A DISTRIBUTED ALGORITHM FOR ADAPTIVE REPLICATION OF DATA

Ouri Wolfson

Columbia University
Dept. of Computer Science
Technical Report CUCS-057-90 (Revised version)

ABSTRACT

We present a distributed algorithm for replication of a data-item in a set of processors interconnected by a tree network. The algorithm is adaptive in the sense that the replication scheme of the item (i.e. the set of processors, each of which stores a replica of the data-item), changes as the read-write pattern of the processors in the network changes. The algorithm is optimal in the sense that when the replication scheme stabilizes, the total number of messages required for the reads and writes is minimal.

A DISTRIBUTED ALGORITHM FOR ADAPTIVE REPLICATION OF DATA

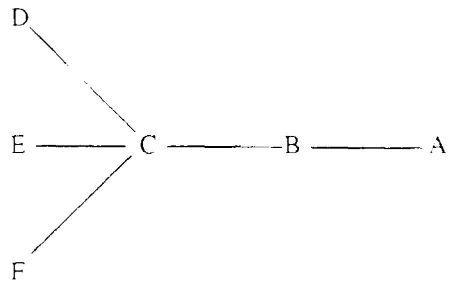
Preliminary Version

Ouri Wolfson[1]
Distributed Computing and Communication Lab.
450 Computer Science Bldg.
Columbia Univ.
New York, NY 10027
wolfson@cs.columbia.edu

1. Introduction

We present a distributed algorithm for replication of a data-item in a set of processors interconnected by a tree network. The algorithm is adaptive in the sense that the replication scheme of the item (i.e. the set of processors, each of which stores a replica of the data-item), changes as the read-write pattern of the processors in the network changes. The algorithm is optimal in the sense that when the replication scheme stabilizes, the total number of messages required for the reads and writes is minimal.

The algorithm, called ADAPTIVE-REPLICATION, works in the read-one-write-all context. Specifically, a read of the data-item is performed from the closest replica in the network. A write is to all the replicas, and it is propagated along the edges of the smallest network-subtree that contains the writer and the processors of the replication scheme. For example, consider the communication network T of the figure below, and suppose that the replication scheme consists of processors C and D.



When processor A writes the item, A sends the item to B, then B sends it to C, and then C in turn sends it to

[1] This research was supported in part by by the Center for Advanced Technology at Columbia University under contracts NYSSTF-CAT(89)-5 and CU01207901, and by NSF grant IRI-90-03341.

D: overall the number of inter-processor messages required for this write is three.

Why would we be interested in an *adaptive* replication scheme? To answer this question, first let us point out that for a given read-write pattern, i.e. number of reads and writes for each processor, the replication scheme determines the necessary number of messages. Consider again the network T, and suppose that the read-write pattern for the data-item is the following: each processor of the network, except A, reads the item once and writes it once during a time-period; processor A reads the item four times and writes it once during the time-period. Then, if the replication scheme consists of a single processor, C, the number of messages required by this read-write pattern is 18. On the other hand, if the replication scheme consists of processors B and C, then the number of messages required by the same read-write pattern is 17. In fact, the latter replication scheme is optimal in the sense that any other scheme requires at least 17 messages for the read-write pattern. However, for the read-write pattern in which each processor of the network reads the item once and writes it once during a time-period, the former replication scheme is optimal. Therefore, the optimal replication scheme depends on the read-write pattern. We would like the replication scheme to adapt dynamically, as the read-write pattern changes. The algorithm ADAPTIVE-REPLICATION performs exactly this function, although the read-write pattern is not known a priori.

The decisions to change in the replication scheme are local at each processor, and come as a result of its execution of reads and writes. In this sense the ADAPTIVE-REPLICATION algorithm is distributed. However, the overall effect of the algorithm is the following. Metaphorically, the replication scheme of the ADAPTIVE-REPLICATION algorithm forms a variable-size amoeba that stays connected at all times, and constantly moves towards the "center of read-write activity". The replication scheme expands as the read activity increases, and it contracts as the write activity increases. When at each "border" processor (i.e. processor of the circumference of the amoeba) the number of reads equals the number of writes, the replication scheme remains fixed. Then this scheme is optimal for the read-write pattern in the network.

The rest of the paper is organized as follows. In section 2 we present, demonstrate and discuss the ADAPTIVE-REPLICATION algorithm. In section 3 we show the optimality of ADAPTIVE-REPLICATION. In section 4 we compare this paper to other work on replicated data, and we discuss future work.

2. The ADAPTIVE-REPLICATION Algorithm

Our proposed algorithm, ADAPTIVE-REPLICATION, works as follows. The initial replication scheme consists of the whole set of processors. At any time, the processors of the replication scheme, R, induce a connected subgraph of the network. For example, ADAPTIVE-REPLICATION will never replicate the item at processors A and C of the network T. Each processor i that is a $\bar{R}$-*neighbor*, i.e. i belongs to the replication scheme but it has a neighbor that does not belong to R, performs the following test for each read request from a processor j that is not in R.

(Expansion–Test) If there are two read requests made by j[2], between which i receives no write request made by i or by a neighbor different than j, then i tells j to join R.

Practically, j joins R simply by saving a copy of the item sent by i as a result of the second consecutive read request. Except for i and j, no other processor is informed of the expansion.

Each processor j that is an R-*fringe* node, i.e. a leaf of the subgraph of the network induced by R, performs the following test for each write request from the single neighbor i that is in R.

(Contraction–Test) If there are two write requests made by i, between which j receives no read request made by j or by a neighbor of j, then j exits R, i.e. ceases to keep a copy.

j exits R by telling i not to send any more write requests to j; any further read requests arriving at j are passed along to i. Except for i and j, no other processor is informed of the contraction.

Finally, the following test is executed by a processor that constitutes the whole replication scheme, R, of the algorithm. Namely, a processor i that is in R but none of its neighbors is in R, performs the following test for each read or write operation from some neighbor n.

(Switch–Test) If i executes two operations o_1 and o_2, requested by n, such that o_1 or o_2 (or both) is a write and between them i executed no operation requested by i or by a neighbor other than n, then the single copy of the item is transferred from i to n (i.e. simultaneously, i exits R and n enters it).

[2] j makes read and write requests, each of which originates either in j or in some other node, k, such that the shortest path from k to R goes through j. For example, consider the network T. If $R = \{C\}$, then B makes read-write requests that originate in B or A.

To summarize the algorithm, each R-neighbor performs the Expansion-Test on a read request; additionally, an R-fringe node that has some neighbor that is also in R performs the Contraction-Test on a write request, and if it does not have a neighbor in R then it performs the Switch-Test on a read or a write request. Note that a node may be both an R-fringe node and a $\bar{R}$-neighbor.

Let us demonstrate the algorithm using the following example.

Example 1: Consider the network T. The initial replication scheme, R, consists of the whole set of processors in the network. Suppose that the first request is a write that originates at D. After it is received by all the processors, the second request is initiated. The second request is a write that originates at E. After the execution of the second write by all the processors, the R-fringe nodes F and A exit the replication scheme, R. For example, A exits since the first two requests were presented to A by B. Then the third request, a write that originates at B, is initiated (after its execution R = {B, C}; E and D exit since each has executed two write requests from C, without an intervening read). The fourth request is a read that originates at A (after its execution R remains {B, C}). The fifth request is a write that originates at A (after its execution R = {B}). The sixth request is a read that originates at A (R = {A, B}). The seventh request is a read that originates at D (R = {A, B}). The eighth request is a read that originates at F (R = {A, B, C}). []

Note that ADAPTIVE-REPLICATION requires that each node of R knows whether it is a $\bar{R}$-neighbor, or an R-fringe node, or a unique node of R. Knowing this requires only that the node knows the identity of its neighbors, and "remembers" for each neighbor whether or not it is in the replication scheme. A processor that does not belong to the replication scheme does not participate in the algorithm; nor does an internal node of the replication scheme, i.e. a node that is not an R-fringe, and that does not have a neighbor outside R.

What does a node need to know in order to execute reads and writes? A node j of the current replication scheme, R, satisfies a read request locally, and transmits each write request to the neighbors that are in the replication scheme (each of which in turn, propagates the write to its neighbors that are in R, except j). Therefore, j has to know the identity of its neighbors, and to remember for each neighbor whether or not it is in the replication scheme (same as the information needed to execute the algorithm ADAPTIVE-

REPLICATION). Interestingly, a node that does not belong to the replication scheme does not have to "search" for the replication scheme in order to execute reads and writes. A node j that is not in the replication scheme R, must remember the node i to which j sent the last announcement that j exits R. i indicates the "direction" of R. Each read or write of j must be sent to i, which in turn, if is not in R any longer, routes the request in the "direction" of R. Therefore, for executing the algorithm ADAPTIVE-REPLICATION and for executing reads and writes, knowledge of the whole network topology is not necessary; nor is knowledge of the whole replication scheme necessary.

Practically, each node, j, has a directory-record for each data-item. The record indicates whether or not j is in the replication scheme, R; if it is, the record also indicates which of j's neighbors are in R, and if it is not, it indicates the direction of R. Before accessing a data-item, j accesses its directory record. This access is part of the transaction that accesses the data-item, and consequently a concurrency control mechanism that ensures serializability of transactions in a static replication environment will also do so in this dynamic environment.

The next comment concerns the exit of a node j from the replication scheme, as a result of the Contraction-Test. j does not leave R unconditionally, simply by announcing j's exit to its neighbor i in R. The reason is that i and j may be the only nodes of the current replication scheme, and j and i may both announce their exit to each other, leaving an empty replication scheme. Therefore, if the Contraction-Test succeeds then j requests permission from i to exit R, but j keeps a copy of the data-item until it receives the next message from i. If the next message from i is i's request to leave R then only one (say the one with the smallest processor-identification-number) leaves R.

Note also that although we specified that ADAPTIVE-REPLICATION starts with a replication scheme that contains all the processors in the network, any replication scheme that induces a connected subgraph will suffice. The only difference is that at the outset, a processor i that does not belong to the replication scheme, R, must know R's direction; i.e. i must know the neighbor to which read and write requests should be sent.

Finally let us mention that, in contrast to [AE1], our model ignores the cost of storing a replica. Therefore, in practice, our algorithm is optimal only in an environment in which storage costs are negligible.

3. Analysis of ADAPTIVE-REPLICATION

In this section we formally define the model and analyze the algorithm. A *network* is an undirected tree, $T=(V,E)$. V represents a set of processors, and an edge in the network between two processors represents a bidirectional communication link between them. The *replication scheme* is a nonempty subset of V. For a given network and replication scheme, the *read cost* of a processor i, denoted r_i, is the length (in edges) of the shortest path in the network between i and a processor of the replication scheme. It represents the number of messages required for the data-item transfer. Obviously, if i is in the replication scheme, then the read cost is zero. Let R be a replication scheme, and assume that processor $i \in V$ writes the data-item. The *write cost* for i, denoted w_i, is the number of edges in the smallest subtree of T that contains $\{i \cup R\}$.

A *schedule* is a sequence $o_1^{j_1}, o_2^{j_2}, \ldots, o_n^{j_n}$. Each o_i is a read or write operation, and each j_i is a processor of V, at which the operation o_i originated. Intuitively each operation represents the initiation of the request, as well as its execution. In other words, we assume in particular that each write request is executed by all the processors of the replication scheme before the next request is issued. Given a schedule, $o_1^{j_1}, o_2^{j_2}, \ldots, o_n^{j_n}$, a *subschedule*, S, is a subsequence $o_i^{j_i}, o_{i+1}^{j_{i+1}}, \ldots, o_{i+k}^{j_{i+k}}$. Suppose that in S each processor, i, of the network performs $\#R_i$ reads, and $\#W_i$ writes. The set of pairs $A = \{ (\#R_i, \#W_i) \mid i$ is a processor in the network, and $\#R_i$ and $\#W_i$ are nonnegative integers$\}$, is called the *read-write-pattern* of S. Given a replication scheme, R, and a read-write pattern, A, the *replication scheme cost* for A, denoted $cost(R,A)$, is defined as $\sum_{i \in V} \#W_i \cdot w_i + \sum_{i \in V} \#R_i \cdot r_i$. Intuitively, $cost(R,A)$ represents the total number of messages sent during the subschedule, assuming that R is the replication scheme. A message is the transmission of the data-item over one communication link (edge). A replication scheme is *optimal* for a read-write pattern, A, if it has the minimum (among all replication schemes) cost for A.

Next we define a dynamic replication algorithm. A *configured-schedule*, $o_1^{j_1}(R_1), o_2^{j_2}(R_2), \ldots, o_n^{j_n}(R_n)$ is a schedule in which each operation is mapped to a replication scheme. Intuitively, it is the replication scheme that exists when the operation is initiated. We assume that any recomputation of the replication scheme is executed before the next operation is initiated. For the above configured schedule, the replication scheme R_i is *associated* with the operation $o_i^{j_i}$. A *dynamic replication algorithm* is a function that maps each schedule to a configured schedule.

Let S' be a schedule and let DRA be a dynamic replication algorithm. Suppose that S' is mapped by DRA to the configured schedule S''. Let S be a subschedule of S''. Suppose that the operations of S and the operation immediately succeeding the last one in S, are all associated with the same replication scheme R. Intuitively this means that S starts and ends with the same replication scheme. Then we say that DRA is *stable* on S, with *stability scheme* R.

Now consider ADAPTIVE-REPLICATION. It moves towards the center of read-write activity in the following sense. Suppose that at some point in time, t, all the processors become quiescent (i.e. stop initiating operations), except for one, i. Furthermore, suppose that i issues only reads. Then it is clear intuitively, and it can be proven formally, that ADAPTIVE-REPLICATION will stabilize, and the stability scheme will include i (i.e. it will be optimal for any subschedule consisting only of reads from i). Specifically, if at time t processor i is in the replication scheme, then as long as it issues read requests and all the other processors are quiescent, the replication scheme will not change. If processor i is not in the replication scheme, R, then R will expand towards i until it reaches i, and from then on ADAPTIVE-REPLICATION will become stable. Each expansion step will take two reads. Therefore, convergence to the optimal replication scheme will occur after a number of reads that is bounded by twice the number of processors in the network.

Now suppose that from point in time t on, processor i issues only writes. Then ADAPTIVE-REPLICATION will stabilize, and the stability scheme will be optimal for any subschedule consisting only of writes from i (i.e. the stability scheme will be a singleton set, consisting of the processor i). Specifically, if at time t processor i is in the replication scheme, then as long as it issues write requests and all the other processors are quiescent, the replication scheme will contract until it consists of the single processor, i. If processor i is not in the replication scheme, R, then denote by j the processor of R that is closest to i. R will contract until it consists of the singleton set $\{j\}$, and then it will switch until it consists of the singleton set $\{i\}$. In both cases, convergence to the optimal replication scheme will occur after a number of writes that is bounded by twice the number of processors in the network.

Therefore, for subschedules consisting of a single operation, the DRA ADAPTIVE-REPLICATION stabilizes, with a stability scheme that is optimal for the operation. However, ADAPTIVE-REPLICATION will stabilize not only when all processors except one become quiescent, but whenever the three tests performed by the algorithm will fail. This occurs when the reads and writes executed by each R-fringe and each $\bar{R}$-neighbor interleave perfectly during some subschedule S. The rest of this section is dedicated to the proof that in this case as well, the stability scheme is (almost) optimal for the read-write pattern in S.

The proof consists of several lemmas, leading to lemma 4; it provides four properties that characterize an optimal replication scheme. Then, theorem 1 states that when ADAPTIVE-REPLICATION stabilizes, the stability scheme possesses these four properties. We will start with some definitions and notations. Intuitively, for a network and a read-write pattern, the median is a node for which the sum of weighted-distances to the other nodes is minimum. Formally, in a tree, let l_{vu} denote the length (in edges) of the simple path between v and u. A *median* of the network is a node, m, for which $\sum_i (\#R_i + \#W_i) \cdot l_{mi}$ is minimum.

Denote the processors of a subtree, t, of the network by $V(t)$. Let R be a subset of nodes that induces a connected subgraph of the network. Denote by i a processor that is either an R-fringe, or, is not in R, but is a neighbor of some processor in R. Consider the removal of the edge between i and its neighbor in R, say j. It disconnects the network into two subtrees: a subtree that contains i, that we denote T_{i-R}, and a subtree that contains R, that we denote T_{R-i} (see Fig. 1). The subtrees may also be denoted T_{i-j} and T_{j-i}, respectively.

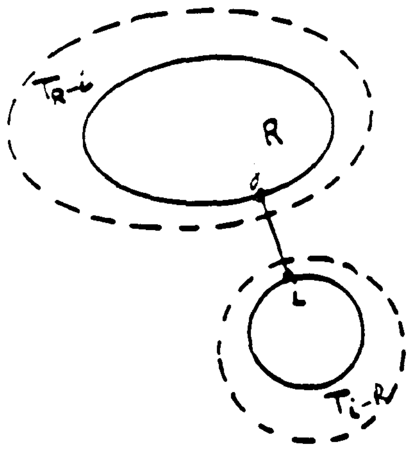


Figure 1: illustration of T_{i-R} and T_{R-i}.

Lemma 1: Let A be a read-write pattern, and let s and t be two neighbors in the tree. Then $\sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{su} = \sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{tu} + \sum_{u\in V(T_{t-s})} (\#R_u+\#W_u) - \sum_{u\in V(T_{s-t})} (\#R_u+\#W_u)$.

Proof: Observe that by considering separately T_{s-t} and T_{t-s}, and by substituting l_{tu} for l_{su} we obtain: $\sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{su} = \sum_{u\in V(T_{t-s})} (\#R_u+\#W_u)\cdot(l_{tu}+1) + \sum_{u\in V(T_{s-t})} (\#R_u+\#W_u)\cdot(l_{tu}-1)$. By manipulation of the right hand side of the equation, the lemma follows. []

Lemma 2: Let A be a read-write pattern. A node m is a median if and only if for each neighbor w of m: $\sum_{u\in V(T_{m-w})} (\#R_u+\#W_u) \geq \sum_{u\in V(T_{w-m})} (\#R_u+\#W_u)$.

Proof: (==>) Suppose that m is a median, but for one of its neighbors, w, $\sum_{u\in V(T_{m-w})} (\#R_u+\#W_u) < \sum_{u\in V(T_{w-m})} (\#R_u+\#W_u)$. Then, by lemma 1, $\sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{wu} < \sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{mu}$, contradiction to m being a median.

(<==) Suppose that $d\neq m$ is a median. Consider the path $d=x(1), x(2), \ldots, x(n-1), x(n)=m$. Since for each neighbor w of m: $\sum_{u\in V(T_{m-w})} (\#R_u+\#W_u) \geq \sum_{u\in V(T_{w-m})} (\#R_u+\#W_u)$, in particular:

(1) $\sum_{u\in V(T_{m-x(n-1)})} (\#R_u+\#W_u) \geq \sum_{u\in V(T_{x(n-1)-m})} (\#R_u+\#W_u)$.

Furthermore, remember that each node performs a nonnegative number of reads and writes. Thus, intuitively, when removing the edge between $x(n-1)$ and $x(n-2)$, rather than the edge between $x(n)$ and $x(n-1)$ as in inequality (1), the operations of $x(n-1)$ move from the right hand side of the inequality to its left hand side. Therefore, $\sum_{u\in V(T_{x(n-1)-x(n-2)})} (\#R_u+\#W_u) \geq \sum_{u\in V(T_{x(n-2)-x(n-1)})} (\#R_u+\#W_u)$. Moreover, by the same argument, for each $n\geq i\geq 2$ $\sum_{u\in V(T_{x(i)-x(i-1)})} (\#R_u+\#W_u) \geq \sum_{u\in V(T_{x(i-1)-x(i)})} (\#R_u+\#W_u)$. In particular, $\sum_{u\in V(T_{x(2)-x(1)})} (\#R_u+\#W_u) \geq \sum_{u\in V(T_{x(1)-x(2)})} (\#R_u+\#W_u)$. This, combined with lemma 1, gives: $\sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{x(2)u} \leq \sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{du}$. By an easy induction on i is can be shown that: $\sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{x(i)u} \leq \sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{x(i-1)u}$, for each $n\geq i\geq 2$. Therefore, $\sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{mu} \leq \sum_{u\in V(T)} (\#R_u+\#W_u)\cdot l_{du}$. Since d is a median, then

$\sum_{u \in V(T)} (\#R_u + \#W_u) \cdot l_{mu} = \sum_{u \in V(T)} (\#R_u + \#W_u) \cdot l_{iu}$, and consequently m is also a median. []

Lemma 3: Let A be a read-write pattern in which each node performs at least one operation (read or write). Then there are either one or two medians. Furthermore, if there are two medians then they are neighbors in the network.

Proof: Suppose by contradiction that the lemma does not hold. Then there must be two medians, d and m, that are not neighbors. Consider the path $d = x(1), x(2), \ldots, x(n-1), x(n) = m$, for $n \geq 3$. By lemma 2, for each neighbor w of m: $\sum_{u \in V(T_{m-w})} (\#R_u + \#W_u) \geq \sum_{u \in V(T_{w-m})} (\#R_u + \#W_u)$. In particular: $\sum_{u \in V(T_{m-x(n-1)})} (\#R_u + \#W_u) \geq \sum_{u \in V(T_{x(n-1)-m})} (\#R_u + \#W_u)$. Furthermore, since each node performs at least one operation, for each $n-1 \geq i \geq 2$ $\sum_{u \in V(T_{x(i)-x(i-1)})} (\#R_u + \#W_u) > \sum_{u \in V(T_{x(i-1)-x(i)})} (\#R_u + \#W_u)$. In particular, $\sum_{u \in V(T_{x(2)-x(1)})} (\#R_u + \#W_u) > \sum_{u \in V(T_{x(1)-x(2)})} (\#R_u + \#W_u)$. By lemma 2, this contradicts the fact that d is a median. []

Lemma 4: Let A be a read-write pattern in which each processor performs at least one operation (read or write). A replication scheme, R, that satisfies the following four conditions is optimal for A: 1. R is a (connected) subtree, and 2. each node i that is a neighbor of R satisfies: $\sum_{j \in V(T_{i-R})} \#R_j \leq \sum_{j \in V(T_{R-i})} \#W_j$, and 3. Suppose that R contains more than one node and let i be an R-fringe node. Then, $\sum_{j \in V(T_{i-R})} \#R_j \geq \sum_{j \in V(T_{R-i})} \#W_j$. 4. If R is a singleton set, then it consists of a median of the network.

Proof: In [WM] we have shown that the following algorithm computes an optimal replication scheme, RS[3].

[3] Actually, in [WM] we have proven that RS is an optimal replication scheme, provided that A is a read-write pattern in which each processor performs at least one read and at least one write. However, the same proof works for a read-write pattern in which each processor performs at least one operation.

```
TREE-RS [ T(V,E), A]: /* Algorithm for finding the optimal residence set, given a tree T
                         and a read-write pattern A in which each processor performs
                         at least one read and at least one write */
1. init: color all the processors of V red; initialize RS to a median, m.
2.     while there exists a processor in RS with at least one red neighbor, i, do:
3.         if $\sum_{j \in V(T_{i-RS})} \#R_j \geq \sum_{j \in V(T_{RS-i})} \#W_j$ then add $i$ to RS; else color $i$ blue.
4.     end while.
5. output: RS.
```

The main idea of the proof is to show that if a replication scheme, R, satisfies the four conditions of the lemma, then its cost is equal to the cost of the replication scheme RS (produced by the algorithm *TREE–RS*). The proof proceeds in three stages. In stage I we will show that R contains a median. In stage II we will show that if R and RS are disjoint, then each one of the sets is a singleton, implying that each set consists of a median, and then, based lemmas 2 and 3 the costs are equal. In stage III we show that if R and RS are not disjoint, then actually R is contained in RS, and the two costs are equal.

(Stage I) Suppose that R does not contain a median. Then, by condition 4 it is not a singleton. We will show that conditions 3 and 4 cannot both be satisfied. Let the shortest path between a median and R be $m, n, \cdots, l, k$, where $k \in R$. Since m is a median and each node performs at least one operation, $\sum_{j \in V(T_{l-R})} (\#W_j + \#R_j) > \sum_{j \in V(T_{R-l})} (\#W_j + \#R_j)$. By condition 2 of the lemma $\sum_{j \in V(T_{l-R})} \#R_j \leq \sum_{j \in V(T_{R-l})} \#W_j$.

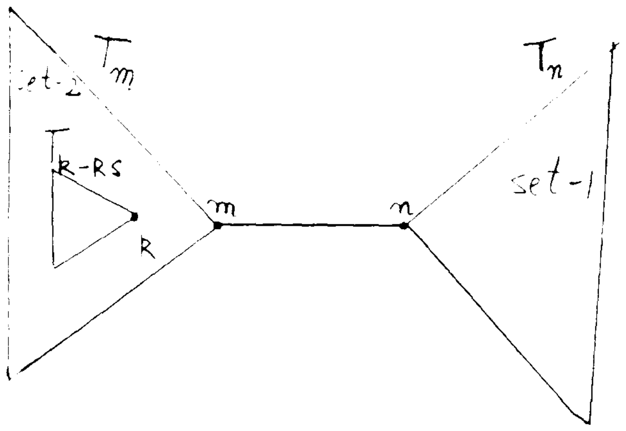
Together, the last two inequalities imply that

(0) $\sum_{j \in V(T_{l-R})} \#W_j > \sum_{j \in V(T_{R-l})} \#R_j$. Since R is not a singleton, let g be an R-fringe that is different than k. It is easy to see that $V(T_{g-R})$ is a proper subset of $V(T_{R-l})$ and $V(T_{l-R})$ is a proper subset of $V(T_{R-g})$. Therefore, based on inequality (0) $\sum_{j \in V(T_{R-g})} \#W_j > \sum_{j \in V(T_{g-R})} \#R_j$, which in turn contradicts condition 3 of the lemma.

(Stage II) The set R contains a median by Stage I, and the set RS contains a median by step 1 of the algorithm *TREE–RS*. Assume that R and RS are disjoint. Consequently, each one of the two sets contains a different median. Let the two medians be m and n, and suppose without loss of generality that $m \in RS$ and $n \in R$.

Suppose, by way of contradiction, that RS contains more than one node. Let k be a node of RS that is different than m, and is a leaf of the subtree induced by RS. We will show that the cost of RS can be reduced by replacing k by n, contradicting the minimality of $cost(RS,A)$. Let us consider how this change will affect the cost of RS. For this purpose we will partition the nodes of the tree network into three disjoint subsets, illustrated in Fig. 2: set-1 consisting of the nodes of T_n, set-2 consisting of the nodes of T_m that are not in T_{k-RS}, set-3 consisting of and the nodes of T_{k-RS}.

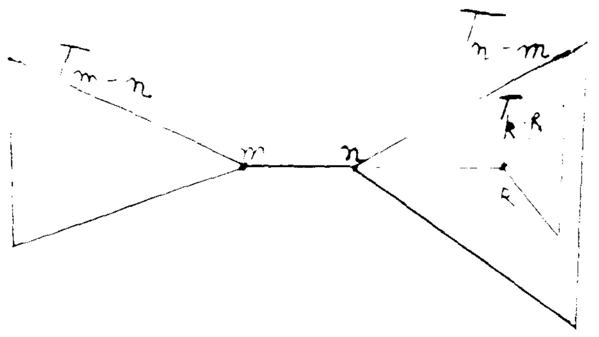
Figure 2:



The cost of each read originating in set-1 will decrease by one (since it will access the replica on n instead of m), and the cost of each write originating in set-1 will decrease by one (since it will not have to access k). The cost of each read originating in set-2 remain the same (since it will access m), and the cost of each write originating in set-2 will remain the same (since it will access m instead of k). The cost of each read originating in set-3 will increase by one (since it will access the neighbor of k in RS, instead of k), and the cost of each write originating in set-3 will increase by one (since it will have to access n). Since m and n are medians, and T_{k-RS} is a proper subset of T_m, there are strictly more operations originating in set-1 than in set-3. Consequently, the replacing k by n will reduce $cost(RS,A)$.

Suppose now, again by way of contradiction, that R contains more than one node. Let k be a node of R that is different than m, and is a leaf of the subtree induced by R. We will show that the four conditions in the lemma cannot be simultaneously satisfied. For this purpose we will consider three subtrees, T_{m-n}, T_{n-m} and T_{k-R}, illustrated in Fig. 3. Note that T_{k-R} is included in T_{n-m}.

Figure 3:



By condition 2 of the lemma,

(1) $\sum_{j \in V(T_{m-n})} \#R_j \leq \sum_{j \in V(T_{n-m})} \#W_j$ (m is the neighbor of R).

By condition 3 of the lemma,

(2) $\sum_{j \in V(T_{m-n})} \#W_j + \sum_{j \in V(T_{n-m})-V(T_{k-R})} \#W_j \leq \sum_{j \in V(T_{k-R})} \#R_j$ (k is the R-fringe node).

Note that the left hand side of inequality (2) is actually $\sum_{j \in V(T_{R-k})} \#W_j$.

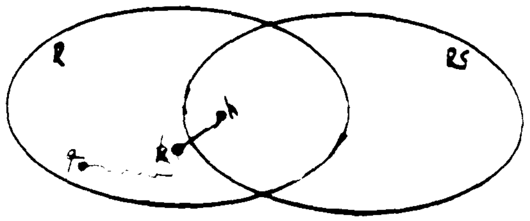
If there is any write issued in some node of $T_{n-m}-T_{k-R}$ (i.e. in a node that is in T_{n-m} but not in T_{k-R}), then, by inequality (2), $\sum_{j \in V(T_{m-n})} \#W_j < \sum_{j \in V(T_{k-R})} \#R_j$, and this, combined with inequality (1), implies that $\sum_{j \in V(T_{m-n})} (\#R_j + \#W_j) < \sum_{j \in V(T_{n-m})} (\#R_j + \#W_j)$ (since T_{k-R} is a subset of T_{n-m}). But, by lemma 2, this contradicts the fact that m is a median. Thus, $\sum_{j \in V(T_{n-m})-V(T_{k-R})} \#W_j = 0$. Since n is in $V(T_{n-m})-V(T_{k-R})$, and it performs at least one operation, this operation must be a read. Combined with (2), this implies that

(3) $\sum_{j \in V(T_{m-n})} \#W_j < \sum_{j \in V(T_{n-m})} \#R_j$.

By inequalities (1) and (3) $\sum_{j \in V(T_{m-n})} (\#R_j + \#W_j) < \sum_{j \in V(T_{n-m})} (\#R_j + \#W_j)$. But this contradicts the fact that m is a median.

(Stage III) Suppose that R and RS are not disjoint. We will show first that R is contained in RS. Suppose that there is a node of R that is not in RS. Since RS induces a connected subtree of the network, there must be leaf, q of the subtree induced by R, that is not in RS (see Fig. 4)

Figure 4:



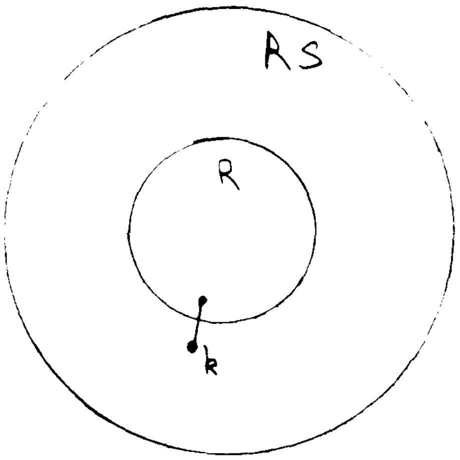
Consider the shortest path $q=x_1, x_2, \cdots, x_{n-1}=k, x_n=h$ from q to a node h in $R \cap RS$. During the construction of RS by TREE-RS, the node k must have been examined in step 3. Since k was not added to RS, $\sum_{j \in V(T_{k-RS})} \#R_j < \sum_{j \in V(T_{RS-k})} \#W_j$. Then, for every $n-1 \le i \le 1$: $\sum_{j \in V(T_{x(n-i)-x(n-i+1)})} \#R_j < \sum_{j \in V(T_{x(n-i+1)-x(n-i)})} \#W_j$.

In other words, for every i, if we remove the edge between $x(i)$ and $x(i+1)$, then there are more writes issued in the subnetwork that contains $x(i+1)$ than there are reads issued in the subnetwork that contains $x(i)$. This is true in particular for $i=1$. But then, since q is an R-fringe node, this contradicts condition 4 of the lemma.

Finally, we will show that if $R \subseteq RS$ then the costs of the two sets are equal. This claim is obviously true is $R = RS$, therefore suppose that $R \subset RS$. We will show that all the nodes of $RS - R$ can be added to R without changing the cost of R, and this will conclude the proof.

Let k be a node in $RS - R$ that is a neighbor of R (see Fig. 5).

Figure 5:



By condition 3 of the lemma, we know that

(4) $$\sum_{j \in V(T_{k-R})} \#R_j \le \sum_{j \in V(T_{R-k})} \#W_j .$$

We will show that actually:

$$(5) \qquad \sum_{j \in V(T_{k-R})} \#R_j = \sum_{j \in V(T_{R-k})} \#W_j .$$

Suppose that all the nodes of $V(T_{k-R})$ that are in RS are removed from RS. This will affect the cost of RS as follows. The cost of all the reads performed in $V(T_{k-R})$ will increase by one, and the cost of all the writes performed in $V(T_{R-k})$ will decrease by one; the cost of any other operation will not change. Since the cost of RS is minimum,

$$(6) \qquad \sum_{j \in V(T_{k-R})} \#R_j \geq \sum_{j \in V(T_{R-k})} \#W_j .$$

By combining (4) and (6) we obtain (5). Equation (5) implies that the addition of k to R will not change the cost of R, for the following reason. This addition will increase the cost of each write performed in $V(T_{R-k})$ by one, and will decrease the cost of each read performed in $V(T_{k-R})$ by one; the cost of any other operation will not change. This proof can be repeated verbatim, if after the addition of k to R the set $RS - R$ is not empty. []

Theorem 1: Let S' be a schedule that is mapped by the algorithm ADAPTIVE-REPLICATION to a configured-schedule, S''. Let c be the number of processors in the tree network, and let S be a subschedule of S'', in which each processor of the network performs at least one operation, and let A be the read-write pattern in S. If ADAPTIVE-REPLICATION is stable on S with stability scheme R, then $cost(R,A)$ is higher than the cost of an optimal replication scheme for A by at most c.

Proof: We will show that either R is optimal for A, or, there is a replication scheme R', that is obtained by applying a transformation to R, such that R' satisfies: 1. it is optimal for A, and 2. $cost(R,A) - cost(R',A) \leq c$. The initial replication scheme, consisting of the whole set of processors, clearly induces a connected subgraph of the network. Since ADAPTIVE-REPLICATION only adds neighbors of the current replication scheme and drops fringe nodes, R induces a connected subgraph of the network. If R also satisfies conditions 2, 3, and 4 of lemma 4, then it is optimal for A, and the proof is complete. Otherwise, we will show that the conditions of lemma 4 are satisfied by the replication scheme R', that is obtained from R by the application of a transformation, called *Transform*. It consists of the addition of some neighbors, and removal of some R-fringe nodes.

Description of *Transform*:

Suppose that condition 2 of lemma 4 is not satisfied for R. Then we execute the following, step 1 of *Transform*. There is at least one neighbor k of R for which:

$$(10) \qquad \sum_{j \in V(T_{k-R})} \#R_j > \sum_{j \in V(T_{R-k})} \#W_j .$$

Transform calls k an R-add node. Now we will make an important observation about k. Denote by i the neighbor of k that is in R. Remember that i performs the expansion test during the operations in the schedule S, and since ADAPTIVE-REPLICATION is stable on S, we know that i does not change the replication scheme. Since inequality (10) holds, the sequence of requests at node i during the schedule S, when considering only the reads from $V(T_{k-R})$ and writes from $V(T_{R-k})$, is the following: $q = r,w,r,......,r,w,r$. In other words, the sequence q starts and ends with a read from $V(T_{k-R})$, and the number of such reads exceeds the number of writes from $V(T_{R-k})$ by one.

Suppose that condition 3 of lemma 4 is not satisfied for R. Then we execute the following, step 2 of *Transform*. There is at least one R-fringe node, k, for which:

$$(11) \qquad \sum_{j \in V(T_{k-R})} \#R_j < \sum_{j \in V(T_{R-k})} \#W_j .$$

Transform calls k an R-drop node. Now we will make the following observation about k. It performs the contraction test during the operations in the schedule S, and since ADAPTIVE-REPLICATION is stable on S, we know that k does not change the replication scheme. Since inequality (11) holds, the sequence of requests at node k during the schedule S, when considering only the reads from $V(T_{k-R})$ and writes from $V(T_{R-k})$, is the following: $q = w,r,w,......,r,w$. In other words, the sequence q starts and ends with a write from $V(T_{R-k})$, and the number of such writes exceeds the number of reads from $V(T_{k-R})$ by one.

Suppose that condition 4 of lemma 4 is not satisfied for R. If step 1 of *Transform* defined any R-add nodes then we go step 4 of *Transform*. Otherwise we execute the following, step 3 of *Transform*. R is a singleton, say $\{k\}$. There is one neighbor of k, n, for which during S the number of operations requested by n is bigger than the number of operations initiated at some node of $V(T_{k-n})$. *Transform* calls k an R-drop node, and n and R-add node. Now we will make the following observation about k and n. k performs the switch and expansion tests during the operations in the schedule S, and since ADAPTIVE-REPLICATION is stable on S, we know that k does not change the replication scheme. Then the sequence of requests at node k during the schedule S is the following: $q = a,b,a,\ldots\ldots,b,a$, where the a's are operations issued by n, and the b's are operations initiated at some node of $V(T_{k-n})$. In other words, in the sequence q the number of a's exceeds the number of b's by one.

Finally, in step 4 of *Transform* we define R' as follows. subtree of T that contains $\{i \cup R\}$. If the set $S \cup \{all\ R-add\ nodes\} - \{all\ R-drop\ nodes\}$ is nonempty, let us call it R'. Otherwise R must consist of two nodes, say k and m, both of which are R-drops. Then, let us define $R' = \{k\}$.

This completes the description of *Transform*.

Now observe that the removal of each R-drop node from R decreases $cost(R,A)$ by one (by the definition of an R-drop node). Similarly, the addition of each R-add node to R decreases $cost(R,A)$ by one (by the definition of an R-add node). To complete the proof of the theorem, left to prove is that R' is optimal, and we will do so by showing that the four conditions of lemma 4 are satisfied for R'.

(Case 1.) Suppose that R' contains more than one node. It can be deduced from the definitions that an R-add and an R-drop cannot be neighbors. Thus, R' is connected. To show that condition 2 of lemma 4 is satisfied for R', consider a node i that is a neighbor of R'. If i is also a neighbor of R, then it obviously satisfies: $\sum_{j \in V(T_{i-R'})} \#R_j \le \sum_{j \in V(T_{R'-i})} \#W_j$. Otherwise, i is either a neighbor of an R-add node, or i is an R-drop node. It can be verified from the definition of an R-add node, and the fact that each node performs at least one operation, that in both cases: $\sum_{j \in V(T_{i-R'})} \#R_j \le \sum_{j \in V(T_{R'-i})} \#W_j$. Thus condition 2 of lemma 4 is satisfied for R'.

To show that condition 3 of lemma 4 is satisfied for R', consider a node i that is an R'-fringe node. If i is also an R-fringe, then it obviously satisfies: $\sum_{j \in V(T_{i-R'})} \#R_j \ge \sum_{j \in V(T_{R'-i})} \#W_j$. Otherwise, i is either an R-add node, or i is in R and it becomes an R'-fringe node as a result of being a neighbor of some R-drop node. In the first case it can be verified from the definition of an R-add node, that: $\sum_{j \in V(T_{i-R'})} \#R_j \ge \sum_{j \in V(T_{R'-i})} \#W_j$.

In the second case this inequality holds by the definition of an R-fringe node, and by the fact that each node performs at least one operation. Thus condition 3 of lemma 4 is satisfied for R'.

(Case 2.) Suppose that R' is a singleton set, $\{k\}$. We have to show that conditions 2 and 4 of lemma 4 are satisfied.

(Case 2.1) Suppose that R is also a singleton set, $\{m\}$.

(Case 2.1.1) Suppose that $R' = R$. Then, since k performs the switch test, and since R is the stability scheme on S, and since *Transform* does not change R, k is a median. Since k performs the expansion-test, and since no R-add nodes were defined by *Transform*, condition 2 of lemma 4 is satisfied for R'.

(Case 2.1.2) Suppose that $R' \neq R$. Then k is a unique R-add and m is a unique R-drop. By the way these were defined by *Transform*, it is easy to see that condition 4 of lemma 4 is satisfied for $\{k\}$. Condition 2 of lemma 4 is satisfied for the following reason. Since the expansion test performed by m does not change R, then:

(12) $$\sum_{j \in V(T_{k-m})} \#R_j \le \sum_{j \in V(T_{m-k})} \#W_j + 1.$$

Consider l, a neighbor of k that is different than m. Since $V(T_{k-l}) \subset V(T_{m-k})$, and since k performs at least one operation, then: $\sum_{j \in V(T_{l-k})} \#R_j \le \sum_{j \in V(T_{k-l})} \#W_j$. Consider m. Since $R \neq R'$ and both are singletons,

(13) $$\sum_{j \in V(T_{k-m})} \#O_j \le \sum_{j \in V(T_{m-k})} \#O_j - 1,$$

where $\#O_j$ denotes the number of operations (read or write) issued at node j. Combining (12) and (13) we obtain: $\sum_{j \in V(T_{m-k})} \#R_j \le \sum_{j \in V(T_{k-m})} \#W_j$.

(Case 2.2) Suppose that R is not a singleton set (i.e. R' is obtained by deleting nodes from R). It can be shown that condition 2 of lemma 4 is satisfied for $\{k\}$, by the same line of reasoning as used in case 1. For showing that condition 4 is satisfied, we will analyze three subcases.

(Case 2.2.1) Suppose that R consists of two nodes: k, and another node m, that is an R-drop. Since k is not an R-drop,

(14) $$\sum_{j \in V(T_{k-m})} \#R_j \ge \sum_{j \in V(T_{m-k})} \#W_j.$$

Since m is an R-drop,

(15) $$\sum_{j \in V(T_{k-m})} \#W_j = \sum_{j \in V(T_{m-k})} \#R_j + 1.$$

Based on (14) and (15),

$\sum_{j \in V(T_{k-m})} \#O_j > \sum_{j \in V(T_{m-k})} \#O_j$, and therefore condition 4 of lemma 4 is satisfied for neighbor m that is an R-drop.

Now consider another neighbor of k, say g (g is not in R since R consists of two nodes, and it is also not an R-add since $R' = \{k\}$. Since g is not an R-add,

(16) $$\sum_{j \in V(T_{g-k})} \#R_j \le \sum_{j \in V(T_{k-g})} \#W_j.$$

Since m is an R-drop, $\sum_{j \in V(T_{k-m})} \#W_j = \sum_{j \in V(T_{m-k})} \#R_j + 1$. Additionally, $V(T_{g-k}) \subset V(T_{k-m})$, and $V(T_{m-k}) \subset V(T_{k-g})$, and k performs at least one operation. Consequently,

(17) $$\sum_{j \in V(T_{g-k})} \#W_j \le \sum_{j \in V(T_{k-g})} \#R_j.$$

Based on (16) and (17) condition 4 of lemma 4 is satisfied for g.

(Case 2.2.2) Suppose that R consists of two nodes: k and m, both of which are R-drops. Since m is an R-drop, equation (15) holds, and since k is an R-drop

(18) $$\sum_{j \in V(T_{m-k})} \#W_j = \sum_{j \in V(T_{k-m})} \#R_j + 1.$$

Based on (15) and (18),

(19) $$\sum_{j \in V(T_{m-k})} \#O_j = \sum_{j \in V(T_{k-m})} \#O_j.$$

Based on (19) it is easy to see that for any other neighbor of k, say l, $\sum_{j \in V(T_{k-l})} \#O_j \ge \sum_{j \in V(T_{l-k})} \#O_j$. Thus condition 4 of lemma 4 is satisfied for R'.

(Case 2.2.3) Suppose that R consists of k plus two or more nodes, all of which are R-drops. Consider an R-drop neighbor of k, say m. Equation (15) holds for m. Now consider another R-drop neighbor of k, say n.

(20) $$\sum_{j \in V(T_{k-n})} \#W_j = \sum_{j \in V(T_{n-k})} \#R_j + 1.$$

Since $V(T_{n-k}) \subset V(T_{k-m})$, and since each node performs at least one operation,

(21) $$\sum_{j \in V(T_{m-k})} \#W_j \le \sum_{j \in V(T_{k-m})} \#R_j.$$

Based on (15) and (21), $\sum_{j \in V(T_{k,m})} \#O_j > \sum_{j \in V(T_{m,k})} \#O_j$. Thus condition 4 of lemma 4 is satisfied for each R-drop node. If there are other neighbors of k, that are not R-drops, the proof that for such a neighbor, say g, $\sum_{j \in V(T_{k,m})} \#O_j \geq \sum_{j \in V(T_{m,k})} \#O_j$ is identical to the one in case 2.2.1. []

Theorem 1 indicates that when ADAPTIVE replication stabilizes, then the difference between the cost of the stability scheme and the optimal cost is at most the number of nodes in the network. This holds regardless of the length of the schedule S, and the cost of its read-write pattern. Obviously, the higher this cost, the more insignificant the difference becomes.

4. Comparison to Relevant Research and Future Work

There are two main purposes for data-replication: performance and reliability. In this paper we have addressed the performance issue. Research works that concentrate on reliability usually devise more sophisticated policies than the read-one-write-all we assumed here (see for example [AE2, GB]).

Most performance-oriented works address the static problem, namely establishing a priori a replication scheme that will remain fixed at run-time. This is called the file-allocation problem, and it has been studied extensively in the literature (see [DF] for a survey). The goal is to optimize the communication cost, as well as other parameters, such as storage costs ([C, ML]), communication channels capacity ([MR]), or the communication network topology ([IK]). Existing works on the file-allocation problem assume a naive write policy, in which the total communication cost of a write is simply the sum of the communication costs between the sender and each one of the receivers (nodes of the replication scheme). This means, for example, that in the network T of the introduction, the cost of the write from A to C and to D is 2 + 3. Obviously, this is a waste for tree networks, where there is a unique path between every pair of nodes. Furthermore, in [WM] we have shown that the write policy affects the optimal replication scheme, namely, a scheme that is optimal for the naive write may not be optimal for the tree write, and vice-versa. Consequently, this paper is novel since it addresses the problem of dynamic (vs. static) replication, and it assumes the proper write policy for tree-networks. The need for dynamic replication was pointed out in [GS].

In future work we intend to extend the ADAPTIVE-REPLICATION algorithm to arbitrary networks; an important intermediate step is ring-topologies. We would also like to extend the algorithm to policies that are different than the read-one-write-all (e.g. [AE2, GB]). We have an additional goal that is motivated by the extensive work on online algorithms that is currently being carried out in the theoretical computer science community (e.g. [BLS, BS, CL]). We would like to show that ADAPTIVE-REPLICATION is competitive in the sense that given any sequence of read-write requests, the ratio *cost(optimal replication scheme) / cost(ADAPTIVE-REPLICATION)* is bounded by some constant.

References

[AE1] D. Agrawal and A. El Abbadi, "Storage Efficient Replicated Databases", IEEE Trans. on Data Engineering, 2(3), Sept. 1990.

[AE2] D. Agrawal and A. El Abbadi, "Efficient Techniques for Replicated Data Management", Proc. of the Workshop on Management of Replicated Data, IEEE CS Press, 1990.

[BLS] A. Borodin, N. Linial, M. Saks "An Optimal Online Algorithm for Metrical Task Systems", Proc. STOC, 1987.

[BS] D. L. Black and D. D. Sleator "Competitive Algorithms for Replication and Migration Problems", manuscript, 1989.

[CL] M. Chrobak and L. L. Larmore "An Optimal Online Algorithm for k Servers on Trees", manuscript, 1990.

[C] R. G. Casey, "Allocation of Copies of a File in an Information Network", Proc. 1972 Spring Joint Computer Conference, AFIPS, 1972.

[DF] L. W. Dowdy and D. V. Foster, "Comparative Models of the File Assignment Problem", ACM Computing Surveys, **14** :2, 1982.

[GB] H. Garcia-Molina and D. Barbara, "How to assign Votes in a Distributed System", Journal of the ACM, **32** :4, 1985.

[GS] B. Gavish and O. Sheng, "Dynamic File Migration in Distributed Computer Systems" Communication of the ACM, **33** :2, 1990.

[IK] K. B. Irani and N. G. Khabbaz, "A Methodology for the Design of Communication Networks and the Distribution of Data in Distributed Supercomputer Systems", IEEE Transactions on Computers, **C-31** :5, 1982.

[ML] H. L. Morgan and J. D. Levin "Optimal Program and Data Location in Computer Networks", Communications of the ACM, **20** :5, 1977.

[MR] S. Mahmoud and J. S. Riordon, "Optimal Allocation of Resources in Distributed Information Network", ACM Transactions on Database Systems, **1** :1, 1976.

[WM] O. Wolfson and A. Milo, "The Multicast Policy and Its Relationship to Replicated Data Placement", ACM Transactions on Database Systems, **16** :1, 1991.