta$ . We are given that  $M(u, c(u, v))$  is a superset of  $N(u, \mu^k 2^{(i+1)b})$ . Since  $M(v, 2c(v, u))$  is a superset of  $M(u, c(u, v))$ , we obtain that  $|M(v, c(v, u))|$  is at least  $\mu^k 2^{(i+1)b}/\Delta$ . The probability that  $u$  is a primary or secondary  $(i, j)$ -neighbor of  $v$  is at most

$$\begin{aligned} & d \binom{\mu^k 2^{(i+1)b}/\Delta}{d} (1 - 1/2^{(i+1)b})^{(\mu^k 2^{(i+1)b}/\Delta) - d} / (2^{ib} 2^{(i+1)bd}) \\ & \leq d(e\mu^k 2^{(i+1)b}/(\Delta d))^d e^{-\mu^k/\Delta} (1 - 1/2^{(i+1)b})^{-d} / (2^{ib} 2^{(i+1)bd}) \\ & \leq 4d(e\mu^k/(\Delta d))^d (e^{-\mu^k/\Delta}/2^{ib}) \\ & \leq 1/((2\mu)^k 2^{ib}). \end{aligned}$$

(The second step holds since  $d \leq 2^b \leq 2^{ib}$  and  $(1 - 1/2^{ib})^{-2^{ib}}$  is at most 4. The third step follows by choosing  $\mu$  large enough with respect to  $\Delta$  and  $d$  such that  $e^{\mu^k/\Delta} \geq (2^k \Delta^k / d^{d-1})(\mu^k/\Delta)^{d+1}$  for all  $k \geq 1$ .)

Thus, the expected number of nodes for which  $u$  is a secondary or nonroot primary neighbor is at most

$$\begin{aligned} & \sum_{i \in [(\log n)/b], j=u[i]} \sum_{k \geq 0} \sum_{v \in N(u, \mu^{k+1} 2^{(i+1)b}) \setminus N(u, \mu^k 2^{(i+1)b})} 1/((2\mu)^k 2^{ib}) \\ & \leq \sum_{i \in [(\log n)/b], j=u[i]} 2^b \mu \\ & = O(\log n). \end{aligned}$$

To obtain a high probability bound on the number of nodes for which  $u$  is a secondary or nonroot primary neighbor, we proceed as follows. For any  $v$  not in  $N(u, \Theta(2^{(i+1)b} \log n))$ , it follows from Lemma 9.1 that the probability that  $u$  is a secondary or nonroot primary  $(i, j)$ -neighbor of  $v$  is  $O(1/\text{poly}(n))$ . For any  $v$  in  $N(u, \Theta(2^{(i+1)b} \log n))$ , the probability that  $u$  is a secondary or nonroot primary  $(i, j)$ -neighbor of  $v$  is at most  $1/2^{(i+1)b}$ . Therefore, the number of nodes for which  $u$  is a secondary or nonroot primary neighbor is  $O(\log^2 n)$  whp.

The bounds on expectation and the high probability bounds together establish that  $E[a_u]$  is  $O(\log n)$  and  $a_u$  is  $O(\log^2 n)$  whp.  $\blacksquare$

**Corollary 9.2.1** *For any  $u$  in  $V$ , the total number of reverse neighbors of  $u$  is  $O(\log^2 n)$  whp, and  $O(\log n)$  expected.*

**Proof:** The desired claim follows directly from Lemma 9.2, since  $v$  is a reverse  $(i, j)$ -neighbor of  $u$  if and only if  $u$  is a primary  $(i, j)$ -neighbor of  $v$ .  $\blacksquare$

Lemma 9.3 is used in the proof of Theorem 3. For a given a node  $u$ , it provides a bound on the number of primary neighbor sequences that have  $u$  in the  $i$ th position. For any  $u$  and  $v$  in  $V$  and  $i$  in  $[(\log n)/b]$ ,  $v$  is said to be an  $i$ -leaf of  $u$  if there exists a sequence  $v = v_0, v_1, \dots, v_{i-1}, v_i = u$ , such that for all  $j$  in  $[i]$ ,  $v_{j+1}$  is a primary  $(j, v_{j+1}[j])$ -neighbor of  $v_j$ .

**Lemma 9.3** *Let  $u$  belong to  $V$ , and let  $i$  be in  $[(\log n)/b]$ . Then the number of  $i$ -leaves of  $u$  is  $O(2^{ib} \log n)$  whp.*

**Proof:** We establish the lemma by showing that if  $v$  is an  $i$ -leaf of  $u$ , then  $v$  is in  $N(u, O(2^{ib} \log n))$  whp. By Corollary 9.1.1, we have that for all  $j$  in  $[i]$ ,  $v_j$  is in  $N(v_{j+1}, c_0 2^{(j+1)b} \log n)$  whp for some sufficiently large real constant  $c_0$ . We will prove by induction on  $j$  in  $[i+1]$  that  $v = v_0$  is in  $N(v_j, c_0 2^{jb} \log n)$  whp.

The induction base follows trivially. For the induction step, assume that  $v$  is in  $N(v_j, c_0 2^{jb} \log n)$ . By Corollary 9.1.1,  $v_j$  belongs to  $N(v_{j+1}, c_0 2^{jb} \log n)$  whp. Let  $r_0$  and  $r_1$  denote  $c(v_{j+1}, v_j)$  and  $c(v_j, v)$ , respectively. The node  $v$  is contained in the ball  $M(v_{j+1}, r_0 + r_1)$ . If  $r_0$  is at least  $r_1$ , then  $|M(v_{j+1}, r_0 + r_1)|$  is at most  $\Delta c_0 2^{jb} \log n$  by Equation 1. Otherwise,  $|M(v_{j+1}, r_0 + r_1)|$  is at most  $|M(v_j, 3r_1)|$ , which is at most  $\Delta^2 c_0 2^{jb} \log n$  by Equation 1. We thus obtain that  $v$  is in  $N(v_{j+1}, \Delta^2 c_0 2^{jb} \log n)$ . Since  $\Delta^2 < 2^b$  (by Equation 2), we obtain that  $v$  is in  $N(v_{j+1}, c_0 2^{(j+1)b} \log n)$ .

Applying the above inductive claim with  $j = i$ , we obtain that  $v$  is in  $N(u, O(2^{ib} \log n))$  whp. This completes the proof. ■

## 9.2 Insert and delete operations

Theorem 2 follows from Lemmas 8.4, 8.18, and 9.2.

**Proof of Theorem 2:** Consider an insert operation performed by  $x$  for any object. The expected cost of the operation is bounded by  $E[\sum_{0 \leq i < \log n/b} c(x_i, x_{i+1})]$ , which by Lemmas 8.4 and 8.18 is  $O(a_{(\log n)/b-1}) = O(C)$ .

We now consider the cost of the delete operation. By Lemma 9.2, for each  $i$ , the number of reverse  $(i, j)$ -neighbors of  $x_i$  for any  $j$  is  $O(\log n)$  whp, where  $x_i$  is the  $i$ th node in the primary neighbor sequence of  $x$ . Therefore, the expected cost of the delete operation executed by  $x$  is bounded by the product of  $E[\sum_{0 \leq i < \log n/b} c(x_i, x_{i+1})]$  and  $O(\log n)$ . By Lemma 8.18, it follows that the expected cost of a delete operation is  $O(C \log n)$ . ■

## 9.3 Auxiliary Memory

**Proof of Theorem 3:** We first place an upper bound on the size of the neighbor table of any  $u$  in  $V$ . By definition, the number of primary and secondary neighbors of  $u$  is at most  $(d+1)2^b(\log n)/b$ , which is  $O(\log n)$ . By Corollary 9.2.1, the number of reverse neighbors of  $u$  is  $O(\log^2 n)$  whp.

We next place an upper bound on the size of the pointer list of any  $u$  in  $V$ . The size of  $Ptr(u)$  is at most the number of triples of the form  $(A, v, \cdot)$ , where  $A$  is in  $\mathcal{A}$  and  $v$  is in  $V$  such that (i) there exists  $i$  in  $[(\log n)/b]$  such that  $v$  is an  $i$ -leaf of  $u$ , (ii)  $A[j] = u[j]$  for all  $j$  in  $[i]$ , and (iii)  $A$  is in the main memory of  $v$ .

By Lemma 9.3, the number of  $i$ -leaves of  $u$  is  $O(2^{ib} \log n)$  whp. The probability that  $A[j] = u[j]$ , for all  $j$  in  $[i]$ , is at most  $1/2^{ib}$ . Since the number of objects in the main memory of any node is at most  $\ell$ , it follows that whp,  $|Ptr(u)|$  is at most  $\sum_{i \in [(\log n)/b]} O(\ell \log n)$  which is  $O(\ell \log^2 n)$ .

Combining the bounds on the sizes of the neighbor table and pointer list, we obtain that the size of the auxiliary memory of  $u$  is  $O(\ell \log^2 n)$  whp. ■

## 9.4 Adaptability

**Proof of Theorem 4:** By Lemma 9.2, for any node  $u$ , the number of nodes for which  $u$  is a primary or secondary neighbor is  $O(\log n)$  expected and  $O(\log^2 n)$  whp. Moreover,  $u$  is a reverse neighbor of  $O(\log n)$  nodes since  $u$  has  $O(\log n)$  primary neighbors. Therefore, the adaptability of our scheme is  $O(\log n)$  expected and  $O(\log^2 n)$  whp. ■

## 10 Future Work

We would like to extend our study to more general classes of cost functions and determine tradeoffs among the various complexity measures. It would also be interesting to consider models that allow faults in the network. We believe that our access scheme can be extended to perform well in the presence of faults, as the distribution of control information in our scheme is balanced among the nodes of the network.

## Acknowledgments

The authors would like to thank Madhukar Korupolu and Satish Rao for several helpful discussions.

## References

- [1] B. Awerbuch, Y. Bartal, and A. Fiat. Distributed paging for general networks. *Journal of Algorithms*, 28:67–104, 1998.
- [2] B. Awerbuch and D. Peleg. Online tracking of mobile users. *Journal of the ACM*, 37:1021–1058, 1995.
- [3] B. Awerbuch and D. Peleg. Routing with polynomial communication space tradeoff. *SIAM Journal on Discrete Mathematics*, 5:151–162, 1990.
- [4] B. Awerbuch and D. Peleg. Sparse partitions. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 503–513, 1990.
- [5] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 39–50, May 1992.
- [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] S. Dolev, E. Kranakis, D. Krizanc, and D. Peleg. Bubbles: Adaptive routing scheme for high-speed dynamic networks. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 528–537, May 1995.
- [8] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, MA, 1989.
- [9] J. D. Guyton and M. F. Schwartz. Locating nearby copies of replicated Internet servers. In *Proceedings of the 1995 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 288–298, August 1995.
- [10] D. Karger, E. Lehman, F. T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 654–663, May 1997.
- [11] 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.
- [12] C. E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, C-34:892–900, 1985.

- [13] B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 284–293, October 1997.
- [14] S. J. Mullender, editor. *Distributed Systems*. Addison-Wesley, 1993.
- [15] S. J. Mullender and P. M. B. Vitányi. Distributed match-making. *Algorithmica*, 3:367–391, 1988.
- [16] C. G. Plaxton and R. Rajaraman. Fast fault-tolerant concurrent access to shared objects. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 570–579, October 1996.
- [17] A. G. Ranade. How to emulate shared memory. *Journal of Computer and System Sciences*, 42:307–326, 1991.
- [18] G. N. Raney. Functional composition patterns and power series reversion. *Transactions of the American Mathematical Society*, 94:441–451, 1960.
- [19] E. Upfal and A. Wigderson. How to share memory in a distributed system. *Journal of the ACM*, 34:116–127, 1987.
- [20] M. van Steen, F. J. Hauck, and A. S. Tanenbaum. A model for worldwide tracking of distributed objects. In *Proceedings of the 1996 Conference on Telecommunications Information Networking Architecture (TINA 96)*, pages 203–212, September 1996.

## A A technical lemma

**Lemma A.1** *Let  $m$  be a nonnegative integer and let  $\langle n \rangle$  be a sequence of  $m$  nonincreasing reals. Let  $\langle p \rangle$  and  $\langle q \rangle$  be two sequences of  $m$  reals each such that (i) for all  $j$  in  $[m]$ ,  $\sum_{0 \leq i \leq j} p_i \geq \sum_{0 \leq i \leq j} q_i$  and (ii)  $\sum_{0 \leq i \leq m} p_i = \sum_{0 \leq i \leq m} q_i$ . Then, we have*

$$\sum_{0 \leq i \leq m} p_i n_i \geq \sum_{0 \leq i \leq m} q_i n_i.$$

**Proof:** The proof is by induction on  $m$ . The induction basis is trivial. For the induction hypothesis, we assume that the statement of the lemma holds for  $m$ . We now establish the claim for  $m + 1$ .

$$\begin{aligned} \sum_{0 \leq i \leq m+1} p_i n_i &= q_0 n_0 + (p_0 - q_0) n_0 + \sum_{1 \leq i \leq m+1} p_i n_i \\ &\geq q_0 n_0 + (p_0 - q_0 + p_1) n_1 + \sum_{2 \leq i \leq m+1} p_i n_i \\ &\geq q_0 n_0 + \sum_{1 \leq i \leq m+1} q_i n_i \\ &= \sum_{0 \leq i \leq m} q_i n_i. \end{aligned}$$

(The third step follows from the induction hypothesis and the inequalities  $n_0 \geq n_1$  and  $p_0 \geq q_0$ . We note that the induction hypothesis can be invoked since  $p_0 - q_0 + p_1 + \sum_{2 \leq i \leq j} p_i \leq \sum_{1 \leq i \leq j} q_i$  and  $p_0 - q_0 + p_1 + \sum_{2 \leq i \leq m+1} p_i = \sum_{1 \leq i \leq m+1} q_i$ .) ■