

Minimization of Communication Cost Through Caching in Mobile Environments

A. Prasad Sistla, *Member, IEEE*, Ouri Wolfson, *Member, IEEE*, and Yixiu Huang

Abstract—Users of mobile computers will soon have online access to a large number of databases via wireless networks. Because of limited bandwidth, wireless communication is more expensive than wire communication. In this paper, we present and analyze various static and dynamic data allocation methods. The objective is to optimize the communication cost between a mobile computer and the stationary computer that stores the online database. Analysis is performed in two cost models. One is connection (or time) based, as in cellular telephones, where the user is charged per minute of connection. The other is message based, as in packet radio networks, where the user is charged per message. Our analysis addresses both the average case and the worst case for determining the best allocation method.

Index Terms—Dynamic data allocation, caching, mobile computing, probabilistic analysis, wireless communication, communication cost.

1 INTRODUCTION

USERS of mobile computers, such as palmtops, notebook computers, and personal communication systems, will soon have online access to a large number of databases via wireless networks. The potential market for this activity is estimated to be billions of dollars annually in access and communication charges. For example, while on the road, passengers will access airline and other carriers' schedules, and weather information. Investors will access prices of financial instruments, salespeople will access inventory data, callers will access location dependent data (e.g., where is the nearest taxi-cab, see [10], [24]), and route-planning computers in cars will access traffic information.

Because of limited bandwidth, wireless communication is more expensive than wire communication. For example, a cellular telephone call costs about 35 cents per minute. As another example, RAM Mobile Data Corp. charges on average 8 cents per data message to or from the mobile computer (the actual charge depends on the length of the message). It is clear that for users that perform hundreds of accesses each day, wireless communication can become very expensive. Therefore, it is important that mobile computers access on-line databases in a way that minimizes communication.

We assume that an online database is a collection of data items, where a data item is, for example, a web page or a file. Users access these data items by a unique i.d., such as a key, one at a time. We minimize communication using an appropriate data-allocation scheme. For example, if a user frequently reads a data-item x , and x is updated infrequently, then it is beneficial for the user to allocate a copy of x to her/his mobile computer. In other words, the mobile user subscribes to receive all the updates of x . This way, the reads access the local copy and do not require communication.

The infrequent updates are transmitted from the online database to the mobile computer. In contrast, if the user reads x infrequently compared to the update rate, then a copy of x should not be allocated to the mobile computer. Instead, access should be on-demand; every read request should be sent to the stationary computer that stores the online database.

Thus, one-copy and two-copies are the two possible allocation schemes of the data item x to a mobile computer. In the first scheme, only the stationary computer has a copy of x , whereas, in the second scheme, both the stationary and the mobile computer have a copy of x . An allocation method determines whether or not the allocation scheme changes over time. In a *static* allocation method, the allocation scheme does not change over time, whereas, in a *dynamic* one it does. The following is an example of a dynamic allocation method. The allocation scheme changes from two-copies to one-copy as a result of a larger number of writes than reads in a window of four minutes.

In mobile computing, the geographical area is usually divided into cells, each of which has a stationary controller. Our stationary computer should not be confused with the stationary controller. The stationary computer is some node in the stationary network that is fixed for a given data item and it does not change when the mobile computer moves from cell to cell.

In this paper, we analyze two static allocation methods, namely the one that uses the one-copy scheme and the one that uses the two-copies scheme, and a family of dynamic data allocation methods. These methods are suggested by the need to select the allocation scheme according to the read/write ratio: If the reads are more frequent, then the methods use the two-copies allocation scheme, otherwise, they use the one-copy scheme. The family consists of all the methods that allocate and deallocate a copy of a data item to the mobile computer, based on a sliding window of k requests. For every read or update (we often refer to updates as writes), the latest k requests are examined. If the number

• The authors are with the Electrical Engineering and Computer Science Department, University of Illinois, Chicago, Illinois 60607.
E-mail: wolfson@eecs.uic.edu.

Manuscript received 25 Feb. 1997; revised 1 Sept. 1997.

For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 104666.

of reads is higher than the number of writes and the mobile computer does not have a copy, then such a copy is allocated to the mobile computer; if the number of writes is higher than the number of reads and the mobile computer does have a copy, then the copy is deallocated. Thus, the allocation scheme is dynamically adjusted according to the relative frequencies of reads and writes.

The algorithms in this family are distributed, and they are implemented by software residing on both the mobile and the stationary computers. The different algorithms in this family differ on the size of the window, k .

Our analysis of the static and dynamic algorithms addresses both worst-case and the expected case for reads and writes that are Poisson-distributed. Furthermore, this analysis is done in two cost models. The first is connection (or time) based, where the user is charged per minute of cellular telephone connection. In this model, if the mobile computer reads the item from the stationary database computer, then the read-request, as well as the response, are executed within one connection of minimum length (say one minute). If writes are propagated to the mobile computer, then this propagation is also executed within one minimum-length connection.

The second cost model is message based. In this model, the user is charged per message and the exact charge depends on the length of the message. Therefore, in this model we distinguish between data-messages that are longer and control-messages that are shorter. Data-messages carry the data-item and control messages only carry control information, specifically, read-requests (from the mobile computer to the stationary computer) and delete-requests (the delete-request is a message that deallocates the copy at the mobile computer). Thus, a remote read-request necessitates one control message and the response necessitates a data message. A write propagated to the mobile computer necessitates a data-message.

The rest of the paper is organized as follows. In the next section, we present a summary of the results of this paper. In Section 3, we formally present the model and, in Section 4, we precisely present the sliding-window family of dynamic allocation algorithms. In Section 5, we develop the results in the connection cost model and, in Section 6, we develop the results in the message model. In Section 7, we discuss some other dynamic allocation methods and extensions to handle read, write operations on multiple data items. In Section 8, we compare our work to relevant literature. In Section 9, we discuss the conclusions of our analysis.

2 SUMMARY OF THE RESULTS

We consider a single data item x and a single mobile computer, and we analyze the static allocation methods ST_1 (mobile computer does not have a copy of x) and ST_2 (mobile computer does have a copy of x), and the dynamic allocation methods SW_k (sliding-window with window-size k).

We assume that reads at the mobile computer are issued according to the Poisson distribution with the parameter λ_r , namely, in each time unit the expected number of reads is λ_r . The writes at the stationary computer are

issued independently according to the Poisson distribution with the parameter λ_w . Other requests are ignored in this paper since their cost is not affected by the allocation scheme. We let θ denote $\frac{\lambda_w}{\lambda_r + \lambda_w}$.

Our analysis of each one of the algorithms uses three measures. The first, called expected cost and denoted EXP , gives the expected cost of a read/write request in the case that θ is known and fixed. The second, called average expected cost and denoted AVG , is important for the case where θ is unknown or it varies over time with equal probability of having any value between zero and one. It gives the average expected cost of a request over all possible values of θ .

Our third measure is for the worst case, and it is based on the notion of competitiveness¹ (see [9], [18], [23], [29], [32]) of an on-line algorithm. Intuitively, a data allocation algorithm A is said to be c -competitive if, for any sequence s of read-write requests, the cost of A on s is at most c times as much as the minimum cost, namely, the cost incurred by an ideal offline algorithm that knows the whole sequence of requests in advance (in contrast, our algorithms are online, in the sense that they service the current request without knowing the next request).

In the remainder of this section we summarize the results for each one of the two cost models discussed in the introduction. These results will be interpreted and discussed at the intuitive level in the conclusion section.

2.1 Summary of Results in the Connection Model

In the connection model, our results are as follows. For ST_1 , the expected cost (i.e., expected number of connections) per request is $1 - \theta$ and, for ST_2 , the expected number of connections per request is θ . For SW_k , the expected cost per request is $\theta \cdot \alpha_k + (1 - \theta) \cdot (1 - \alpha_k)$, where α_k is the probability that the majority of k consecutive requests are reads (the formula for this probability is in (5)). Furthermore, we show that for any fixed k , SW_k is not lower than $\min\{\theta, 1 - \theta\}$. Thus, if $\theta \geq \frac{1}{2}$, then the static allocation method ST_1 has the best expected cost per request and, if $\theta \leq \frac{1}{2}$, then the static allocation method ST_2 has the best expected cost per request.

Next, consider the average expected cost. SW_k has the best average (over the possible values of θ) expected cost per request. This cost is $\frac{1}{4} + \frac{1}{4 \cdot k + 8}$, and it decreases as k increases, coming within 6 percent of the optimum for $k = 15$. In contrast, ST_1 and ST_2 both have an average expected cost of $\frac{1}{2}$.

For the worst case, we show that ST_1 and ST_2 are not competitive, i.e., the ratio between their performance and the performance of the optimal, perfect-knowledge algorithm is unbounded. In contrast, we show that SW_k is $(k + 1)$ -competitive, and this competitiveness factor is tight.

1. The traditional worst case complexity as a function of the size of the input is inappropriate since all the algorithms discussed in this paper have the same complexity under this measure. For example, in the connection model, for each algorithm there is a sequence of requests of size m on which the algorithm incurs cost m .

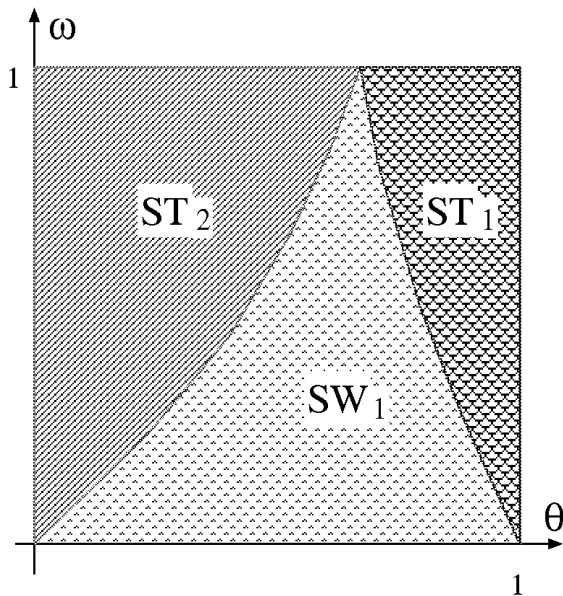


Fig. 1. Superiority coverage in message model.

In summary, in the worst case, the cost of the SW_k family of allocation algorithms increases as k increases, whereas the average expected cost decreases as k increases. The window size k should be chosen to strike a balance between these two conflicting requirements. For example, $k = 15$ may provide a reasonable compromise.

2.2 Summary of Results in the Message Passing Model

In this model, our results are as follows. Let the cost of a data message be 1 and the cost of a control message be ω , where $0 \leq \omega \leq 1$. For ST_1 , the expected cost per request is $(1 + \omega) \cdot (1 - \theta)$ and, for ST_2 , the expected cost is θ . For SW_1 , the expected cost is $\theta \cdot (1 - \theta) \cdot (1 + 2 \cdot \omega)$ and, for SW_k ($k > 1$), we derived the expected cost as a function of ω and θ , as shown in (15) of Section 6.3.² From these formulae of the expected costs, we conclude the following. If $\theta > \frac{1+\omega}{1+2\omega}$, then ST_1 has the best expected cost; if $\theta < \frac{2+\omega}{1+2\omega}$, then ST_2 has the best expected cost; otherwise, namely if $\frac{2+\omega}{1+2\omega} < \theta < \frac{1+\omega}{1+2\omega}$, the SW_1 algorithm has the best expected cost. The dominance graph of these three strategies is shown in Fig. 1. It indicates the superior algorithm for each value of θ and ω .

Next, we consider the average expected cost and we obtain the following results. ST_1 has an average expected cost of $\frac{1+\omega}{2}$; ST_2 has an average expected cost of $\frac{1}{2}$; SW_1 has an average expected cost of $\frac{1+2\omega}{6}$; and the average expected cost of SW_k (for $k \neq 1$) is given by (16) of Section 6.3, and it has a lower bound of $\frac{2+\omega}{8}$. Then, we conclude that, if $\omega \leq 0.4$, then SW_1 has the best average expected cost; if $\omega > 0.4$, then

the average expected cost decreases as the window size k increases (see Corollary 2 in Section 6.3).

For the worst case, we show that, as in the connection cost model, neither ST_1 nor ST_2 are competitive. Similarly, we show that the sliding-window algorithm SW_1 is $(1 + 2 \cdot \omega)$ -competitive, and SW_k (for $k > 1$) is $\left[\left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega\right]$ -competitive.

In summary, the trade-off between the average expected cost and the worst case is similar to the connection model. Namely, a dynamic allocation algorithm is superior to the static ones, with the worst case improving with a decreasing window size; whereas, the average expected cost decreases as the window size increases.

3 THE MODEL

A mobile computer system consists of a mobile computer MC and a stationary computer SC that stores the online database. We consider a data item x that is stored at the stationary computer at all times. Reads and writes are issued at the mobile or stationary computers. Actually, the reads and writes at the stationary computer may have originated at other computers, but the origin is irrelevant in our model. Furthermore, we ignore the reads issued by the stationary computer and the writes issued by the mobile computer, since the cost of each such request is fixed (zero and one, respectively), regardless of whether or not MC has a copy of the data item. Thus, the *relevant* requests are writes that are issued by the stationary computer, and reads that are issued by the mobile computer. A *schedule* is a finite sequence of relevant requests to the data item x . For example, w, r, r, r, w, r, w is a schedule. When each request is issued, either the MC has a copy of the data item or it does not. For the purpose of analysis, we assume that the relevant requests are sequential. In practice, they may occur concurrently, but then some concurrency control mechanism will serialize them; therefore, our analysis still holds. We assume that messages between the stationary computer and each mobile computer are delivered in a first-in-first-out order.

We consider the following two cost models. The first is called the *connection* model. In this model, for each algorithm (static or dynamic), the cost of requests is as follows. If there does not exist a copy of the data item at the MC when a read request is issued, then the read costs one connection (since the data item must be sent from the SC). Otherwise, the read costs zero. For a write at the SC, if the MC has a copy of the data item, then the write costs one connection; otherwise, the write costs zero. The total cost of a schedule ψ , denoted by $COST(\psi)$, is the sum of the costs for all requests in ψ .

The second model is called the *message* cost model. In this model, we assume that a data message cost is 1, and a control message cost is ω . Since the length of a control message is not higher than the length of a data message, $0 \leq \omega \leq 1$. In this model, the cost of requests is as follows. For a read request, if there exists a copy at the MC, then the read does not require communication; otherwise, it necessitates a control message (which forwards the request

2. The SW_1 algorithm is not a special case of the SW_k algorithms, as pointed out at the end of Section 4.

to the SC) and a data message (which transfers the data to the MC) with a total cost of $1 + \omega$.

For a write request, if the MC does not have a copy of the data item, then the write costs 0. Otherwise, the write costs 1, ω , or $1 + \omega$, depending on the algorithm and on the result of the comparison of reads and writes executed by the MC in response to the write request. If the write is propagated to the MC and the MC does not deallocate its copy in response, then the cost is 1; if the MC deallocates its copy in response, then the cost is $1 + \omega$ (ω accounts for the deallocate request). Finally, as will be explained in the next section, SW_1 does not propagate writes to the MC; it simply deallocates the copy at the MC at each write request. Then, the cost of the write is ω .

We assume that the reads issued from the MC are Poisson distributed with parameter λ_r , and the writes issued from the SC are Poisson distributed with parameter λ_w . Denote $\frac{\lambda_w}{\lambda_w + \lambda_r}$ by θ . Observe that, since the Poisson distribution is memoryless, at any point in time θ is the probability that the next request is a write, and $1 - \theta = \frac{\lambda_r}{\lambda_w + \lambda_r}$ is the probability that the next request is a read.

Suppose that A is a data allocation algorithm, and λ_r and λ_w are the read and write distribution parameters, respectively. We denote by $EXP_A(\theta)$ the *expected cost* of a relevant request. Suppose now that θ varies over time with equal probability of having any value between zero and one. Then, we define the *average expected cost* per request, denoted AVG_A , to be the mean value of $EXP_A(\theta)$ for θ ranging from zero to one, namely

$$AVG_A = \int_0^1 EXP_A(\theta) d\theta. \quad (1)$$

The average expected cost should be interpreted as follows. Suppose that time is subdivided into sufficiently large periods, where, in the first period, the reads and writes are distributed with parameters λ_r^1 and λ_w^1 , and $\theta_1 = \frac{\lambda_w^1}{\lambda_w^1 + \lambda_r^1}$; in the second period, the reads and writes are distributed with parameters λ_r^2 and λ_w^2 , and $\theta_2 = \frac{\lambda_w^2}{\lambda_w^2 + \lambda_r^2}$; etc. Suppose further that each θ_i has equal probability of having any value between zero and one (i.e., the probability density function of θ has value one everywhere between zero and one, and is zero everywhere else). In other words, each θ_i is a random number between zero and one. Then, when using the algorithm A , the expected cost of a relevant request over all the periods of time is the integral denoted AVG_A . In other words, AVG_A is the expected value of the expected cost. One can also argue that AVG_A is the appropriate objective cost function when θ is unknown and it has equal probability of having any value between zero and one.

For the worst-case study, we take competitiveness as a measure of the performance of an on-line data allocation algorithm. Formally, a c -competitive data allocation algorithm A is defined as follows. Suppose that M is the perfect

data allocation algorithm that has complete knowledge of all the past and future requests. Data allocation algorithm A is c -competitive if there exist two numbers $c (\geq 1)$, and $b (\geq 0)$, such that for any schedule ψ , $COST_A(\psi) \leq c \cdot COST_M(\psi) + b$. We call c the *competitiveness factor* of the algorithm A . A competitive algorithm bounds the worst-case cost of the algorithm to be within a constant factor of the minimum cost.

We say an algorithm A is *tightly c -competitive* if A is c -competitive and, for any number $d < c$, A is not d -competitive.

4 SLIDING-WINDOW ALGORITHMS

The Sliding-Window(k) algorithm allocates and deallocates a copy of the data item x at the mobile computer. It does so by examining a window of the latest relevant read and write requests. The window is of size k and, for ease of analysis, we assume that k is odd. Recall that the reads are issued at the mobile computer and the writes are issued at the stationary computer.

Observe that at any point in time, whether or not the mobile computer has a copy of x , either the mobile computer or the stationary computer is aware of all the relevant requests. If the mobile computer has a copy of x , then all the reads issued at the mobile computer are satisfied locally, and all the writes issued at the stationary computer are propagated to the mobile computer; thus, the mobile computer receives all the relevant requests. Else, i.e., if the mobile computer does not have a copy, then all reads issued at the mobile computer are sent to the stationary computer; thus, the stationary computer receives all the relevant requests.

Thus, either the mobile computer or the stationary computer (but not both) is *in charge* of maintaining the window of k requests. The window is tracked as a sequence of k bits (e.g., 0 represents a read and 1 represents a write). At the receipt of any relevant request, the computer in charge drops the last bit in the sequence and adds a bit representing the current operation. Then it compares the number of reads and the number of writes in the window.

If the number of reads is bigger than the number of writes and there is a copy of x at the mobile computer, then the SW_k algorithm simply waits for the next operation. If the number of reads is bigger than the number of writes and there is no copy at the mobile computer (i.e., the stationary computer is in charge), then such a copy is allocated as follows. Observe that the last request must have been a read. The stationary computer responds to the read request by sending a copy of x to the mobile computer. The SW_k algorithm piggybacks on this message

- 1) an indication to save the copy in MC's local database, in which the SC also commits to propagate further writes to the MC, and
- 2) the current window of requests.

From this point onward, the MC is in charge.

If the number of writes is bigger than the number of reads and there is no copy of x at the MC, then the SW_k algorithm waits for the next request. If the number of writes is bigger than the number of reads and there is a copy of x

at the MC (i.e., the MC is in charge), then the copy is deallocated as follows. The SW_k algorithm sends to the SC

- 1) an indication that the SC should not propagate further writes to the MC, and
- 2) the current window of requests.

From this point onward the SC is in charge.

This concludes the description of the algorithm and at this point we make two remarks. First, when the window size is 1 and the MC has a copy of x , then a write at the SC will deallocate the copy (since the window will consist of only this write). Therefore, instead of sending to the MC a copy of x , the SC simply sends the delete-request that deallocates the copy at the MC. Thus, SW_1 denotes the algorithm so optimized. Observe that SW_1 is the classic write-invalidate protocol.

5 CONNECTION COST MODEL

In this section, we analyze the algorithms in the connection cost model. The section is divided into three subsections. In the first subsection, we probabilistically study the static data allocation algorithms and, in the second, we study the family of sliding window algorithms. In each of these subsections, we derive the expected cost first, then the average expected cost, and then we compare the algorithms based on these measures. Finally, in Section 5.3, we analyze the worst case performance of all the algorithms.

5.1 Probabilistic Analysis of the Static Algorithms

For the ST_1 algorithm, a write request costs 0, and a read request cost 1 connection. For the ST_2 algorithm, every write costs 1, and every read costs 0. Hence, $EXP_{ST_1}(\theta)$ and $EXP_{ST_2}(\theta)$ are simply equal to the probabilities that a request is a read and a write, respectively. Thus,

$$EXP_{ST_1}(\theta) = 1 - \theta \text{ and } EXP_{ST_2}(\theta) = \theta. \quad (2)$$

Concerning the average expected cost, by (1) and (2) we obtain

$$AVG_{ST_1} = \int_0^1 EXP_{ST_1}(\theta) d\theta = \frac{1}{2}$$

and

$$AVG_{ST_2} = \int_0^1 EXP_{ST_2}(\theta) d\theta = \frac{1}{2}. \quad (3)$$

5.2 Probabilistic Analysis of SW_k Algorithms

In this section, we derive the expected cost of the SW_k algorithms, and we show that for each k and for each θ , the SW_k algorithm has a higher expected cost than one of the static algorithms. Then, we derive the average expected cost of the SW_k algorithms, and we show that for any k , the SW_k algorithm has a lower average expected cost than both static algorithms. Also, we show that the average expected cost of the SW_k algorithms decreases when k increases.

Recall that we are assuming that the size of the window $k (= 2 \cdot n + 1)$ is an odd number. At any point in time, the

probability that there exists a copy at the MC (which we denote by α_k) is the probability that the majority among the preceding k requests are reads, and this is the same as the probability that the number of writes in the preceding k requests is less than or equal to n , namely

$$\alpha_k = \sum_{j=0}^n \binom{k}{j} \cdot \theta^j \cdot (1 - \theta)^{k-j}. \quad (4)$$

THEOREM 1. For every k and for every θ , the expected cost of the SW_k algorithm is

$$EXP_{SW_k}(\theta) = \theta \cdot \alpha_k + (1 - \theta) \cdot (1 - \alpha_k). \quad (5)$$

PROOF. Let us consider a single request, q . When there is a copy at the MC, then the expected cost of q is equal to the probability that q is a write operation, and it equals θ . When there is no copy at the MC, the expected cost of q is $1 - \theta$. The expected cost of q is the probability that there is a copy at the MC times the expected cost of q when there is a copy at the MC, plus the probability that there is no copy at the MC times the expected cost of q when there is no copy at the MC. Thus, we conclude the theorem. \square

The next theorem compares the expected costs of the SW_k and the static algorithms.

THEOREM 2. For every k and every θ ,

$$EXP_{SW_k}(\theta) \geq \min\{EXP_{ST_1}(\theta), EXP_{ST_2}(\theta)\}.$$

PROOF. From (2), (5) it follows that

$$EXP_{SW_k}(\theta) = EXP_{ST_2} \cdot \alpha_k + EXP_{ST_1} \cdot (1 - \alpha_k).$$

The theorem follows due to the fact that the weighted average of two values is not smaller than the minimum of the two values. \square

Now, let us consider the average expected costs.

Theorem 3. For the sliding-window algorithm with window size k , SW_k , the average expected cost per request is

$$AVG_{SW_k} = \int_0^1 EXP_{SW_k}(\theta) d\theta = \frac{1}{4} + \frac{1}{4 \cdot (k + 2)}. \quad (6)$$

PROOF: Our derivation of (6) uses the following identity for positive integers a and b ,

$$\int_0^1 x^a \cdot (1 - x)^b dx = \frac{a! \cdot b!}{(a + b + 1)!}. \quad (7)$$

Using (5), it is straightforward to show that

$$AVG_{SW_k} = \frac{1}{2} + 2 \cdot \int_0^1 \alpha_k \cdot \theta d\theta - \int_0^1 \alpha_k d\theta. \quad (8)$$

Using (4) and the identity given by (7), and after some algebraic simplifications, it can be shown that

$$\int_0^1 \alpha_k \cdot \theta d\theta = \frac{(n + 1)(n + 2)}{2(k + 1)(k + 2)} \quad (9)$$

and

$$\int_0^1 \alpha_k d\theta = \frac{(n + 1)}{(k + 1)}. \quad (10)$$

Substituting for $\int_0^1 \alpha_k \cdot \theta d\theta$ and $\int_0^1 \alpha_k d\theta$ in (8), and after some simplification, we get the result given by (6). \square

COROLLARY 1. *The average expected cost of the SW_k algorithms decreases when the window size k increases, and $AVG_{SW_k} < \min\{AVG_{ST_1}, AVG_{ST_2}\}$ for any $k \geq 1$.*

PROOF. From Theorem 3, it is easy to see that AVG_{SW_k} decreases when k increases, and

$$AVG_{SW_k} \leq AVG_{SW_1} = \frac{1}{4} + \frac{1}{12} = \frac{1}{3}.$$

From (3) in Section 5.1, we conclude the corollary. \square

5.3 Worst Case Analysis in Connection Model

In this section, we show that the static algorithms, ST_1 and ST_2 , are not competitive. Then, we show that the SW_k algorithm is $(k+1)$ -competitive. Therefore, our competitiveness study suggests that for optimizing the worst case, one has to choose the sliding window algorithm with a small window size k .

First, let's consider the two static strategies. For the ST_1 algorithm, we can pick a long schedule which consists of only reads. Then, the cost of the ST_1 algorithm is unboundedly higher than the cost of the optimal algorithm on this schedule (which is zero if we keep a copy at the MC). For the ST_2 algorithm, we can also pick a long schedule which consists of only writes. Then, the cost of the ST_2 algorithm on this schedule is also unboundedly higher than the optimal cost (which is zero if we do not keep a copy at the MC). Therefore, the static algorithms, ST_1 and ST_2 , are not competitive.

THEOREM 4. *The sliding-window algorithm SW_k is tightly $(k+1)$ -competitive.*

PROOF. We prove this by showing that for any schedule ψ of requests, $COST_{SW_k}(\psi) \leq N_\psi \cdot (k+1) + (k+1)$ where N_ψ is the number of read requests in ψ that occur immediately after a write request. We will also exhibit a schedule ψ_0 for which $COST_{SW_k}(\psi_0) = N_{\psi_0} \cdot (k+1) + (k+1)$. Since it can be shown that the cost of an optimal offline algorithm on a schedule ψ is N_ψ , it follows that SW_k is tightly $(k+1)$ -competitive. As before, we assume throughout the proof that $k = 2 \cdot n + 1$.

First, we prove that

$$COST_{SW_k}(\psi) \leq N_\psi \cdot (k+1) + (k+1).$$

Let ψ be a schedule consisting of read and write requests. Let N_ψ be the number of read requests in ψ that occur immediately after a write request. We divide the schedule ψ into maximal blocks consisting of similar requests. Formally, let B_1, B_2, \dots, B_r be the division of ψ into blocks such that the requests in any block are all reads or they are all writes, and successive blocks have different requests.

It should be easy to see that the total number of read blocks in ψ , i.e., blocks that only contain read requests, is less than or equal to $(N_\psi + 1)$. Similarly, the total number of write blocks in ψ is less than or equal

to $(N_\psi + 1)$. Now, we analyze the cost of read and write requests separately. Consider any read block B_i . It should be easy to see that only the first $n+1$ reads in B_i may each incur a connection. After the first $n+1$ reads, the window will definitely have more reads than writes, and the algorithm will maintain two copies (consequently, further reads in the block do not cost any connections). Thus, the cost of executing all the reads in B_i is bounded by $(n+1)$. Hence, the cost of all the reads in ψ is bounded above by $(n+1) \cdot (N_\psi + 1)$. By a similar argument, it can be shown that the cost of all the writes in a write block is bounded by $(n+1)$. As a consequence, the cost of all the writes in ψ is bounded by $(n+1) \cdot (N_\psi + 1)$. Hence, $COST_{SW_k}(\psi) \leq 2 \cdot (n+1) \cdot (N_\psi + 1)$. Substituting $k = 2 \cdot n + 1$ and rearranging the terms, we get

$$COST_{SW_k}(\psi) \leq (k+1) \cdot N_\psi + (k+1).$$

To show that the above bound is tight, assume that initially there is a single copy of the data item. Consider a schedule ψ_0 that starts with a block of read requests, ends with a block of write requests and, in each block, there are exactly k requests. It should be easy to see that $COST_{SW_k}(\psi_0) = (k+1) \cdot N_{\psi_0} + (k+1)$. \square

6 MESSAGE COST MODEL

This section is divided into four subsections. In the first subsection, we probabilistically analyze the static algorithms; in the second, we analyze SW_1 ; and, in the third, we analyze the family of sliding window algorithms SW_k for $k > 1$.³ In each one of the first three subsections, we study the algorithm's expected cost first, then the average expected cost. We also study the relation among the expected costs of all the static and dynamic algorithms, and the relation among the average expected costs. In Section 6.4, we study the worst case of all the algorithms.

Recall that in this model, we assume that a data message cost is 1 and a control message cost is ω , where ω ranges from zero to one.

6.1 Probabilistic Analysis of the Static Algorithms

For the ST_1 algorithm, the write does not require any communication, whereas the read costs $(1 + \omega)$; for the ST_2 algorithm, every write costs 1, the read costs 0. So,

$$EXP_{ST_1}(\theta) = (1 + \omega) \cdot (1 - \theta) \quad \text{and} \quad EXP_{ST_2}(\theta) = \theta \quad (11)$$

$$AVG_{ST_1} = \int_0^1 EXP_{ST_1}(\theta) d\theta = \frac{1 + \omega}{2}$$

and

$$AVG_{ST_2} = \int_0^1 EXP_{ST_2}(\theta) d\theta = \frac{1}{2}. \quad (12)$$

3. As mentioned at the end of Section 4, SW_1 is not simply SW_k with $k = 1$. In this cost model, this difference in the algorithms results in a different analysis, thus, the need for a separate subsection dedicated to the analysis of SW_1 .

6.2 Probabilistic Analysis of the SW_1 Algorithm

First, we derive the expected cost of a relevant request.

THEOREM 5. *The expected cost of the SW_1 algorithm is*

$$EXP_{SW_1}(\theta) = \theta \cdot (1 - \theta) \cdot (1 + 2 \cdot \omega). \quad (13)$$

PROOF. For the SW_1 algorithm, a read that immediately follows a write costs $1 + \omega$ (ω for the control message that conveys the read request, and 1 for the data message); a write that immediately follows a read costs ω (the cost of a control message deallocating the copy at the MC). No other relevant requests cost any communication. Therefore, the expected cost of a request q is the expected cost of a read that immediately follows a write times the probability of q being such a read, plus the expected cost of a write that immediately follows a read times the probability of q being such a write, namely,

$$\begin{aligned} EXP_{SW_1}(\theta) &= \theta \cdot (1 - \theta) \cdot (1 + \omega) + (1 - \theta) \cdot \theta \cdot \omega \\ &= \theta \cdot (1 - \theta) \cdot (1 + 2 \cdot \omega). \end{aligned} \quad \square$$

In the next theorem, we study the relation of the expected costs of three algorithms, i.e., $EXP_{ST_1}(\theta)$, $EXP_{ST_2}(\theta)$, and $EXP_{SW_1}(\theta)$. The results of this theorem are graphically illustrated in Fig. 1.

THEOREM 6. *The expected costs $EXP_{SW_1}(\theta)$, $EXP_{ST_1}(\theta)$, and $EXP_{ST_2}(\theta)$ are related as follows, depending on θ and ω .*

1) If $\theta > \frac{1+\omega}{1+2\omega}$, then

$$EXP_{ST_1}(\theta) < EXP_{SW_1}(\theta) < EXP_{ST_2}(\theta).$$

2) If $\frac{2\omega}{1+2\omega} < \theta < \frac{1+\omega}{1+2\omega}$, then

$$EXP_{SW_1}(\theta) < \min\{EXP_{ST_1}(\theta), EXP_{ST_2}(\theta)\}.$$

3) If $\theta < \frac{2\omega}{1+2\omega}$, then

$$EXP_{ST_2}(\theta) < EXP_{SW_1}(\theta) < EXP_{ST_1}(\theta).$$

PROOF. This is a straight-forward algebraic derivation that uses (11), (13), and the fact that $\omega < 1$. \square

Now we are ready to consider the average expected cost.

Theorem 7. *The average expected cost of the SW_1 algorithm is*

$$AVG_{SW_1} = \frac{1 + 2 \cdot \omega}{6} \quad (14)$$

$$\text{and } AVG_{SW_1} \leq AVG_{ST_2} \leq AVG_{ST_1}.$$

PROOF. Equation (14) can be easily obtained from (13), based on the definition of the average expected cost

(1). Since $0 \leq \omega \leq 1$, we obtain $\frac{1+2\omega}{6} \leq \frac{1}{2} \leq \frac{1+\omega}{2}$. From (12) in Section 6.1, we conclude the theorem. \square

6.3 Probabilistic Analysis of SW_k

In this section we consider the SW_k algorithms, for $k = 2 \cdot n + 1 > 1$. First, we derive the formula of the expected cost for SW_k . Then, we show that for any k and for any θ , the expected cost of SW_k is higher than the minimum of the expected costs of SW_1 , ST_1 , and ST_2 . Thus, we conclude that

for a known fixed θ , SW_k is inferior to the other algorithms. Then, we derive the formula of the average expected cost for SW_k . Then, we show that SW_k has the best average expected cost for some $k \geq 1$, and we determine this optimal k as a function of ω , the cost of a control message.

THEOREM 8. *For every $k > 1$, the expected cost of the SW_k algorithm is*

$$\begin{aligned} EXP_{SW_k}(\theta) &= \\ &\alpha_k \cdot \theta + (1 - \alpha_k) \cdot (1 - \theta) \cdot (1 + \omega) + \\ &\omega \cdot \binom{2 \cdot n}{n} \cdot \theta^{n+1} \cdot (1 - \theta)^{n+1}. \end{aligned} \quad (15)$$

PROOF. Consider a write request w . It costs a data message if there exists a copy at the MC when the request is issued. The probability of having a copy at the MC is α_k . Additionally, if the MC deallocates its copy as a result of this write, the write will necessitate a delete message sent from the MC to the SC. It can be argued (and we omit the details) that this occurs if and only if the sequence of k requests immediately preceding w starts with a read and has exactly n writes. Therefore, the expected cost of w is

$$\alpha_k + \omega \cdot (1 - \theta) \cdot \binom{2 \cdot n}{n} \cdot \theta^n \cdot (1 - \theta)^n.$$

Now, consider a read request r . It does not require communication if there is a copy at the MC when the request is issued. Otherwise, it costs a control message for the request, and a data message for the response. Thus, the expected cost of r is $(1 - \alpha_k) \cdot (1 + \omega)$.

Therefore, the expected cost of a request is the expected cost of a write times the probability that the request is a write, plus the expected cost of a read times the probability that the request is a read, namely,

$$\begin{aligned} EXP_{SW_k}(\theta) &= \theta \cdot \left[\alpha_k + \omega \cdot \binom{2 \cdot n}{n} \cdot \theta^n \cdot (1 - \theta)^{n+1} \right] + \\ &(1 - \theta) \cdot (1 - \alpha_k) \cdot (1 + \omega). \end{aligned}$$

A simple algebraic manipulation of the above expression leads to (15). \square

THEOREM 9. *For any θ and for any $k > 1$, the expected cost of algorithm SW_k is higher than the expected cost of at least one of the algorithms SW_1 , ST_1 , and ST_2 . Namely,*

$$EXP_{SW_k}(\theta) \geq \min\{EXP_{SW_1}(\theta), EXP_{ST_1}(\theta), EXP_{ST_2}(\theta)\}.$$

In order to prove this theorem, we need the following three lemmas.

Lemma 1. *For any $k > 1$, if $\theta \leq 0.5$, then $EXP_{SW_k}(\theta) \geq EXP_{ST_2}(\theta)$.*

PROOF. From (11) and (15) we derive

$$\begin{aligned} EXP_{SW_k}(\theta) - EXP_{ST_2}(\theta) &\geq \alpha_k \cdot \theta + (1 - \alpha_k) \cdot (1 - \theta) - \theta \\ &= (1 - \alpha_k) \cdot (1 - 2 \cdot \theta) \geq 0. \end{aligned} \quad \square$$

LEMMA 2. *If $\theta > 0.5$, then α_k decreases when k increases and $1 - \theta - \alpha_k > 0$ for any $k > 1$.*

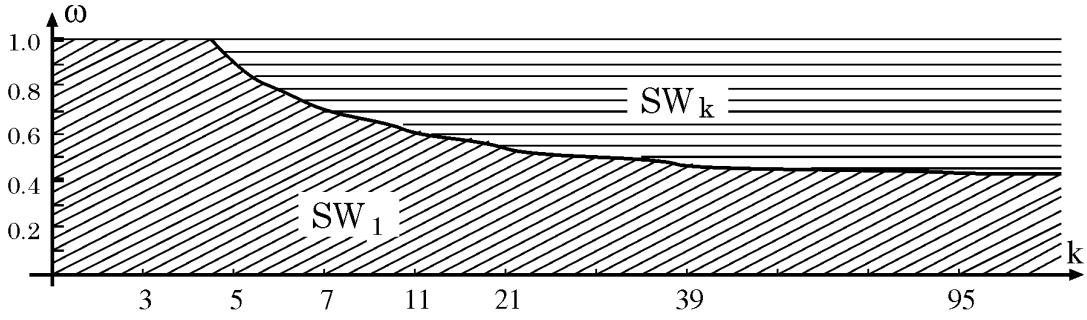


Fig. 2. Results of Corollaries 3 and 4.

PROOF. From the definition of α_k (see (4)), we can derive $1 - \theta = \alpha_1$, and

$$\alpha_{k+2} - \alpha_k = \binom{k}{n} \cdot \theta^{n+1} \cdot (1 - \theta)^{n+1} \cdot (1 - 2 \cdot \theta)$$

(details are omitted). From this formula, we see that $\alpha_{k+2} - \alpha_k$ is negative for all $k \geq 1$. Hence, α_k decreases with k . As a consequence $\alpha_1 - \alpha_k = (1 - \theta - \alpha_k) > 0$, for any $k > 1$. \square

LEMMA 3. For any $k > 1$ and any $\theta > 0.5$,

- 1) if $\omega < \frac{2\theta-1}{1-\theta}$, then $EXP_{SW_k}(\theta) > EXP_{ST_1}(\theta)$;
- 2) if $\omega \geq \frac{2\theta-1}{1-\theta}$, then $EXP_{SW_k}(\theta) > EXP_{SW_1}(\theta)$.

PROOF.

$$\begin{aligned} EXP_{SW_k}(\theta) - EXP_{ST_1}(\theta) &\geq_{(by(11)and(15))} \alpha_k \cdot \theta + \\ &(1 - \alpha_k) \cdot (1 - \theta) \cdot (1 + \omega) - (1 - \theta) \cdot (1 + \omega) = \\ &\alpha_k \cdot [2 \cdot \theta - 1 - \omega \cdot (1 - \theta)], \end{aligned}$$

namely,

$$EXP_{SW_k}(\theta) - EXP_{ST_1}(\theta) \geq \alpha_k \cdot [2 \cdot \theta - 1 - \omega \cdot (1 - \theta)].$$

Based on the above inequality, it is easy to show that if $\omega < \frac{2\theta-1}{1-\theta}$, then $EXP_{SW_k} > EXP_{ST_1}$. Thus, we have proved the first claim of the lemma.

$$\begin{aligned} EXP_{SW_k}(\theta) - EXP_{SW_1}(\theta) &\geq_{(by(13)and(15))} \alpha_k \cdot \theta + \\ &(1 - \alpha_k) \cdot (1 - \theta) \cdot (1 + \omega) - \theta \cdot (1 - \theta) \cdot (1 + 2 \cdot \omega) = \\ &\theta \cdot [\alpha_k - (1 - \theta \cdot \omega)] + (1 - \theta) \cdot (1 + \omega) \cdot (1 - \alpha_k - \theta) \geq_{(\omega \leq 1)} \\ &\theta \cdot [\alpha_k - (1 - \theta)] + (1 - \theta) \cdot (1 + \omega) \cdot (1 - \alpha_k - \theta) = \\ &(1 - \theta - \alpha_k) \cdot [1 - 2 \cdot \theta + \omega \cdot (1 - \theta)], \end{aligned}$$

namely,

$$EXP_{SW_k}(\theta) - EXP_{SW_1}(\theta) \geq (1 - \theta - \alpha_k) \cdot [1 - 2 \cdot \theta + \omega \cdot (1 - \theta)].$$

Based on the above inequality and Lemma 2, it is easy to show that if $\omega \geq \frac{2\theta-1}{1-\theta}$, then $EXP_{SW_k}(\theta) \geq EXP_{SW_1}(\theta)$. \square

PROOF OF THEOREM 9. If $\theta \leq 0.5$, then Lemma 1 indicates that $EXP_{SW_k}(\theta) \geq EXP_{ST_1}(\theta)$; if $\theta > 0.5$, then Lemma 3 indicates that $EXP_{SW_k}(\theta) \geq \min\{EXP_{SW_1}(\theta), EXP_{ST_1}(\theta)\}$. The theorem follows. \square

Now, let's consider the average expected cost of the SW_k algorithms for $k > 1$.

THEOREM 10. For the SW_k algorithm with window size $k > 1$, the average expected cost is

$$\begin{aligned} AVG_{SW_k} &= \int_0^1 EXP_{SW_k}(\theta) d\theta = \frac{1}{4} + \frac{1}{4 \cdot (k+2)} + \\ &\omega \cdot \left[\frac{1}{8} + \frac{3}{8 \cdot (k+2)} + \frac{1}{4 \cdot k \cdot (k+2)} \right]. \end{aligned} \quad (16)$$

PROOF. From the definition of α_k in Section 5.2, (7), and (15), we can derive (16). The tedious intermediate derivation steps are omitted. \square

COROLLARY 2. For $k > 1$, AVG_{SW_k} decreases when k increases, and $AVG_{SW_k} > \frac{1}{4} + \frac{\omega}{8}$.

PROOF. This corollary is straight forward from (16). \square

In Theorem 7, we have shown that the average expected cost of the SW_1 algorithm is better (i.e., lower) than that of the static algorithm. In the following corollaries, we analyze when the average expected cost of SW_k (for $k > 1$) is lower than the average expected cost of SW_1 based on (14) and (16). In Corollary 3 below, we show that when $\omega \leq 0.4$, the average expected cost of SW_k is always higher than that of SW_1 .

Corollary 3. If $\omega \leq 0.4$, then $AVG_{SW_k} > AVG_{SW_1}$ for any $k > 1$.

PROOF. If $\omega < 0.4$, then $\frac{1+2\omega}{6} < \frac{1}{4} + \frac{\omega}{8}$. Thus, by Theorem 7 and Corollary 2, we conclude this corollary. \square

In the next corollary, we study the case where $\omega > 0.4$. We show that for a given $\omega > 0.4$, there is some k_0 , such that if $k \geq k_0$, then the average expected cost of SW_k is lower than that of SW_1 . Fig. 2 illustrates the results of Corollaries 3 and 4. For example, if $\omega = 0.45$, then only when $k \geq 39$, the SW_k algorithm has a lower expected cost than that of SW_1 ; if $\omega = 0.8$, then only when $k \geq 7$, the SW_k algorithm has a lower expected cost than that of SW_1 .

Corollary 4. If $\omega > 0.4$, then $AVG_{SW_k} \leq AVG_{SW_1}$ for any k which satisfies

$$k \geq \frac{10 - \omega + \sqrt{100 - 68 \cdot \omega + 121 \cdot \omega^2}}{2 \cdot (5 \cdot \omega - 2)}.$$

PROOF. Algebraic manipulation using (14) and (16). \square

6.4 Worst Case in Message Model

In this section, we study the competitiveness of the algorithms ST_1 , ST_2 , and SW_k for $k \geq 1$ in the message cost model. The result for SW_1 is stated separately, since it is a special case (see Section 4). We conclude that the static algorithms are not competitive, as is the case in the connection model. Then, we show that SW_1 is more competitive than SW_k for $k > 1$, and we show that the competitiveness factor of the SW_k algorithms deteriorates when k increases, thus, SW_1 performs the best in the worst case.

As in the connection model, we can easily derive that the static algorithms are not competitive in the message model.

THEOREM 11. *The algorithm SW_1 is tightly $(1 + 2 \cdot \omega)$ -competitive in the message cost model, where $\omega (< 1)$ is the ratio of control message cost to data message cost.*

PROOF. Similar to the proof of Theorem 4, we let N_ψ be the number of reads in ψ that occur immediately after a write, where ψ is an arbitrary schedule of requests. It is easy to see that N_ψ is the minimum cost to satisfy all the requests in ψ . Let B_1, B_2, \dots, B_r be the division of ψ into blocks such that the requests in any block are all reads or they are all writes, and successive blocks have different requests.

It should be easy to see that the total number of read blocks in ψ is less than or equal to $(N_\psi + 1)$, and a read block costs at most $(1 + \omega)$ since, after the first read, the mobile computer will keep a copy of the data item. The total cost of reads is bounded by $(N_\psi + 1) \cdot (1 + \omega)$. Similarly, the total number of write blocks in ψ is less than or equal to $(N_\psi + 1)$, and a write block costs only ω since the first write in the block will invalidate the copy at the MC. Thus, the total cost of writes in ψ is bounded by $(N_\psi + 1) \cdot \omega$, and

$$\begin{aligned} COST_{SW_1}(\psi) &\leq (N_\psi + 1) \cdot (1 + \omega) + (N_\psi + 1) \cdot \omega \\ &= (1 + 2 \cdot \omega) \cdot N_\psi + (1 + 2 \cdot \omega). \end{aligned}$$

To show that the above bound is tight, assume that initially there is a single copy of the data item. Consider a schedule ψ_0 that starts with a read request, ends with a write request, and in each block there is exactly one request. It should be easy to see that $COST_{SW_1}(\psi_0) = (k + 1) \cdot N_\psi + (k + 1)$. \square

THEOREM 12. *The algorithm SW_k (for $k > 1$) is tightly $\left[\left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega\right]$ -competitive in the message cost model, where $\omega (< 1)$ is the ratio of control message cost to data message cost.*

PROOF. Similar to the proofs of Theorems 4 and 11, we prove that for any schedule ψ of requests,

$$\begin{aligned} COST_{SW_k}(\psi) &\leq N_\psi \cdot \left[\left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega \right] + \\ &\quad \left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega, \end{aligned}$$

where N_ψ is the number of read requests in ψ that occur immediately after a write request. We will also exhibit a schedule ψ_0 for which

$$COST_{SW_k}(\psi_0) = N_\psi \cdot \left[\left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega \right] + \gamma,$$

where γ is a constant. Since it can be shown that the cost of an optimal off-line algorithm on ψ is N_ψ , it follows that SW_k is tightly $\left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega$ -competitive. As before, we assume throughout the proof that $k = 2 \cdot n + 1$.

Let ψ be a schedule consisting of read and write requests. We prove that

$$\begin{aligned} COST_{SW_k}(\psi) &\leq N_\psi \cdot \left[\left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega \right] + \\ &\quad \left(1 + \frac{\omega}{2}\right) \cdot (k + 1) + \omega, \end{aligned}$$

as follows. We divide the schedule ψ into maximal blocks consisting of similar requests. Formally, let B_1, B_2, \dots, B_r be the division of ψ into blocks such that the requests in any block are all reads or they are all writes and successive blocks have different requests.

It should be easy to see that the total number of read blocks in ψ , i.e., blocks that only contain read requests, is less than or equal to $(N_\psi + 1)$. Similarly, the total number of write blocks in ψ is less than or equal to $(N_\psi + 1)$. Now, we analyze the cost of read and write requests separately. Consider any read block B_i . It should be easy to see that only the first $n + 1$ reads in B_i may each cost $(1 + \omega)$. After the first $n + 1$ reads, the window will definitely have more reads than writes and the algorithm will maintain two copies and further reads in the block do not cost any communication. Thus, the cost of executing all the reads in B_i is bounded by $(n + 1) \cdot (1 + \omega)$. Hence, the cost of all the reads in ψ is bounded above by $(n + 1) \cdot (1 + \omega) \cdot (N_\psi + 1)$. Now, consider a write block B_j . It is easy to see that B_j will cost at most $(n + 1)$ data message, since after the first $n + 1$ writes, the window will definitely have more writes than reads and the copy at the MC will be deallocated, and this deallocation may cost this block an additional control message. Thus, the cost of a write block is bounded by $(n + 1) + \omega$. As a consequence, the cost of all the writes in ψ is bounded by $(n + 1 + \omega) \cdot (N_\psi + 1)$. Hence,

$$COST_{SW_k}(\psi) \leq (2 + \omega) \cdot (n + 1) \cdot (N_\psi + 1) + \omega \cdot (N_\psi + 1).$$

Substituting $k = 2 \cdot n + 1$ and rearranging the terms, we get

$$\begin{aligned} COST_{SW_k}(\psi) &\leq \left[(k + 1) \cdot \left(1 + \frac{\omega}{2}\right) + \omega \right] \cdot N_\psi + \\ &\quad (k + 1) \cdot \left(1 + \frac{\omega}{2}\right) + \omega. \end{aligned}$$

To show that the above bound is tight, assume that initially there is a single copy of the data item. Consider a schedule ψ_0 that starts with a block of read requests, ends with a block of write requests, and in each block there are exactly k requests. It should be easy to see that $COST_{SW_k}(\psi_0) = (k + 1) \cdot N_\psi + (k + 1)$. \square

7 EXTENSIONS

In this section, we discuss various extensions to the previous methods. In particular, in the first subsection, we show how to modify the static algorithms to make them competitive and, in the second subsection, we discuss extensions of the algorithm to optimize the case where multiple data items can be read and written in a single operation.

7.1 Modifications to the Static Methods

We have presented two simple static methods that use the one-copy and two-copies schemes. The static methods can be chosen if the value of θ is known in advance. For example, in the connection model, the static method using a single copy at the stationary computer has the best expected cost if $\theta > 0.5$. Similarly, the static method using the two-copy scheme has the best expected cost when $\theta \leq 0.5$. However, the static methods do not have a good worst case behavior, i.e., they are not competitive. For example, a static method using a single copy will incur a high cost on a sequence of requests consisting of only reads from the mobile computer. This cost can be arbitrarily large, depending on the length of the sequence. Even though such a sequence is highly improbable, it can occur with nonzero probability.

We can overcome this problem by simple modifications to the static methods that actually make them dynamic. For example, we can modify the one-copy static method as follows. It will normally use the one-copy scheme until m consecutive reads occur; then, it changes to the two-copies scheme and uses this scheme until the next write. Then, it reverts back to one-copy scheme and repeats this process. We refer to this algorithm as $T1_m$. It can be shown that $T1_m$ is $m + 1$ -competitive and that its expected cost is $(1 - \theta) + (1 - \theta)^m (2\theta - 1)$ in the connection model. Note that the second term is the additional expected cost over the static method (it can be shown that, for each $\theta > 0.5$, $T1_m$ has a lower expected cost than SW_m and they are both equally competitive). This is the price of competitiveness. Thus, if we know that $\theta > 0.5$, then we can choose the $T1_m$ algorithm instead of ST_1 , for an appropriate value of m .

Similarly, we can modify the ST_2 algorithm to obtain the $T2_m$ algorithm that has almost the same expected cost as ST_2 , and is $(m + 1)$ -competitive.

7.2 Multiple Data Items

In this paper, we have addressed the problem of choosing an allocation method for single data items. These results can be extended to the case where multiple data items can be read and written in a single operation.

We will sketch an algorithm that gives an optimal static allocation method in the connection model for multiple data items when the frequencies of operations on the data items are known in advance. Assume that multiple data items can be remotely read in one connection; this is similar for the remote writes. We present the algorithm for the case when we have only two data items x and y . This can be generalized to more than two data items. Also, we discuss how this approach can be extended to the dynamic window based algorithms.

Assume that we have two data items x and y . We classify the read operations into three classes: reads of x only, reads

of y only, and reads that access both x and y . We assume that these three different reads occur according to independent Poisson distributions with frequencies $\lambda_{r,x}$, $\lambda_{r,y}$, and $\lambda_{r,*}$, respectively. We classify the writes similarly and assume that these writes occur with frequencies $\lambda_{w,x}$, $\lambda_{w,y}$, and $\lambda_{w,*}$, respectively. It is to be noted that $\lambda_{r,*}$ and $\lambda_{w,*}$ denote the frequencies of joint reads and writes, respectively. Now, we have four possible allocation methods for x and y : ST_1 (both x and y have only one copy), ST_2 (both x and y have two copies), $ST_{1,2}$ (x has one copy and y has one copy), and $ST_{2,1}$ (x has two copies and y has only one). For each of these allocation methods, we can obtain the expected cost of a single operation using the above frequencies and, then, choose the one with the lowest expected cost. For example, the expected cost for ST_1 is $(\lambda_{r,x} + \lambda_{r,y} + \lambda_{r,*})/\lambda$ and that of $ST_{1,2}$ is $(\lambda_{r,x} + \lambda_{w,y} + \lambda_{r,*} + \lambda_{w,*})/\lambda$, where λ is the sum of all the read and write frequencies. The above method can be generalized to any finite set of data items. We need the frequencies of various joint operations on these data items.

To use the method given above, we need to know the frequencies of various operations in advance. If these frequencies are not known in advance, then we can use the window based approach that dynamically calculates these frequencies. In this case, we need to keep track of the number of operations of different kinds (i.e., the joint/exclusive read/write operations for multiple data items) in the window. From these numbers, we can calculate the frequencies of these operations, compute their expected costs (similar to the static methods given in the previous paragraph) using these frequencies, and choose an appropriate future allocation method. To avoid excessive overhead, this recomputation can be done periodically instead of after each operation. Future work will address the performance analysis of this method.

8 COMPARISON WITH RELEVANT LITERATURE

As far as we know, this is the first paper to study the communication cost of static and dynamic allocation in distributed systems using both average case and worst case analysis. There are two bodies of relevant work, each of which is discussed in one of the following two subsections. In the first subsection, we compare this paper with database literature on data allocation in distributed systems. In the second subsection, we compare this paper to literature on caching and distributed virtual memory.

8.1 Data Allocation in Distributed Systems

Data allocation in distributed systems is either static or dynamic. In [35] and [36], we considered dynamic data allocation algorithms, and we analyzed them using the notion of convergence, which is different than the measures used in this paper, namely, expected case and worst case. Additionally, the algorithms in those works are different than the ones discussed here. Furthermore, in [35] and [36], we did not consider static allocation algorithms, and we did not consider the connection cost model.

Other dynamic data allocation algorithms were introduced in [22] and [23]. Both works analyze dynamic data allocation in the worst case only. Actually, the SW_1 algorithm was first analyzed in [23]. However, the model there

requires a minimum of two copies in the system for availability purposes. Thus, even for the worst case the results are different. In contrast, in this paper we assume that availability constraints are handled exclusively within the stationary system, independently of the mobile computers.

There has also been work addressing dynamic data allocation algorithms in [9]. This work also addresses the worst case only. Additionally, the model there does not allow concurrent requests, and it requires centralized decision making by a processor that is aware of all the requests in the network. In contrast, our algorithms are distributed, and allow concurrent read-write requests.

Static allocation was studied in [37], [14]. These works address the following file-allocation problem. They assume that the read-write pattern at each processor is known a priori or it can be estimated, and they find the optimal static allocation scheme. However, works on the file-allocation problem do not compare static and dynamic allocation, and they do not quantify the cost penalty if the read-write pattern deviates from the estimate.

Many works on the data replication problem (such as [4], [6], [11], [13], [17], [21], [34]) and on file systems (such as CODA [30], [33]) solely address the availability aspect, namely, how to ensure availability of a data item in the presence of failures. In contrast, in this paper we addressed the communication cost issue.

The works done by Alonso and Ganguly in [5], [20] are also related to the present paper in the sense that they also address the optimization issue for mobile computers. However, their optimization objective is energy, whereas ours is communication.

The work on broadcast disks ([2]) also addresses performance issues related to push/pull of data in a mobile computing environment. However, this work assumes read-only data items and it does not perform the type of analytical performance evaluation present in this paper.

8.2 Caching and Virtual Memory

In the computer architecture and operating systems literature, there are studies of two subjects related to dynamic data allocation, namely, caching and distributed virtual memory (see [1], [3], [7], [8], [12], [15], [16], [19], [25], [29], [30], [27], [28], [31], [33], [38]).

However, there are several important differences between Caching and Distributed Virtual Memory (CDVM) on one hand, and replicated data in distributed systems on the other. Therefore, our results have not been obtained previously. First, many of the CDVM methods do not focus on the communication cost alone, but consider the collection of factors that determine performance; the complexity of the resulting problem dictates that their analysis is either experimental or it uses simulation. In contrast, in this paper we assumed that since, in mobile computing wireless communication involves an immediate out-of-pocket expense, the optimization of wireless communication is the sole caching objective, and we performed a thorough analytical cost evaluation.

Second, in CDVM, the size of the cache is assumed to be limited. Thus, the important issues in CDVM literature are cache utilization, and the page replacement strategy (e.g., LRU, MRU, etc.), namely, which page to replace in the

cache when the cache is full and a new page has to be brought in. In other words, in contrast to replicated data in distributed systems, which may reside on secondary and even tertiary storage, in CDVM a page may be deleted from the cache as a result of limited storage. One may argue whether or not limited storage is a major issue in distributed databases, however, in this paper we assumed that storage at the mobile computer is abundant.

There have been some CDVM methods which consider communication cost as one of the optimization criteria (e.g., TreadMarks [26]). However, they do not use dynamic allocation schemes.

9 CONCLUSIONS

In this paper, we have considered several data allocation algorithms for mobile computers. In particular, we have considered the one-copy and the two-copies allocation schemes. We have investigated static and dynamic allocation methods using the above schemes. In a static method, the allocation scheme remains unchanged throughout the execution. In a dynamic method, the allocation scheme changes dynamically based on a window consisting of the last k requests; if, in the window, there are more reads at the mobile computer than writes at the stationary computer, then we use the two-copy scheme; otherwise, we use the one-copy scheme. We get different dynamic methods for different values of k . For $k = 1$, the dynamic method is simply the classic write-invalidate protocol.

We have considered two cost models—the connection model and the message model. In the connection model, the cost is measured in terms of the number of (wireless telephone) connections, whereas, in the message model, the cost is measured in terms of the number of control and data messages.

We have considered three different measures—expected cost, average expected cost, and the worst case cost which uses the notion of competitiveness. Roughly speaking, an algorithm A is said to be k -competitive if, for every sequence s of read-write requests, the cost of A on the sequence s is at most k times the cost of an ideal off-line algorithm on s which knows s in advance. An algorithm A is said to be competitive if, for some $k > 0$, A is k -competitive. The expected cost is the standard expected cost per request assuming fixed probabilistic distributions for reads and writes. We believe that an allocation method should be chosen based on the expected cost as well as the worst case cost. Specifically, we think that an allocation method should be chosen to minimize the expected cost, provided that it has some bound on the worst case behavior.

Now we explain the difference between the expected cost and the average expected cost. We have assumed that both reads at the mobile computer and writes at the stationary computer occur according to independent Poisson distributions with frequencies λ_r and λ_w , respectively. When the values of λ_r and λ_w are known (more specifically, when the value of $\theta = \frac{\lambda_w}{\lambda_r + \lambda_w}$ is known), then an allocation method should be chosen based on the expected cost and the competitiveness. However, when θ varies and it is

equally likely to have any value between 0 and 1, then an allocation method should be chosen based on the average expected cost (in addition to competitiveness). The average expected cost is the integral of the expected cost over θ from zero to one. An allocation method with a lower average expected cost will have a lower average cost per request in a sequence of requests in which the frequencies of reads and writes vary over time. Furthermore, the average expected cost can also provide an insight and/or a measure for selecting an allocation method in the case when θ is unknown, but it is equally likely to have any value between zero and one.⁴

In the connection model, if θ is greater than 0.5, i.e., the read frequency is lower than the write frequency, then the static allocation method using only one copy at the stationary computer has the best expected cost. Similarly, if θ is smaller than 0.5, then the static allocation method using one copy at the stationary computer and one at the mobile computer has the best expected cost.

When λ_r and λ_w change over time (i.e., θ changes over time), then one of the dynamic methods SW_k for an appropriate value of k should be chosen. This is due to the fact that the average expected cost of the SW_k algorithms is lower than either one of the static methods. The value of the window size k should be chosen to strike a balance between the average expected cost (which decreases as k increases, see (6)) and competitiveness (the SW_k algorithm is $(k + 1)$ -competitive, thus, competitiveness becomes worse as k increases). For example, for $k = 9$, the sliding-window algorithm will have an average expected cost that is within 10 percent of the optimum and, in the worst case, will be at most 10 times worse than the optimum offline algorithm.

In the message model, the static allocation methods are still not competitive and the dynamic allocation methods SW_k are again competitive, although with a different competitiveness factor. For a given θ , the expected cost of one of the three methods ST_1 , ST_2 , and SW_1 is lowest; the particular one depends on the values of θ and ω (the ratio between the control message cost and the data message cost). The lowest expected-cost algorithm as a function of θ and ω is given in Fig. 1.

If θ is unknown or it varies over time, then one of the sliding window methods provides the optimal average expected cost. The particular one depends on the value of ω . If $\omega \leq 0.4$, then the SW_1 algorithm should be chosen as it has the least average expected cost; for other values of ω , the higher the value of k , the lower the average expected cost of the SW_k algorithm (see Fig. 2). Again, the appropriate value of k should be chosen to strike a balance between average expected cost and competitiveness.

The data allocation methods and the results of this paper pertain to applications where the data items accessed by the various mobile computers are mostly disjoint, and the read requests must be satisfied by the most-up-to-date version of the data item. For applications that do not satisfy these assumptions, techniques that use data broadcasting and batching of updates may be appropriate, and our results need to be extended. This is the subject of future work.

4. If θ is not uniformly distributed, then the average expected cost should be defined as the integral of the expected cost multiplied by the density function for θ .

ACKNOWLEDGMENTS

We wish to thank the referees for their insightful comments. A preliminary version of this paper appeared in the Proceedings of the ACM-SIGMOD, May, 1994, Minneapolis, Minnesota. This research was supported in part by U.S. National Science Foundation Grants IRI-9224605, IRI-9408750, IRI-9712967, by DARPA Grant N66001-97-2-8901, by NATO Grant CRG-960648, and by AFSOR Grant F49620-93-1-0059.

REFERENCES

- [1] J. Archibald and J. Baer, "An Evaluation of Cache Coherence Solutions in Shared-Bus Multiprocessors," *ACM Trans. Computer Systems*, vol. 4, no. 4, pp. 273-298, Nov. 1986.
- [2] S. Acharya, M. Franklin, and S. Zdonik, "Balancing Push and Pull for Data Broadcast," *Proc. ACM-SIGMOD '97*, pp. 183-194, 1997.
- [3] A. Agarwal, R. Simoni, J. Hennessy, and M. Horowitz, "An Evaluation of Directory Schemes for Cache Coherence," *Proc. 15th Int'l Symp. Computer Architecture*, pp. 280-289, June 1988.
- [4] M. Ahamad and M.H. Ammar, "Multidimensional Voting," *ACM Trans. Computer Systems*, vol. 9, no. 4, pp. 399-431, 1991.
- [5] R. Alonso and S. Ganguly, "Query Optimization for Energy Efficiency in Mobile Environments," *Proc. 1993 Int'l Workshop Foundations of Models and Languages for Data and Objects*, Aigen, Austria, 1993.
- [6] D. Agrawal and A. El Abbadi, "The Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data," *Proc. 16th Very Large Data Bases*, Aug. 1990.
- [7] J.K. Bennett, J.B. Carter, and W. Zwaenepoel, "Adaptive Software Cache Management for Distributed Shared Memory Architectures," *Proc. 17th Int'l Symp. Computer Architecture*, pp. 148-159, May 1990.
- [8] J.K. Bennett, J.B. Carter, and W. Zwaenepoel, "Munin: Distributed Shared Memory Based on Type-Specific Memory Coherence," *Proc. 1990 Conf. Principles and Practice of Parallel Programming*, 1990.
- [9] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive Algorithms for Distributed Data Management," *Proc. 24th Ann. ACM STOC*, Victoria, B.C., Canada, May 1992.
- [10] B.R. Badrinath and T. Imielinski, "Replication and Mobility," *Proc. Second Workshop Management of Replicated Data (WMRD-II)*, pp. 9-12, Monterey, Calif., 1992.
- [11] S.Y. Cheung, M.H. Ammar, and M. Ahamad, "The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data," *Proceedings Sixth Int'l Conf. Data Eng.*, pp. 438-445, Jan. 1990.
- [12] M.J. Carey, M.J. Franklin, M. Livny, and E.J. Shekita, "Data Caching Tradeoffs in Client-Server BDMS Architectures," *Proc. ACM-SIGMOD '91*, pp. 357-366, 1991.
- [13] M.J. Carey and Miron Livny, "Distributed Concurrency Control Performance: A Study of Algorithms, Distribution and Replication," *Proc. 14th Very Large Data Bases Conf.*, Los Angeles, 1988.
- [14] L.W. Dowdy and D.V. Foster, "Comparative Models of the File Assignment Problem," *ACM Computing Surveys*, vol. 14, no. 2, 1982.
- [15] S.J. Eggers and R.H. Katz, "A Characterization of Sharing in Parallel Programs and Its Application to Coherency Protocol Evaluation," *Proc. 15th Int'l Symp. Computer Architecture*, pp. 373-382, June 1988.
- [16] S.J. Eggers and R.H. Katz, "Evaluating the Performance of Four Snooping Cache Coherency Protocols," *Proc. 16th Int'l Symp. Computer Architecture*, June 1989.
- [17] D.L. Eager, and K.C. Sevick, "Achieving Robustness in Distributed Database Systems," *ACM Trans. Database Systems*, vol. 8, no. 3, pp. 354-381, Sept. 1983.
- [18] A. Fiat, R. Karp, M. Luby, L.A. McGeoch, D. Sleator, N.E. Yong, "Competitive Paging Algorithms," *J. Algorithms*, vol. 12, pp. 685-699, 1991.
- [19] M. Franklin, *Client Data Caching: A Foundation for High Performance Object Database Systems*. Kluwer Academic Publishing, 1996.
- [20] S. Ganguly and R. Alonso, "Query Optimization in Mobile Environments," technical report, Rutgers Univ., Dec. 1993.
- [21] D. Gifford, "Weighted Voting for Replicated Data," *Proc. Seventh ACM Symp. Operating System Principles*, pp. 150-162, 1979.

- [22] Y. Huang and O. Wolfson, "A Competitive Dynamic Data Replication Algorithm," *IEEE Proc. Ninth Int'l Conf. Data Eng.* '93, pp. 310-317, Vienna, Austria, 1993.
- [23] Y. Huang and O. Wolfson, "Dynamic Allocation in Distributed System and Mobile Computers," *IEEE Proc. Tenth Int'l Conf. Data Eng.* '94, pp. 20-29, Houston, Tex., 1994.
- [24] T. Imielinski and B.R. Badrinath, "Querying in Highly Mobile Distributed Environments," *Proc. 18th Int'l Conf. Very Large Data Bases '92*, pp. 41-52, 1992.
- [25] K. Li, "Shared Virtual Memory on Loosely Coupled Multiprocessors," PhD thesis, Dept. of Computer Science, Yale Univ., Sept. 1986.
- [26] P. Keleher, S. Dwarkadas, A.L. Cox, and W. Zwaenepoel, "TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems," *Proc. Winter 94 Usenix Conf.*, pp. 115-131, Jan. 1994.
- [27] K. Li and P. Hudak, "Memory Coherence in Shared Virtual Memory Systems," *ACM Trans. Computer Systems*, vol. 7, no. 4, pp. 321-359, Nov. 1989.
- [28] J. Lee and U. Ramachandram, "Synchronization with Multiprocessor Caches," *Proc. 17th Int'l Symp. Computer Architecture*, pp. 27-37, May 1990.
- [29] A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator, "Competitive Snoopy Caching," *Algorithmica*, vol. 3, pp. 79-119, 1988.
- [30] J.J. Kistler, and M. Satyanarayanan, "Disconnected Operation in the Coda File System," *ACM Trans. Computer Systems*, vol. 10, no. 1, pp. 3-25, Feb. 1992.
- [31] D.J. Makaroff and D.L. Eager, "Disk Cache Performance for Distributed Systems" *Proc. 10th Int'l Conf. Distributed Computing Systems*, pp. 212-219, May 1990.
- [32] M. Manasse, L.A. McGeoch, and D. Sleator, "Competitive Algorithms for Online Problems," *Proc. 20th ACM STOC*, pp. 322-333, 1988.
- [33] M. Satyanarayanan, J.J. Kistler, P. Kumar, M.E. Okasaki, E.H. Siegel, and D.C. Steere, "Coda: A Highly Available File System for a Distributed Workstation Environment," *IEEE Trans. Computers*, vol. 39, no. 4, pp. 447-459, Apr. 1990.
- [34] R.H. Thomas, "A Majority Consensus Approach to Concurrency Control for Multiple Copy Database," *ACM Trans. Database Systems*, vol. 4, no. 2, pp. 180-209, June 1979.
- [35] O. Wolfson and S. Jajodia, "Distributed Algorithms for Dynamic Replication of Data," *Proc. ACM Principles of Database Systems*, 1992.
- [36] O. Wolfson and S. Jajodia, "An Algorithm for Dynamic Data Distribution," *Proc. Second Workshop Management of Replicated Data (WMRD-II)*, pp. 62-65, 1992.
- [37] O. Wolfson and A. Milo, "The Multicast Policy and Its Relationship to Replicated Data Placement," *ACM Trans. Database Systems*, vol. 16, no. 1, 1991.
- [38] Y. Wang, L.A. Rowe, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture," *Proc. ACM SIGMOD '91*, pp. 367-376, 1991.



A. Prasad Sistla obtained the PhD degree in computer science/applied mathematics from Harvard University in 1983. Prior to that, he obtained the ME degree in computer science from the Indian Institute of Science, Bangalore, India. He is currently an associate professor in the Department of Electrical Engineering and Computer Science at the University of Illinois at Chicago. Prof. Sistla has done extensive research in the areas of analysis and verification of concurrent systems, formal method for concurrent and distributed systems, and also in active and multimedia database systems. His current research interests include all of the above areas. Prof. Sistla has published extensively in leading computer science journals and conferences. He has served on the program committees of important computer science conferences. Prof. Sistla's research has been funded by leading organizations, such as the U.S. National Science Foundation, AFOSR, DARPA, etc. He is a consultant for Bell Labs/Lucent Technologies.



Ouri Wolfson received his BA degree in mathematics and his PhD degree in computer science from the Courant Institute of Mathematical Sciences, New York University, in 1984. He is currently an associate professor in the Department of Electrical Engineering and Computer Science at the University of Illinois at Chicago, where he directs the Distributed Computing and Databases Laboratory. He has consulted for Argonne National Laboratory, for the U.S. Army Research Laboratories, and for the Center of

Excellence in Space Data and Information Sciences at NASA. Before joining the University of Illinois, he was a research faculty member at Columbia University, and prior to that he was a systems analyst at Bell Laboratories and at American Broadcasting Co.

Dr. Wolfson has authored more than sixty publications in leading journals and conference proceedings. He is an editor of the *ACM/URSI/Baltzer Wireless Networks Journal*, and a guest editor of the *ACM/Baltzer Journal on Special Topics in Mobile Networks*. From 1991 to 1996, he served as a national lecturer for the ACM professional society. He participated in numerous conferences as a program committee member, speaker, session chairman, and panelist. Most recently, he was the program committee cochair of the First International Workshop on Satellite-Based Information Systems (Wosbis), and the general cochair of the second international workshop on the same topic. His work has been funded by the U.S. National Science Foundation, the Air Force Office of Scientific Research, Defense Advanced Research Projects Agency, NATO, the New York State Science and Technology Foundation, Hughes Research Laboratories, and Informix Co.



Yixiu Huang received his Bachelor's degree in mathematics from Hunan University, China, in 1983. In 1986, he obtained a master's degree in applied mathematics from the Beijing Institute of Technology, Beijing, China. After graduation, he taught mathematics at Chongqing Institute of Technology and Management from 1986 to 1988. Dr. Huang obtained his PhD degree in computer science in 1995 from the University of Illinois at Chicago. Currently, Dr. Huang is working as a senior systems engineer on web development for Nations Bank.