New On-Line Algorithms for the Page Replication Problem[*]

Susanne Albers[†] Hisashi Koga[‡]

Abstract

We present improved competitive on-line algorithms for the page replication problem and concentrate on important network topologies for which algorithms with a constant competitive ratio can be given. We develop an optimal randomized on-line replication algorithm for trees and uniform networks; its competitive ratio is approximately 1.58. This performance holds against oblivious adversaries. We also give a randomized memoryless replication algorithm for trees and uniform networks that is 2-competitive against adaptive on-line adversaries. Furthermore we consider on-line replication algorithms for rings and present general techniques that transform c-competitive algorithms for trees into $2c$-competitive algorithms for rings. As a result we obtain a randomized on-line algorithm for rings that is 3.16-competitive. We also derive two 4-competitive on-line algorithms for rings which are either deterministic or randomized and memoryless. Again, the randomized results hold against oblivious adversaries. Apart from these techniques, we finally give a randomized memoryless replication algorithm for rings that is 4-competitive against adaptive on-line adversaries.

1 Introduction

This paper deals with problems that arise in the memory management of large multiprocessor systems. Such multiprocessing environments typically consist of a network of processors, each of which has its local memory. A global shared memory is modeled by distributing the physical pages among the local memories. Accesses to the global memory are accomplished by accessing the local memories. Suppose a processor p wants to read a memory address from page B. If B is stored in p's local memory, then this read operation can be accomplished locally. Otherwise, p determines a processor q holding the page and sends a request to q. The desired information is then transmitted from q to p, and the communication cost incurred thereby is proportional to the distance from q to p. If p has to access page B frequently, it may be worthwhile to move or

[*]This paper combines and extends results of two conference papers. First paper: H. Koga. Randomized on-line algorithms for the page replication problem. In *Proc. 4th International Annual Symposium on Algorithms and Computation (ISAAC93)*. Second paper: S. Albers and H. Koga. New on-line algorithms for the page replication problem. In *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT94)*.

[†]Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. Work was supported in part by the ESPRIT Basic Research Actions of the EU under contract No. 7141 (project ALCOM II). E-mail: albers@mpi-sb.mpg.de

[‡]Fujitsu Laboratories Limited., 211-88 Kawasaki, Japan. This work was done while the second author was a student in the Department of Information Science, University of Tokyo. E-mail: koga@flab.fujitsu.co.jp

copy B from q to p because subsequent accesses will become cheaper. However, transmitting an entire page incurs a high communication cost proportional to the page size times the distance from q to p.

If a page is writable, it is reasonable to store only one copy of the page in the entire system. This avoids the problem of keeping multiple copies of the page consistent. The *migration problem* is to decide in which local memory the single copy of the writable page should be stored so that a sequence of memory accesses can be processed at low cost. On the other hand, if a page is read-only, it is possible to keep several copies of the page in the system, i.e., a page may be copied from one local memory to another. In the *replication problem* we have to determine which local memories should contain copies of the read-only page. Finding efficient migration and replication strategies is an important problem that has been studied from a practical and theoretical point of view [2, 4, 6, 7, 8, 11, 13]. In this paper we study on-line algorithms for the page replication problem. We analyze the performance of on-line algorithms using *competitive analysis* [12], the worst case ratio of the cost incurred by an on-line algorithm and the cost incurred by an optimal off-line algorithm.

Awerbuch *et al.* [2] presented a deterministic on-line replication strategy for general graphs that achieves an optimal competitive ratio of $\Theta(\log n)$, where n is the number of processors. However, for many important topologies, this bound is not very expressive. Black and Sleator [6], who initiated the theoretical study of the replication problem, proposed a 2-competitive deterministic on-line algorithm for trees and uniform networks. A uniform network is a complete graph in which all edges have the same length. Black and Sleator also proved that no deterministic on-line replication algorithm can be better than 2-competitive. Bartal *et al.* [4] presented a randomized $2(2+\sqrt{3})$-competitive replication algorithm against adaptive on-line adversaries for the case that the network topology forms a ring. We note that $2(2+\sqrt{3}) \approx 7.5$. Using the $2(2+\sqrt{3})$-competitive algorithm by Bartal *et al.*, one can construct a deterministic replication algorithm for the ring that achieves a competitive ratio of $2^2(2+\sqrt{3})^2 \approx 55.7$, see [5]. However, that algorithm is very complicated and not useful in practical applications.

In this paper we develop a number of new deterministic and randomized on-line replication algorithms. We concentrate on network topologies that are important in practice and for which on-line algorithms with a constant competitive ratio can be developed. In Section 4.1 we present a randomized on-line replication algorithm for trees and uniform networks, called GEOMETRIC, which is $(\frac{\rho^r}{\rho^r-1})$-competitive against oblivious adversaries. Here $\rho = \frac{r+1}{r}$ and r is the page size factor. For large values of r, which occur in practice, GEOMETRIC's competitiveness is approximately $\frac{e}{e-1} \approx 1.58$. We also show that GEOMETRIC is optimal. Specifically we prove that no randomized on-line replication algorithm can be better than $(\frac{\rho^r}{\rho^r-1})$-competitive against oblivious adversaries. Interestingly, our algorithm GEOMETRIC uses only one random number during an initialization phase and runs completely deterministically thereafter. Lund *et al.* [9] have independently developed the same results for trees and uniform networks using a different approach. Moreover, we give a randomized memoryless on-line replication algorithm for trees and uniform networks that is 2-competitive against adaptive on-line adversaries. This is the best competitiveness that can be achieved against adaptive on-line adversaries.

In Section 5 we consider replication algorithms for rings. We present a deterministic technique that transforms c-competitive algorithms for trees into $2c$-competitive algorithms for rings.

Combining this technique with the algorithm GEOMETRIC, we obtain a randomized algorithm for rings that achieves a competitive ratio of $(\frac{2\rho^r}{\rho^r-1}) \approx 3.16$. We also derive two 4-competitive algorithms for rings which are either deterministic or randomized and memoryless. The randomized performances hold against oblivious adversaries. Our 4-competitive deterministic algorithm is very simple and greatly improves the competitive ratio of 55.7 mentioned above. We also present a randomized version of our deterministic technique for constructing ring algorithms; this variant achieves the same performance. Finally, using an approach different from the above technique, we develop a randomized memoryless replication algorithm for rings that is 4-competitive against adaptive on-line adversaries.

2 Problem statement and competitive analysis

Formally, the page replication problem can be described as follows. We are given an undirected graph G. Each node in G corresponds to a processor and the edges represent the interconnection network. Associated with each edge is a *length* that is equal to the distance between the connected processors. We assume that the edge lengths satisfy the triangle inequality. In the page replication problem we generally concentrate on one particular page. We say that *a node v has the page* if the page is contained in v's local memory. *A request at a node v* occurs if v wants to read an address from the page. The request can be satisfied at zero cost if v has the page. Otherwise the request is served by accessing a node w holding the page and the incurred cost equals the distance from v to w. After the request is satisfied, the page may be replicated from node w to any other node v' that does not hold the page (node v' may coincide with node v). The cost incurred by this replication is r times the distance from w to v'. Here r denotes the page size factor. In practical applications, r is a large value, usually several hundred or thousand. (The page may only be replicated *after* a request because it is impossible to delay the service of the memory access while the entire page is copied.) We study the page replication problem under the assumption that a node having the page never drops it. A page replication algorithm is usually presented with an entire sequence of requests that must be served with low total cost. A page replication algorithm is *on-line* if it serves every request without knowledge of any future requests.

We analyze the performance of on-line page replication algorithms using *competitive analysis* [12]. In a competitive analysis, the cost incurred by an on-line algorithm is compared to the cost incurred by an *optimal off-line algorithm.* An optimal off-line algorithm knows the entire request sequence in advance and can serve it with minimum cost. Let $C_A(\sigma)$ and $C_{OPT}(\sigma)$ be the cost of the on-line algorithm A and the optimal off-line algorithm OPT on a request sequence σ. Usually an on-line algorithm is called c-competitive if there exists a constant a such that $C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + a$ holds for every request sequence. Note, however, that if the constant a depends on r and the number of processors in the network, then an on-line replication algorithm can be 0-competitive by replicating the page initially to all processors and assigning the total cost of initial replications to a. On the other hand, if a does not depend on r, an additive constant cannot reduce the competitiveness of an on-line replication algorithm because r can be large relative to the cost of serving a number of accesses. Therefore, we use a stronger definition.

We call an on-line replication algorithm c-competitive if

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma)$$

for all request sequences σ. If A is a randomized algorithm, then $C_A(\sigma)$ must be replaced by the expected cost incurred by A, where the expectation is taken over the random choices made by A. In this paper we evaluate randomized on-line algorithms only against *oblivious* and *adaptive on-line adversaries*, see [5] for details. An oblivious adversary has to generate a request sequence in advance and is not allowed to see the random choices made by the on-line algorithm. An adaptive on-line adversary may see the random choices made by the on-line algorithm, i.e., when generating a new request the adversary can see all the on-line algorithm's random choices on past requests. However, an adaptive on-line adversary also has to serve the request sequence on-line.

3 Basic definitions and techniques

Before describing our new algorithms in the following sections, we introduce some basic definitions for trees. These will be useful throughout the paper, since, even when considering uniform networks or rings, we will often reduce the algorithms and their analyses to the case that the underlying topology forms a tree.

The root of the given tree is generally denoted by s. We assume that initially, only s has the page. Consider an undirected edge $e = \{v, w\}$ in the tree. The node in $\{v, w\}$ that is farther away from the root is called the *child node* of e. The length of e is denoted by $l(e)$. Given two nodes u and v in the tree, let $l(u, v)$ denote the length of the (unique) path from u to v.

In the following we will always assume that if an algorithm (on-line or off-line) replicates the page from a node w to a node v, then the page is also replicated to all nodes on the path from w to v. This does not incur extra cost. Thus, the nodes with the page always form a connected component of the given tree. Note that if a node v does not have the page, then the closest node w with the page lies on the path from v to the root, and all paths from v to a node with the page pass through w. Therefore, we may assume without loss of generality that a replication algorithm always serves requests at a node not holding the page by accessing the closest node with the page. This cannot increase the total cost incurred in serving the whole request sequence.

We present a technique that we will frequently use to analyze on-line replication algorithms for trees. Let T be a tree and σ be a request sequence for T. We usually analyze an on-line replication algorithm A by partitioning the costs that are incurred by A and by OPT into parts that are incurred by each edge of the tree. Suppose an algorithm serves a request at a node v. Then an edge e incurs a cost equal to the length of e if e belongs to the path from v to the closest node with the page. If e does not belong to that path, then e incurs a cost of zero. An edge also incurs the cost of a replication across it. Let $C_A(\sigma, e)$ denote the cost that is incurred by edge e when A serves σ. Analogously, let $C_{OPT}(\sigma, e)$ be the cost that is incurred by e when OPT serves σ. (If A is a randomized algorithm, then $C_A(\sigma, e)$ is the expected cost incurred by e.) The performance of an on-line algorithm A is generally evaluated by comparing $C_A(\sigma, e)$ to

$C_{OPT}(\sigma, e)$ for all edges e of the tree. In order to analyze $C_A(\sigma, e)$, we introduce some notation. Let $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$ be a request sequence of length m and let $\sigma(t)$, $1 \leq t \leq m$, be the request at time t. Suppose $\sigma(t)$ is a request at node v. We set

$$a_\sigma(e, t) = 1$$

if e belongs to the path from v to the root. Otherwise we set

$$a_\sigma(e, t) = 0.$$

If $a_\sigma(e, t) = 1$, we say that $\sigma(t)$ *causes an access at edge* e. Let

$$a_\sigma(e) = \sum_{t=1}^{m} a_\sigma(e, t),$$

i.e., $a_\sigma(e)$ is the number of requests that cause an access at edge e. The following simple lemma is crucial in our analyses.

Lemma 1 *Let A be an on-line replication algorithm that, given an arbitrary tree T and a request sequence σ for T, satisfies*

$$C_A(\sigma, e) \leq c \cdot \min\{a_\sigma(e), r\} \cdot l(e) \tag{1}$$

for all edges e. Then the algorithm A is c-competitive. (If A is a randomized algorithm, then $C_A(\sigma, e)$ is the expected cost incurred by e and the competitive ratio of c holds against any oblivious adversary.)

Proof: We prove that for any edge e, $C_{OPT}(\sigma, e) = \min\{a_\sigma(e), r\} \cdot l(e)$. By Eq. (1), this implies $C_A(\sigma, e) \leq c \cdot C_{OPT}(\sigma, e)$ for all edges e, and hence A is c-competitive. If $a_\sigma(e) < r$, then OPT does not replicate the page across e and e incurs a cost of $a_\sigma(e) l(e)$. Hence $C_{OPT}(\sigma, e) = a_\sigma(e) \cdot l(e) = \min\{a_\sigma(e), r\} l(e)$. On the other hand, if $a_\sigma(e) \geq r$, then OPT replicates the page across e immediately, before serving any requests, and e incurs a cost of $r l(e)$. Thus $C_{OPT}(\sigma, e) = r \cdot l(e) = \min\{a_\sigma(e), r\} l(e)$. □

4 Algorithms for trees and uniform networks

First, in Section 4.1, we describe and analyze two randomized on-line algorithms for trees. The first of these algorithms achieves an optimal competitive ratio against any oblivious adversary. We also give an algorithm that is competitive against any adaptive on-line adversary. In Section 4.2 we demonstrate that both our algorithms can be easily applied to uniform networks, while maintaining their competitive performance. Throughout this section let $\rho = \frac{r+1}{r}$.

4.1 Trees

Algorithm GEOMETRIC (for trees): The algorithm first chooses a random number from the set $\{1, 2, \ldots, r\}$. Specifically, the number i is chosen with probability $p_i = \alpha \cdot \rho^{i-1}$, where $\alpha = \frac{\rho - 1}{\rho^r - 1}$. While processing the request sequence, the algorithm maintains a count on each edge of the tree. Initially, all counts are set to 0. If there is a request at a node v that does not have the page, then all counts along the path from v to the closest node with the page are incremented by 1. When a count reaches the value of the randomly chosen number, the page is replicated to the child node of the corresponding edge.

Before we analyze the performance of GEOMETRIC, we mention a few observations and remarks. The algorithm is called GEOMETRIC because $p_{i+1}/p_i = \rho$ is constant for all $i = 1, 2, \ldots, r-1$. It is easy to verify that $\sum_{i=1}^{r} p_i = 1$. Suppose that GEOMETRIC processes a request sequence σ. We can easily prove by induction on the number of requests processed so far that the counts on a path from the root to a node v are always monotonically non-increasing. Furthermore, after each request, a node (except for the root s) has the page if and only if it is the child node of an edge whose count is equal to the value of the randomly chosen number.

Theorem 1 *For any tree, the algorithm GEOMETRIC is* $(\frac{\rho^r}{\rho^r-1})$*-competitive against any oblivious adversary.*

Note that $\frac{\rho^r}{\rho^r-1}$ goes to $\frac{e}{e-1} \approx 1.58$ as r tends to infinity. Furthermore, GEOMETRIC uses only one random number during an initialization phase and runs completely deterministically thereafter.

Proof: Consider an arbitrary tree T and a request sequence σ for T. Let e be an edge of the tree and let $E[C_G(\sigma, e)]$ denote the expected cost incurred by edge e when GEOMETRIC serves σ. We will show that

$$E[C_G(\sigma, e)] \leq (\frac{\rho^r}{\rho^r - 1}) \cdot \min\{a_\sigma(e), r\} \cdot l(e) \tag{2}$$

for any edge e of T. Lemma 1 implies the theorem.

Let $k = a_\sigma(e)$ and $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k)$ be the requests in σ that cause an access at the edge e. Note that the algorithm GEOMETRIC increases the count of e exactly at the requests $\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_k)$, provided that the page has not been replicated across e so far.

First, assume that $k > r$. Since $\sum_{i=1}^{r} p_i = 1$, GEOMETRIC has replicated the page across e before the request $\sigma(t_{r+1})$. Thus the edge e incurs the same cost as if we had $k = r$. For this reason it suffices to consider the case that k satisfies $1 \leq k \leq r$ and show $E[C_G(\sigma, e)] \leq c \cdot k \cdot l(e)$, where $c = \frac{\rho^r}{\rho^r-1}$. This proves (2).

So suppose we have $1 \leq k \leq r$. The algorithm GEOMETRIC first chooses a random number i from the set $\{1, 2, \ldots, r\}$. If i satisfies $i \leq k$, the edge e incurs a cost of $r + i$. Otherwise e incurs a cost of k. Thus

$$\begin{aligned}
E[C_G(\sigma, e)] &= l(e)(\sum_{i=1}^{k}(r+i)p_i + \sum_{i=k+1}^{r} kp_i) \\
&= l(e)(\sum_{i=1}^{k} r\alpha\rho^{i-1} + \sum_{i=1}^{k} i\alpha\rho^{i-1} + \sum_{i=k+1}^{r} k\alpha\rho^{i-1}) \\
&= \alpha l(e)(\frac{r(\rho^k - 1)}{\rho - 1} + \frac{k\rho^{k+1} - (k+1)\rho^k + 1}{(\rho-1)^2} + \frac{k(\rho^r - \rho^k)}{\rho - 1}).
\end{aligned}$$

We have $\rho - 1 = \frac{1}{r}$. Thus

$$\begin{aligned} E[C_G(\sigma, e)] &= \frac{\alpha l(e)}{\rho - 1}(r(\rho^k - 1) + k\rho^k - r(\rho^k - 1) + k(\rho^r - \rho^k)) \\ &= \frac{\alpha l(e)}{\rho - 1}(k\rho^r) \\ &= \frac{\rho^r}{\rho^r - 1} \cdot k \cdot l(e). \ \Box \end{aligned}$$

We now prove that GEOMETRIC's competitive ratio is optimal for all values of r.

Theorem 2 *Let A be a randomized on-line replication algorithm. Then A cannot be better than $(\frac{\rho^r}{\rho^r-1})$-competitive against any oblivious adversary, even on a graph consisting of two nodes.*

Proof: Let s and t be two nodes connected by an edge of length 1. We assume that initially, only node s has the page. We will construct a request sequence σ consisting of requests at node t such that the expected cost incurred by A is at least $\frac{\rho^r}{\rho^r-1}$ times the optimal off-line cost.

For $i = 1, 2, \ldots$, let q_i be the probability that A replicates the page from s to t after exactly i requests, given a request sequence that consists only of requests at node t. In the following we compare the algorithm A to the algorithm GEOMETRIC. Let $E[C_A(\sigma)]$ and $E[C_G(\sigma)]$ denote the expected cost incurred by A and GEOMETRIC on a request sequence σ. Furthermore, for $i = 1, 2, \ldots, r$, let $p_i = \alpha \cdot \rho^{i-1}$. We consider two cases.

Case 1: There exists an l, where $1 \le l \le r$, such that $\sum_{i=1}^{l} q_i \ge \sum_{i=1}^{l} p_i$.
Let k be the smallest number satisfying the above inequality, i.e., $\sum_{i=1}^{k} q_i \ge \sum_{i=1}^{k} p_i$ and $\sum_{i=1}^{j} q_i < \sum_{i=1}^{j} p_i$ for all j with $1 \le j < k$. Let σ be the request sequence that consists of k requests at node t. We show that the inequality $E[C_A(\sigma)] - E[C_G(\sigma)] \ge 0$ holds. This implies $E[C_A(\sigma)] \ge E[C_G(\sigma)] = \frac{\rho^r}{\rho^r-1} \cdot k$ and A cannot be better than $(\frac{\rho^r}{\rho^r-1})$-competitive because the optimal off-line cost on σ equals k. Since, with probability $1 - \sum_{i=1}^{k} q_i$, A has not replicated the page to t after the service of the request sequence σ, we have

$$E[C_A(\sigma)] = \sum_{i=1}^{k}(r + i)q_i + k(1 - \sum_{i=1}^{k} q_i).$$

Similarly, we also have

$$E[C_G(\sigma)] = \sum_{i=1}^{k}(r + i)p_i + k(1 - \sum_{i=1}^{k} p_i).$$

Hence $$E[C_A(\sigma)] - E[C_G(\sigma)] = \sum_{i=1}^{k} i(q_i - p_i) + (r - k)\sum_{i=1}^{k}(q_i - p_i).$$

Since $\sum_{i=1}^{k} q_i \ge \sum_{i=1}^{k} p_i$ and $r - k \ge 0$, we obtain

$$E[C_A(\sigma)] - E[C_G(\sigma)] \ge \sum_{i=1}^{k} i(q_i - p_i) = \sum_{i=1}^{k}(\sum_{j=i}^{k} q_j - \sum_{j=i}^{k} p_j).$$

For $i = 2, 3, \ldots, k$ we have $\sum_{j=1}^{i-1} q_j < \sum_{j=1}^{i-1} p_j$ and hence $\sum_{j=i}^{k} q_j - \sum_{j=i}^{k} p_j > \sum_{j=i}^{k} q_j - \sum_{j=i}^{k} p_j + \sum_{j=1}^{i-1} q_j - \sum_{j=1}^{i-1} p_j = \sum_{j=1}^{k} q_j - \sum_{j=1}^{k} p_j$. We conclude

$$E[C_A(\sigma)] - E[C_G(\sigma)] \geq \sum_{i=1}^{k}(\sum_{j=i}^{k} q_j - \sum_{j=i}^{k} p_j) \geq \sum_{i=1}^{k}(\sum_{j=1}^{k} q_j - \sum_{j=1}^{k} p_j) \geq 0.$$

Case 2: For all $k = 1, 2, \ldots, r$, the inequality $\sum_{i=1}^{k} q_i < \sum_{i=1}^{k} p_i$ is satisfied.
Let σ be the request sequence that consists of $2r$ requests at node t. Let A' be the on-line algorithm with $q_i' = q_i$, for $i = 1, 2, \ldots, r-1$, and $q_r' = 1 - \sum_{i=1}^{r-1} q_i$. Then

$$\begin{aligned} E[C_A(\sigma)] &= \sum_{i=1}^{2r}(r+i)q_i + 2r(1 - \sum_{i=1}^{2r} q_i) \;\geq\; \sum_{i=1}^{r-1}(r+i)q_i + 2r(1 - \sum_{i=1}^{r-1} q_i) \\ &= \sum_{i=1}^{r-1}(r+i)q_i' + 2rq_r' \;=\; E[C_{A'}(\sigma)]. \end{aligned}$$

Since $\sum_{i=1}^{r} q_i' = \sum_{i=1}^{r} p_i = 1$ and $\sum_{i=1}^{j} q_i' < \sum_{i=1}^{j} p_i$ for all j with $1 \leq j < r$, Case 1 immediately implies $E[C_A(\sigma)] \geq E[C_{A'}(\sigma)] \geq E[C_G(\sigma)] = \frac{\rho^r}{\rho^r - 1} \cdot r$, and A cannot be better than $(\frac{\rho^r}{\rho^r-1})$-competitive because the optimal off-line cost equals r. □

Next we present another on-line replication algorithm for trees. This algorithm has the advantage of being *memoryless*, i.e., it does not need any memory (for instance for counts) in order to determine when a replication should take place. Also, its competitive performance holds against adaptive on-line adversaries.

Algorithm COINFLIP (for trees): If there is a request at a node with the page, then the algorithm performs no action. If there is a request at a node v without the page, the algorithm serves the request by accessing the closest node u with the page. Then with probability $\frac{1}{r}$, the algorithm replicates the page from u to v.

Theorem 3 *The algorithm COINFLIP is 2-competitive against any adaptive on-line adversary.*

Proof: We use a potential function Φ to analyze COINFLIP. For any request sequence σ generated by an adaptive on-line adversary ADV, we compare simultaneous runs of COINFLIP and ADV on σ by merging the actions of both algorithms into a single sequence of events. This sequence contains two types of events: (Type I) ADV replicates the page. (Type II) A request is served by COINFLIP and ADV; this event may be accompanied by COINFLIP replicating the page to the requesting node.

For any event, let ΔC_{CF} and ΔC_{ADV} denote the costs incurred by COINFLIP and ADV during the event, and let $\Delta\Phi$ denote the change in potential. We will show that for any event,

$$E[\Delta C_{CF}] + E[\Delta\Phi] \leq 2\Delta C_{ADV}. \tag{3}$$

Summing up this inequality for all events, we obtain

$$E[C_{CF}(\sigma)] + E[\Phi_{end}] - E[\Phi_{start}] \leq 2C_{ADV}(\sigma), \tag{4}$$

where Φ_{start} and Φ_{end} denote the initial and final potential. Since we will choose the potential function such that Φ is always non-negative and such that the initial potential is 0, (4) implies that COINFLIP is 2-competitive.

We define the potential function. Let E be the set of edges e in the tree T such that ADV has replicated the page to the child node of e but COINFLIP has not replicated the page to the child node. Let

$$\Phi = 2r \sum_{e \in E} l(e).$$

In the following we prove (3) for all events. Let $ch(E)$ denote the set of the child nodes of all edges contained in E.

Type I: ADV replicates the page.
Suppose that the page is replicated from node u to node v (and to all nodes along the path from u to v). Then $\Delta C_{ADV} = rl(u,v)$ and $\Delta C_{CF} = 0$. Thus we must show $\Delta\Phi \leq 2rl(u,v)$. There are two cases to consider depending of whether $v \in ch(E)$ after the replication.
Case 1: $v \notin ch(E)$ after the replication.
Then $\Delta\Phi = 0$.
Case 2: $v \in ch(E)$ after the replication.
If u was in $ch(E)$ before the replication, then $\Delta\Phi = 2rl(u,v)$. Otherwise $\Delta\Phi \leq 2rl(u,v)$.

Type II: A request is served by COINFLIP and ADV.
Let v be the node at which the request occurs. We have to consider two cases.
Case 1: In the tree maintained by COINFLIP, node v already has the page.
Then $\Delta C_{CF} = 0$ and $\Delta C_{ADV} \geq 0$. Also $\Delta\Phi = 0$ because COINFLIP does not replicate the page. Inequality (3) is satisfied.

Case 2: In the tree maintained by COINFLIP, node v does not have the page.
Let u_{CF} be the node closest to v in the tree to which COINFLIP has replicated the page. Recall that u_{CF} lies on the path from v to the root. COINFLIP incurs a cost of $l(u_{CF}, v)$ in serving the request. Then with probability $\frac{1}{r}$, COINFLIP also replicates the page from u_{CF} to v. Therefore $E[\Delta C_{CF}] = l(u_{CF}, v) + \frac{1}{r} rl(u_{CF}, v) = 2l(u_{CF}, v)$. For the evaluation of ΔC_{ADV}, we have to consider three cases, depending on ADV's configuration of nodes with the page. First suppose that ADV has not replicated the page beyond u_{CF} when the request occurs. Then $\Delta C_{ADV} \geq l(u_{CF}, v) = \frac{1}{2}E[\Delta C_{CF}]$. Note that $E[\Delta\Phi] = 0$ because Φ does not change regardless of whether COINFLIP replicates the page or not. Inequality (3) holds. Next imagine that ADV has replicated the page beyond u_{CF} but not beyond v. Let u_{ADV} be the node closest to v to which ADV has replicated the page. Then $\Delta C_{ADV} = l(u_{ADV}, v)$. We have to show that the expected change in potential is $E[\Delta\Phi] = -2l(u_{CF}, u_{ADV})$. This implies $E[\Delta C_{CF}] + E[\Delta\Phi] = 2l(u_{CF}, v) - 2l(u_{CF}, u_{ADV}) = 2l(u_{ADV}, v) = 2\Delta C_{ADV}$. We have $\Delta\Phi = 0$ if COINFLIP does not replicate the page; otherwise $\Delta\Phi = -2rl(u_{CF}, u_{ADV})$. Hence $E[\Delta\Phi] = \frac{1}{r}(-2rl(u_{CF}, u_{ADV})) = -2l(u_{CF}, u_{ADV})$. Finally, suppose that ADV has replicated the page beyond v. In this case $\Delta C_{ADV} = 0$. If COINFLIP does not replicate the page, then $\Delta\Phi = 0$. Otherwise $\Delta\Phi = -2rl(u_{CF}, v)$. Therefore, $E[\Delta\Phi] = \frac{1}{r}(-2rl(u_{CF}, v)) = -2l(u_{CF}, v) = -E[\Delta C_{CF}]$. Inequality (3) holds. $\Box$

The COINFLIP algorithm achieves the best possible performance. No randomized on-line algorithm A can be better than 2-competitive against any adaptive on-line adversary, even on a graph consisting of two nodes. This can be seen as follows. Consider two nodes s and t connected by an edge of length 1 and assume that s has the page initially. An adaptive on-line adversary issues requests at t until A replicates the page to t. With probability $\frac{1}{2}$, the adversary initially replicates the page to t, and with probability $\frac{1}{2}$ it serves all the requests by accessing s. Suppose that A replicates the page after k requests. Then A's cost is $k + r$, whereas the expected cost of the adversary is $\frac{1}{2}(k + r)$. If A never replicates the page to t, then, by making the request sequence sufficiently long, we can achieve a lower bound of $2 - \epsilon$, for any $\epsilon > 0$.

4.2 Uniform networks

Any replication algorithm for trees can be easily applied to uniform networks. Consider an arbitrary uniform network and let s be the node that has the page initially. Since all edges in the graph have the same length, we may assume without loss of generality that a replication algorithm (on-line or off-line) serves requests and replicates the page only along edges $\{s, v\}$. Hence the network can be reduced to a tree by neglecting the edges $\{v, w\}$ with $v \neq s$, $w \neq s$. Run on this tree, any on-line algorithm for trees can maintain its competitive performance. The results given in Section 4.1 imply the following corollaries.

Corollary 1 *The algorithm GEOMETRIC for uniform networks is* $(\frac{\rho^r}{\rho^r - 1})$*-competitive against any oblivious adversary. This is the best competitive ratio that a randomized on-line replication algorithm can achieve against this type of adversary.*

Corollary 2 *The algorithm COINFLIP for uniform networks is 2-competitive against any adaptive on-line adversary. This is the best competitive ratio that a randomized on-line replication algorithm can achieve against this type of adversary.*

5 Algorithms for the ring

In this section we assume that the given net of processors forms a ring. First we will present techniques that transform c-competitive algorithms for trees into $2c$-competitive algorithms for rings. Using these techniques, we obtain a deterministic ring algorithm and randomized ring algorithms that are competitive against any oblivious adversary. Then we will develop a randomized replication algorithm for rings that is competitive against any adaptive on-line adversary.

We assume that initially, only one node of the ring, say s, has the page. Let n be the number of nodes in the ring and let $v_1, v_2, \ldots, v_n$ be the nodes if we scan the ring in clockwise direction starting from s, i.e., $v_1 = s$. For $i = 1, 2, \ldots, n$, let $e_i = \{v_i, v_{i+1}\}$ be the undirected edge from v_i to v_{i+1}. Naturally, v_{n+1} equals v_1. Again, for any edge e_i, $l(e_i)$ is the length of e_i. Let x and y be any two points on the ring; x and y need not necessarily be processor nodes. We denote by (x, y) the arc of the ring that is obtained if we start in x and go to y in clockwise direction. Let $l(x, y)$ be the length of the arc (x, y).

5.1 General techniques

First we present a deterministic strategy for constructing ring algorithms.

Algorithm RING: Let P, $P \neq s$, be the point on the ring satisfying $l(s,P) = l(P,s)$, i.e., P is the point "opposite" to s. The algorithm first cuts the ring at P. It regards the resulting structure as a tree T with root $s = v_1$. The arc (s,P) represents one branch of the tree and the arc (P,s) represents another branch of the tree (see Figure 1). We assume that the point P becomes part of the arc (s,P). This is significant if P coincides with one of the processor nodes v_i. The algorithm RING then uses an on-line replication algorithm A for trees in order to serve a request sequence σ. That is, RING assumes that σ is a request sequence for T and serves the request sequence using the tree algorithm A.

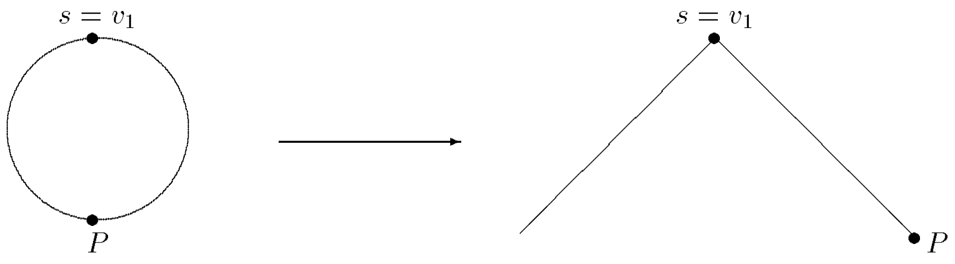

Fig. 1: A cut of the ring

Theorem 4 *Let A be an on-line replication algorithm that is c-competitive for an arbitrary tree. If the algorithm RING uses A as tree algorithm, then the resulting algorithm is $2c$-competitive. (If A is a randomized on-line algorithm, then the competitive ratio of $2c$ holds against any oblivious adversary.)*

Before we prove this theorem, we mention some important implications. Theorem 1 immediately implies the following result.

Corollary 3 *If RING uses the algorithm GEOMETRIC as the tree algorithm, then the resulting algorithm is c-competitive against any oblivious adversary, where $c = \frac{2\rho^r}{\rho^r - 1}$.*

We observe that c goes to $\frac{2e}{e-1} \approx 3.16$ as r tends to infinity. Also note that if RING uses the GEOMETRIC algorithm, then only one random number is used during an initialization phase.

Next we consider the deterministic replication algorithm for trees proposed by Black and Sleator [6]. The algorithm achieves an optimal competitive ratio of 2.

Algorithm DETERMINISTIC_COUNT: The algorithm works in the same way as the algorithm GEOMETRIC. However DETERMINISTIC_COUNT does not choose a random number in order to determine when a replication should occur. Rather it replicates the page to the child node of an edge when the corresponding count reaches r.

Corollary 4 *If the algorithm RING uses DETERMINISTIC_COUNT as the tree algorithm, then the resulting algorithm is 4-competitive.*

We remark that the combination of RING and DETERMINISTIC_COUNT is a complete deterministic on-line algorithm.

Theorem 4 and Theorem 3 imply the following result.

Corollary 5 *If RING uses the algorithm COINFLIP as tree algorithm, then the resulting algorithm is 4-competitive against any oblivious adversary.*

Note that the combination of RING and COINFLIP is memoryless.

Next we present a randomized variant of the algorithm RING and a statement analogous to Theorem 4.

Algorithm RING(RANDOM): The algorithm works in the same as the algorithm RING. However, instead of cutting the ring at the point opposite to s, the algorithm RING(RANDOM) chooses a point P uniformly at random on the ring and cuts the ring at that point P.

Theorem 5 *Let A be an on-line replication algorithm that is c-competitive for an arbitrary tree. If the algorithm RING(RANDOM) uses A as tree algorithm, then the resulting algorithm is $2c$-competitive against any oblivious adversary.*

Theorem 5 implies that if the cutting point P is chosen randomly, the same competitive performance is obtained as if the cutting point is chosen deterministically to be the point opposite to s. Therefore, statements analogous to Corollaries 3 - 5 hold. Note, however, that a combination of RING(RANDOM) and DETERMINISTIC_COUNT is not a purely deterministic algorithm.

It remains to prove the above theorems. In the following we present a detailed proof of Theorem 4. Since Theorem 5 is an interesting statement but does not yield stronger results than Theorem 4, we omit a proof of Theorem 5. The proof of Theorem 5 is similar to that of Theorem 4.

Proof of Theorem 4: Let σ be a request sequence for the ring. We start with some observations on how the optimal off-line algorithm OPT serves σ. Consider the state of the ring after OPT has served σ. Let u_a and u_b be the nodes farthest from s to which OPT has replicated the page in clockwise and counter-clockwise direction, respectively. Figure 2(a) illustrates this situation. We may assume without loss of generality that OPT replicates the page from s to u_a and from s to u_b at the beginning of the request sequence, before any requests are served. This does not incur a higher cost as if the replication is done while requests are processed.

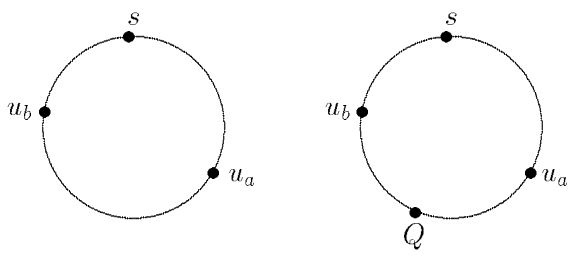

Fig. 2. (a) (b)

Any request at a node that belongs to (s, u_a) or (u_b, s) can then be served at zero cost. Let Q be the point on (u_a, u_b) which satisfies $l(u_a, Q) = l(Q, u_b)$, see Figure 2(b). Any request at a node v that belongs to (u_a, Q) is served by accessing u_a and the incurred cost equals $l(u_a, v)$. Any request at a node v that belongs to (Q, u_b) is served by accessing u_b, and the incurred cost equals $l(v, u_b)$.

Let $C_R(\sigma)$ be the cost incurred by RING in serving σ. Furthermore, let T be the tree that is obtained if the ring is cut at point P.

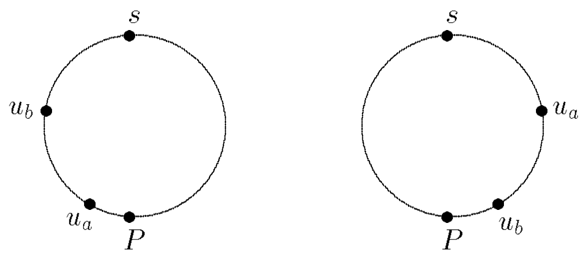

Fig. 3. (a) (b)

Case 1: Suppose that P belongs either to (s, u_a) or to (u_b, s), see Figure 3.
Let TOPT be the off-line algorithm that serves σ optimally, i.e. with minimal cost, on the tree T. By assumption, since the tree algorithm A is c-competitive, $C_R(\sigma) \le c \cdot C_{TOPT}(\sigma)$. Also, $C_{TOPT}(\sigma) \le 2r \cdot l(s, P)$. Since $C_{OPT}(\sigma) \ge r \cdot l(s, P)$, we obtain

$$C_R(\sigma) \le c \cdot C_{TOPT}(\sigma) \le 2cr \cdot l(s, P) \le 2c \cdot C_{OPT}(\sigma)$$

and the theorem is proved.

Case 2: Now suppose that P belongs neither to (s, u_a) nor to (u_b, s).
We only consider the case that $l(s, u_a) \ge l(u_b, s)$. The case $l(u_b, s) \ge l(s, u_a)$ is symmetric. Now, let TOPT be the algorithm that first replicates the page from s to u_a and from s to u_b in clockwise and counter-clockwise direction, respectively, and then serves the request sequence as follows. Any request at a node v that belongs to (u_a, P) is served by accessing u_a, and any request at a node v that belongs to (P, u_b), $v \ne P$, is served by accessing u_b. Since both RING and TOPT use T as underlying tree and since tree algorithm A is c-competitive, we have

$$C_R(\sigma) \le c \cdot C_{TOPT}(\sigma).$$

In the following we will show that

$$C_{TOPT}(\sigma) \le 2 \cdot C_{OPT}(\sigma). \tag{5}$$

This implies the theorem.

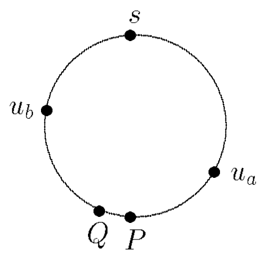

Fig. 4.

We compare the cost incurred by TOPT and OPT. Note that only requests at nodes on (P,Q) are served in different ways by TOPT and OPT. For each request on (P,Q), TOPT incurs a cost that is by at most $2l(P,Q)$ greater than the cost incurred by OPT. There occur at most r requests on (P,Q), since otherwise OPT would have replicated the page from u_a beyond P. Thus

$$C_{TOPT}(\sigma) \leq C_{OPT}(\sigma) + r \cdot 2l(P,Q).$$

We have $l(P,Q) = \frac{1}{2}(l(s,u_a) - l(u_b,s)) \leq \frac{1}{2}l(s,u_a)$. Thus,

$$C_{TOPT}(\sigma) \leq C_{OPT}(\sigma) + r \cdot l(s,u_a) \leq 2 \cdot C_{OPT}(\sigma)$$

and (5) is proved. □

5.2 A randomized algorithm against adaptive adversaries

We present an on-line replication algorithm that is 4-competitive against any adaptive on-line adversary. This algorithm has the additional advantage of being memoryless. The replication strategy used by the algorithm is motivated by the HARMONIC k-server algorithm [10].

Algorithm HARM-RING: If there is a request at a node with the page, then the algorithm performs no action. If there is a request at a node v without the page, then let w_a and w_b be the nodes farthest from s to which HARM-RING has replicated the page in clockwise and counter-clockwise direction, respectively. With probability $\frac{l(w_a,v)}{l(w_a,w_b)}$ the algorithm serves the request by accessing w_b, and with probability $\frac{l(v,w_b)}{l(w_a,w_b)}$ the algorithm serves the request by accessing w_a. Then with probability $\frac{1}{r}$, HARM-RING replicates the page to v from the node that was actually accessed during the service of the request.

Theorem 6 *The algorithm HARM-RING is 4-competitive against any adaptive on-line adversary.*

Proof: For any request sequence σ generated by an adaptive on-line adversary, we compare simultaneous runs of HARM-RING and ADV on σ. As in the proof of Theorem 3, the actions of HARM-RING and ADV can be classified into two types of events. (Type I) ADV replicates the page. (Type II) A request is served by HARM-RING and ADV; this event may be accompanied by HARM-RING replicating the page to the requesting node. As before, we will give a non-negative potential function Φ, that is initially 0, so that

$$E[\Delta C_{HR}] + E[\Delta\Phi] \leq 4 \cdot \Delta C_{ADV} \qquad (6)$$

for all events. This implies the theorem. Here ΔC_{HR} and ΔC_{ADV} denote the cost incurred by HARM-RING and ADV during the event; $\Delta\Phi$ is the change in potential.

We define the potential function. At any given time, let u_a and u_b be the nodes farthest from s to which ADV has replicated the page in clockwise and counter-clockwise direction, respectively. Recall that w_a and w_b are the nodes farthest from s to which HARM-RING has replicated the page. Let

$$\Phi = 4r\max\{0, l(s,u_a) - l(s,w_a)\} + 4r\max\{0, l(u_b,s) - l(w_b,s)\}.$$

Intuitively, Φ is the length of the range of the ring at which ADV has the page but HARM-RING has not. We will show (6) for all events.

(Type I) ADV replicates the page.
Suppose that the page is replicated from node u to node v. Assume without loss of generality that the page is replicated in clockwise direction, i.e., the arc (s, u_a) is extended. The case that the arc (u_b, s) is extended is analogous. We have $\Delta C_{ADV} = rl(u, v)$ and $\Delta C_{HR} = 0$. We must show $E[\Delta\Phi] \leq 4rl(u, v)$. If neither node u nor v has the page in the ring maintained by HARM-RING, then $\Delta\Phi = 4rl(u, v)$. Otherwise $\Delta\Phi \leq 4rl(u, v)$. Inequality (6) holds.

(Type II) A request is served by HARM-RING and ADV.
Let v be the node requesting the page. If v has the page in the ring maintained by HARM-RING, then $\Delta C_{HR} = 0$, $\Delta C_{ADV} \geq 0$ and $\Delta\Phi = 0$ because HARM-RING does not replicate the page. Inequality (6) is satisfied. In the remainder we assume that v does not have the page in the ring maintained by HARM-RING, i.e., v lies on the arc (w_a, w_b) and $w_a \neq v \neq w_b$. Then

$$\begin{aligned} E[\Delta C_{HR}] &= \frac{l(v, w_b)}{l(w_a, w_b)}\left(l(w_a, v) + \frac{1}{r} rl(w_a, v)\right) + \frac{l(w_a, v)}{l(w_a, w_b)}\left(l(v, w_b) + \frac{1}{r} rl(v, w_b)\right) \\ &= 4\frac{l(w_a, v) \cdot l(v, w_b)}{l(w_a, w_b)}. \end{aligned}$$

We may assume without loss of generality that in the ring maintained by ADV, v lies on the arc (s, u_a) or ADV serves the request by accessing u_a. The case that v lies on the arc (u_b, s) or that ADV serves the request by accessing u_b is symmetric. For the evaluation of $E[\Delta\Phi]$ and ΔC_{ADV} we investigate three cases regarding the relative positions of u_a, w_a and v, as illustrated in Figure 5.

We introduce an ordering on the nodes of the ring such that for two nodes x, y:

$$x < y \quad \text{if} \quad l(s, x) < l(s, y).$$

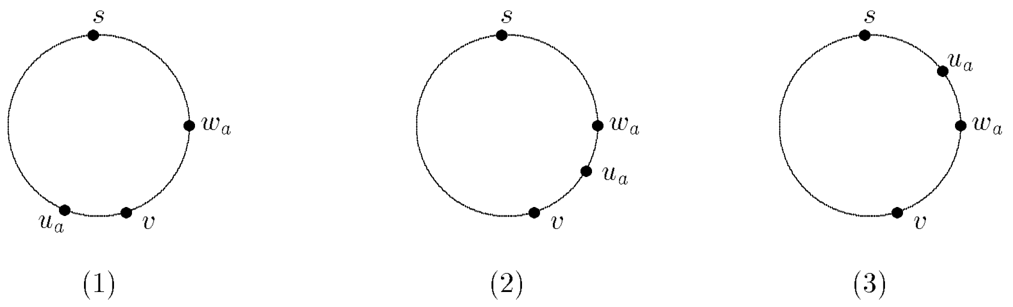

Fig. 5: Three case regarding the locations of u_a, w_a, and v

Case 1: Suppose that $w_a < v \leq u_a$.
We have $\Delta C_{ADV} = 0$. The change in potential is $-4rl(w_a, v)$ if HARM-RING replicates the page from w_a to v. If HARM-RING replicates the page from w_b to v, then the change in potential is non-positive. Thus

$$E[\Delta\Phi] \leq \frac{1}{r}\frac{l(v, w_b)}{l(w_a, w_b)}(-4rl(w_a, v)) = -4\frac{l(w_a, v) \cdot l(v, w_b)}{l(w_a, w_b)} = -E[\Delta C_{HR}]$$

and (6) holds.

Case 2: Suppose that $w_a \leq u_a < v$.
In this case $\Delta C_{ADV} = l(u_a, v)$. If HARM-RING replicates the page from w_a to v, then the potential change is $-4rl(w_a, u_a)$. Again, if HARM-RING replicated the page from w_b to v, then the change in potential is non-positive. Therefore

$$E[\Delta\Phi] \leq \frac{1}{r}\frac{l(v, w_b)}{l(w_a, w_b)}(-4rl(w_a, u_a)) = -4\frac{l(w_a, u_a) \cdot l(v, w_b)}{l(w_a, w_b)}.$$

We obtain

$$E[\Delta C_{HR}] + E[\Delta\Phi] \leq 4\frac{l(u_a, v) \cdot l(v, w_b)}{l(w_a, w_b)} \leq 4l(u_a, v) = 4\Delta C_{ADV}.$$

The second inequality holds because $\frac{l(v,w_b)}{l(w_a,w_b)} \leq 1$. Inequality (6) is satisfied.

Case 3: Suppose that $u_a < w_a < v$. Here $\Delta C_{ADV} = l(u_a, v)$ and $\Delta\Phi \leq 0$. We have

$$E[\Delta C_{HR}] = 4\frac{l(w_a, v) \cdot l(v, w_b)}{l(w_a, w_b)} \leq 4l(w_a, v) \leq 4l(u_a, v) = 4\Delta C_{ADV}.$$

As before, the first inequality follows from the fact that $\frac{l(v,w_b)}{l(w_a,w_b)} \leq 1$. Again, Eq. (6) holds. □

6 Conclusion and open problems

We have investigated the page replication problem for important network topologies such as trees, uniform networks and rings. For these topologies we have developed deterministic and randomized on-line algorithms that achieve a constant competitive ratio. Our randomized algorithms for trees and uniform networks achieve the best possible competitive ratios. While the competitiveness achieved by deterministic and randomized algorithms is settled for trees and uniform networks, a number of open problems remain with respect to the ring topology. One interesting problem is to tighten the gap for deterministic algorithms. We have presented a 4-competitive deterministic replication algorithm. Black and Sleator [6] mention (without proof) that no deterministic on-line algorithm for rings can be better than $\frac{5}{2}$-competitive. Moreover, no lower bounds are known on the competitiveness achieved by randomized on-line algorithms on rings. An interesting problem is to develop lower or improved upper bounds for rings.

This paper (and almost all other related previous work) studies the page replication under the assumption that the local memories of the processors have infinite memory capacity. That is, whenever an algorithm wants to replicate a given page into the local memory of a processor, there is room for it; no other page needs to be dropped. An important problem is to study the page replication under the assumption that the local memories have bounded capacity. Bartal *et al.* [4] showed that in this model, no on-line replication algorithm in any topology can be better than $\Omega(m)$-competitive, where m is the total number of pages that can be stored in the network. They also gave an $O(m)$-competitive algorithm for uniform networks. One approach to overcome the $\Theta(m)$ bound might be to consider special memory types. Albers and Koga [1] studied the page migration problem for *direct-mapped* memories, i.e., the processors use a hash function in order to locate pages in their local memories. Awerbuch *et al.* [3] investigated distributed paging problems when an on-line algorithm has slightly more memory capacity than the off-line algorithm.

Topology	Problem
Ring	Determine the competitive ratios c achieved by deterministic and randomized on-line algorithms. Best bounds known for – det. alg.: $2.5 \leq c \leq 4$ – rand. alg. against oblivious adv.: $c \leq 3.16$ – rand. alg. against adaptive on-line adv.: $c \leq 4$
Arbitrary	Find memory models for which on-line algorithms with a competitive ratio of $o(m)$ can be developed.

Table 1: Summary of open problems

References

[1] S. Albers and H. Koga. Page migration with limited local memory capacity. In Proc. 4th International Workshop on Algorithms and Data Structures (WADS95), Springer LNCS, Vol. 955, pages 147–158, 1995.

[2] B. Awerbuch, Y. Bartal and A. Fiat. Competitive distributed file allocation. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, pages 164–173, 1993.

[3] B. Awerbuch, Y. Bartal and A. Fiat. Distributed paging for general networks. In *Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 574–538, 1996.

[4] Y. Bartal, A. Fiat and Y. Rabani. Competitive algorithms for distributed data management. In *Proc. 24th Annual ACM Symposium on Theory of Computing*, pages 39–50, 1992.

[5] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, **11**(1):2–14, 1994.

[6] D.L. Black and D.D. Sleator. Competitive algorithms for replication and migration problems. Technical Report Carnegie Mellon University, CMU-CS-89-201, 1989.

[7] M. Chrobak, L.L. Larmore, N. Reingold and J. Westbrook. Page migration algorithms using work functions. In *Proc. 4th International Annual Symposium on Algorithms and Computation*, Springer LNCS Vol. 762, pages 406–415, 1993.

[8] D. Downey and D. Foster. Comparative models of the file assignment problem. *Computing Surveys*, **14**(2):287–313, 1982.

[9] C. Lund, N. Reingold, J. Westbrook and D. Yan. On-line distributed data management. In *Proc. of the 2nd Annual European Symposium on Algorithms*, Springer LNCS Vol. 855, pages 202–214, 1994.

[10] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. In *Proc. 16th International Colloquium on Automata, Languages and Progamming*, Springer LNCS Vol. 372, pages 687–703, 1989.

[11] C. Scheurich and M. Dubois. Dynamic page migration in multiprocessors with distributed global memory. *IEEE Transactions on Computers*, **38**(8):1154–1163, 1989.

[12] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, **28**:202–208, 1985.

[13] J. Westbrook. Randomized algorithms for the multiprocessor page migration. *SIAM Journal on Computing*, **23**:951–965, 1994.