PLACEMENT OF REPLICATED DATA IN
COMPUTER NETWORKS

by

A. Milo and O. Wolfson

Technical Report #473

November 1987

We address the problem of determining an allocation scheme for replicated data in a distributed database, with the purpose of minimizing required communication. An allocation scheme establishes the number of copies of each data-item, and at which processors of a given computer network the copies should be located. The problem for general networks is shown NP-Complete, but we provide efficient algorithms to obtain an optimal allocation scheme for three common types of network topologies. They are completely-connected, tree, and ring networks. We also propose a new method by which a processor in a computer network should write all the copies of a replicated data item. This method leads to surprising results concerning the allocation schemes.
1. Introduction

The two main purposes of replicating data in distributed databases are improved performance, and improved reliability. In this paper we concentrate on the performance issue, and also discuss how reliability should be addressed in our framework. Replicated data usually improves performance because it enables a data-read to be performed locally or at a "close-by" site. On the other hand, replicated data increases the overhead of a data-write because it has to be transmitted to all processors which store replicas. Therefore, in order to obtain a performance gain the number of data-item copies, and their placement in the computer network should be carefully determined.

In this paper we address the problem of placing replicas of a data-item, i.e. files, relations, or data-blocks, in a given computer-network for minimizing the number of information messages. These are the messages that carry the data-items, and place the bulk of the load on the communication network (control messages such as read-requests, acknowledgments, and subtransaction-initiation are ignored). Our aim is to suggest an algorithm that given a communication network, and read-write ratio for a data-item, will output an allocation scheme that minimizes the number of information messages carrying the data-item. The communication network is modeled by an undirected graph, and we are careful to consider that transmitting the data-item from processor \( a \) to processor \( b \) requires a number of messages equal to the distance in the network between \( a \) and \( b \). The read-write ratio is assumed common to all processors in the network. The allocation scheme output consists of the number of replicas, and at which processors they should be located. We establish that the problem of determining an optimal allocation scheme is \( NP-Complete \) (even for our simple problem definition), but can be solved efficiently (algorithms are provided) for the following widely-used network topologies: completely interconnected, tree, and ring. Our computer network is homogeneous in the sense that the message-costs on all communication links are equal.

As mentioned, in addition to improved performance, data-replication improves system-reliability, and more specifically availability; if a processor that stores a replica fails, then another replica of the data-item stored at a different processor, continues to be available for reads and writes in the network. Therefore, often performance has to be optimized subject to the constraint that the number of replicas in the network is not lower than some threshold, \( t \). This threshold is calculated a priori by means outside the scope...
of this paper, such as the failure-probability of a processor. For each topology for which we find an optimal allocation scheme, we also show how to find the optimal allocation scheme in which the number of replicas is at least \( r \). This scheme may be different than the absolute optimal allocation scheme.

This work is distinguished from the previous extensive research on the general file-allocation problem (see [DF] for a survey) by several features. First, our problem formulation is simple, and therefore for specialized topologies we are able to provide optimal solutions in polynomial time. Specifically, the only two parameters to the problem are the network topology, and the read-write ratio. As a consequence, we do not optimize overall performance; we do however optimize one very important factor which determines performance, namely message traffic. In this sense we take what is called in [CMP] the user viewpoint. Namely, we assume a predefined configuration and network topology. Other works attempt to optimize the communication costs as well as storage costs ([C, ML]), communication channels capacity ([MR]), or the communication network topology ([IK]). Additional advantages of our simple model are: independence of many parameters that are hard to estimate or change frequently (such as file-size or storage capacity); and independence of the allocation scheme of one data-item from another, enabling incremental configuration of the system.

A second feature distinguishing the present work is that we assume that the individual-processor algorithm, at run-time, optimizes message traffic. For example, suppose that three processors, \( a, b \) and \( c \) are connected in a string, namely \( a \) is connected by a bidirectional link to \( b \), and \( b \) is connected to \( c \). Assume that \( a \) writes a data-item which is replicated at processors \( b \) and \( c \). In other models \( a \) is assumed to transmit the item separately to \( b \) and to \( c \). This generates one information message for the first transmission and two for the second. But the transmission from \( a \) to \( c \) traverses through \( b \) anyway, because of the network topology. So a simple message-optimization that we assume \( a \) to make at run-time, is to send the item to \( b \), and "ask" \( b \) to propagate it to \( c \). This will save one information message. A similar approach was proposed in [SW] to minimize the number of control messages required for transaction commitment. Indeed, it makes sense that if we determine the allocation scheme with the purpose to minimize messages, then each processor does so at run-time. Therefore, we argue that the communication cost of a multicast transmission is not simply the sum of the communication costs between the sender and each one
of the receivers. On this subtle point the present paper differs from all other works on the subject, leading to surprising results concerning the allocation schemes. It turns out that the replicas should be clustered in neighboring processors of the network, rather than being "spread out" evenly to allow close access for each processor.

The rest of the paper is organized as follows. The minimal communication cost of a read and a write for a fixed allocation scheme is determined in section 2. The optimal residence set problem is defined and shown in \textit{NP-Complete} for general networks in section 3. In section 4 we present the positive results concerning completely-connected, tree, and ring networks. In section 5 we discuss the effect that reliability considerations have on the results presented in section 4. In section 6 we conclude and discuss future work.

2. Read and Write Message-Costs.

Read and write operations for logical entities (or data-items) are issued at each processor in a computer network. Each such operation is eventually translated into zero or more information messages transmitted in the network; they carry the entity to or from the processors storing the physical replicas of the logical entity. In this section we establish the minimal number of network messages required for an arbitrary read and write by a processor. The read case is simple, the write case slightly more involved. We start with some definitions. A \textit{communication network}, or \textit{network} for short, is an undirected connected graph, \( G=(V,E) \). \( V \) represents a set of processors, and an edge in the network between processors \( v \) and \( w \) represents a bidirectional communication link between them. Given a network we define a \textit{residence set} to be a subset of \( V \). It represents the processors where some arbitrary fixed entity is replicated. We assume that reading of an entity by a processor is implemented by transferring the closest replica to it. Therefore, for a given network and residence set the \textit{read cost} of a processor \( v \), denoted \( r_v \), is the length (in edges) of the shortest path in the network between \( v \) and a processor of the residence set. It represents the number of information messages required for the entity transfer. Obviously, if \( v \) is in the read set the read cost is zero.

Next we establish the write cost for a processor, given a residence set, \( R \). We assume that processor \( v \in V \) writes the logical entity, and call it the \textit{writer}; it may or may not belong to the residence set. The
processors in $R \cup \{v\}$ are the participants in the write protocol, and are denoted by $P$. A write instance is a directed graph, $I=(P,A)$. Each arc of $A$ represents a replica transfer between two participants, and its cost is the shortest path in the network between the processors at its endpoints. Since the entity sent by the writer reaches every other participant we require that there is a path in $I$ from $v$ to each processor in $R$.

The write-instance cost is the total cost of its arcs. We are interested in establishing the minimal cost of a write instance. Clearly, for this purpose the only instances to be considered are acyclic. A possible algorithm which sends exactly the messages of a given acyclic instance, $I$, is the following (assuming $I$ is known to all participants): the writer sends the entity to its sons and each processor after receiving the entity forwards it to its sons and so on, until the entity reaches the leaves of $I$.

For the next proposition we need to define the distance graph, $D_G(\hat{V})$, for an arbitrary subset of processors $\hat{V} \subseteq V$. $D_G(\hat{V})$ is a complete weighted graph with the set of nodes $\hat{V}$. The weight of an edge between $j$ and $k$ in $D_G(\hat{V})$ is the length of the shortest path in $G$ between $j$ and $k$. Denote by $\text{mst}(D_G(\hat{V}))$ a minimum spanning tree of $D_G(\hat{V})$.

**Proposition 2.1:** Let $G$ be a network, $R$ a residence set, and $v$ a writer. Denote the set $R \cup \{v\}$ by $P$. Then the necessary and sufficient cost of a write instance is the total weight of $\text{mst}(D_G(P))$.

**Proof:** (necessary) By the definition of a write instance the underlying graph of any write instance is a connected subgraph of $D_G(P)$ that spans all the processors of $P$. The minimal weight of such a graph is the weight of $\text{MST}(D_G(P))$.

(sufficient) Given an $\text{MST}(D_G(P))$ we can build the required write instance whose cost is equal to the weight of $\text{MST}(D_G(P))$, by directing the edges of the $\text{MST}(D_G(P))$ to form a rooted tree, rooted at the writer. □

Therefore, given a residence set $R$ we define the write cost of processor $v$, denoted $w_v$, to be the total weight of $\text{mst}(D_G(P))$. Denote by $d_v$ the length of the shortest path in $G$ between $v$ and (some processor of) $R$. The next lemma will be used extensively in our proofs.

**Lemma 2.1:** Let $G=(V,E)$ be a network, $R$ a residence set, and $i \in V$ a processor. If $R$ induces a connected subgraph of $G$, then $w_i=d_i+|R|-1$. 
Proof: We prove first that any spanning tree of $D_G(R \cup \{i\})$ is of weight at least $d_i + |R| - 1$. The $|R| + 1$ nodes of the distance graph require $|R| + 1$ edges in order to form a spanning tree. The lightest edge that is connected to $i$ in $D_G(R \cup \{i\})$ is obviously of weight $d_i$. Each of the other $|R| - 1$ edges is at least of weight one, therefore there is no spanning tree whose weight is less than $d_i + |R| - 1$.

Now we prove that a spanning tree of such weight exists. $R$ induces a connected subgraph, thus, there are $|R| - 1$ edges of weight one, and this set of edges spans the nodes of $R$ in the distance graph. Also, there exists an edge of weight $d_i$ between node $i$ and some node of $R$. Therefore there exists a spanning tree whose weight is $d_i + |R| - 1$. □

3. The residence set problem

In this section we define the residence set problem, namely the problem of placing the replicas of a given data item to minimize overall message traffic in a computer network. Then its complexity is established.

We assume that the transaction processing load is balanced in the following sense. The number of data-item read operations per time unit, $\#R$, is equal at all the processors, and the same holds for the number of write operations per time unit, $\#W$. We denote the ratio $\#R/\#W$ by $\alpha$. In other words, for each write there are $\alpha$ reads at each processor. Given a residence set, $R$, and $\alpha$, the residence set cost, denoted $\text{cost}(R)$, is defined as $\sum_{i \in V} w_i + \alpha \cdot \sum_{i \in V} r_i$. We shall refer to the first sum in the expression as the total-write-cost, and to the last sum multiplied by $\alpha$, as the total-read-cost. Intuitively, $\sum_{i \in V} w_i + \alpha \cdot \sum_{i \in V} r_i$ represents the total communication cost when $R$ is the residence set and the time unit’s chosen such that $\#W = 1$. (or, it can be looked at as $\#W \cdot \sum_{i \in V} w_i + \#R \cdot \sum_{i \in V} r_i$ divided by $\#W$). We would like to find an optimal residence set, i.e. a residence set with the minimal cost. The residence set problem, denoted $RS$, is defined as follows:

Input: A communication network graph $G = (V,E)$, and two positive real numbers, $\alpha$ and $C$.

Question: Is there a residence set $R \subseteq V$, such that $\sum_{i \in V} w_i + \alpha \cdot \sum_{i \in V} r_i < C$?

Theorem 3.1: $RS$ is NP-Complete.

Proof: See Appendix.

Although the problem is NP-Complete in general, for certain input parameters it can be solved efficiently.
Theorem 3.2: Let \( G = (V,E) \) be a network and assume that the read-write ratio \( \alpha \), is bigger than \( |V|^{-1} \).

Then there is a unique optimal residence set, and it is the set of all processors, \( V \).

**Proof**: Suppose that \( R \subset V \) is an optimal residence set. Let \( k \) be a processor in \( V - R \), which has a neighbor in \( R \). It is easy to see that \( \text{cost}(R \cup \{k\}) < \text{cost}(R) \), as follows. The total read-cost for \( R \cup \{k\} \) is lower than the total read-cost for \( R \), by at least \( \alpha \). For each \( i \neq k \), \( w_i \) for \( R \cup \{k\} \) is higher than \( w_i \) for \( R \), by at most one. Therefore, the total write-cost for \( R \cup \{k\} \) is higher than the total write cost for \( R \), by at most \( |V|^{-1} \). Overall, since \( \alpha > |V|^{-1} \), \( \text{cost}(R) > \text{cost}(R \cup \{k\}) \), contradicting the optimality of \( R \). \( \square \)

In the next section we show that for certain common network topologies the problem can be solved efficiently for any read-write ratio.

4. Special topologies

4.1 Completely Connected Network

In this subsection we establish that in a complete-network (i.e. a clique), if \( \alpha \geq |V|^{-1} \), then the optimal residence set consists of one processor, and if \( \alpha < |V|^{-1} \), then the optimal residence set consists of all processors.

The proof for this fact is as follows. Given a residence set, \( R \), for every processor \( j \) in \( R \), \( r_j = 0 \); for every processor \( k \) that is not in \( R \), \( r_k = 1 \). Thus, \( \sum_{i \in V} r_i = |V| - |R| \). For every processor \( j \) in \( R \), \( w_j = |R| \cdot 1 \)

and for every processor \( k \) that is not in \( R \), \( w_k = |R| \). Thus, \( \sum_{i \in V} w_i = (|V| - |R|) \cdot |R| + |R| \cdot (|R| - 1) = (|V| - 1) \cdot |R| \).

\[ \text{cost}(R) = \sum_{i \in V} w_i + \alpha \cdot \sum_{i \in V} r_i = |R| \cdot (|V| - 1) = |V| \cdot \alpha. \] (1)

Expectedly, the actual residence set is irrelevant, only its size determines the communication cost. If \( \alpha < |V|^{-1} \) then the minimum of the \( \text{cost}(R) \) function is obtained when \( |R| = 1 \), and if \( \alpha \geq |V|^{-1} \) then the minimum is obtained when \( |R| = |V| \). Therefore, the size of the residence set as a function of \( \alpha \) is a step function with two values. Interestingly, an intermediate residence set is never optimal (regardless of \( \alpha \)). Obviously, the size of the optimal residence set can be computed in constant time.
4.2 Tree Network

In this subsection we will first present a simple algorithm, called TREE-RS, which determines the optimal residence set in a tree-network. Then we shall prove the correctness of the algorithm, and finally we shall show that the algorithm works in linear time.

In the algorithm TREE-RS we use the term median of a tree. The median is the node for which the sum of distances to the other nodes is minimal. Note that generally the median of a tree is different than its center. The latter is the node for which the maximal distance is minimal. Formally, in a tree let \( l_{vu} \) denote the length of the simple path between \( v \) and \( u \). A median is a node, \( m \), for which \( \sum_{u} l_{mu} \) is minimal. It can be shown that in a tree there are one or two medians \((IZ)\).

The algorithm TREE-RS is given as parameters a tree network, and a read-write ratio \( \alpha \). Starting from some median, it incrementally constructs the optimal residence set, RS, by adding a processor to RS if it does not increase the cost of the set. The algorithm colors a processor blue if it is not in RS, and red if it is in RS, or has not been checked yet.

```
TREE-RS \[ T(V,E), \alpha \]: /* Algorithm for finding the optimal residence set */
1. init: color all the processors of \( V \) by red; initialize RS to a median, \( m \).
2. while there exists a processor in RS with at least one red neighbor, \( j \), do:
3. if \( \text{cost}(RS \cup \{j\}) \leq \text{cost}(RS) \) then add \( j \) to RS; else color \( j \) by blue.
4. end while.
5. output: RS.
```

Next we prove that RS is an optimal residence set. The proof is concluded by Theorem 4.2.1, and uses 5 lemmas. The first states that the optimal residence set must induce a connected subgraph of the network.

**Lemma 4.2.0:** Assume that there exists an optimal residence set \( R \), which induces a disconnected graph in a tree network. Then there is another residence set, \( R' \), such that \( \text{cost}(R') < \text{cost}(R) \), and \( |R| > |R'| \).

**Proof:** The construction proceeds as follows. Since the graph induced by \( R \) is not connected, there must be at least two processors of \( R \), \( i \) and \( j \), such that if we denote the unique path between them by \( i, b_1,..., b_k, j \) for \( k \geq 1 \), then the \( b_i \)'s do not belong to \( R \). To obtain \( R' \) we add to \( R \) all the processors on this path. The total read cost for \( R' \) is less than the total read cost for \( R \), since \( R \subset R' \). The write cost of a pro-
cessor \( i \), for \( R' \) is equal to its write cost for \( R \), for the following reason. In a minimum spanning tree of \( D_O(R \cup \{i\}) \), the path between \( i \) and \( j \) can be replaced by the path \( i, b_1, \ldots, b_k, j \) to obtain an equal weight minimum spanning tree of \( D_O(R' \cup \{i\}) \).

Now, let \( R \) be a residence set that induces a connected subgraph of the tree-network, and assume that processor \( i \) 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 \( R \). It disconnects the network into two subtrees: a subtree that contains \( i \), denoted \( T_i \), and a subtree that contains \( R \), denoted \( T_R \). Figure 4.2.1 illustrates the description. Denote the processors of a graph \( G \) by \( V(G) \).

![Figure 4.2.1](image)

**Lemma 4.2.1:** \( \text{cost}(R \cup \{i\}) = \text{cost}(R) - \alpha \cdot |V(T_i)| + |V(T_R)| \).

**Proof:** Observe that adding \( i \) to \( R \) decreases the read cost of each processor of \( T_i \) by one, and does not change the read cost of \( T_R \)'s processors. Also, adding \( i \) to \( R \) increases the write cost of each processor of \( T_R \) by one, and does not change the write cost of \( T_i \)'s processors. Thus the lemma follows.

**Lemma 4.2.2:** Let \( R \) be a residence set that induces a connected subgraph of a tree-network. Assume that processor \( i \) is a neighbor of some processor in \( R \), and \( t_i \) is a subtree of the network such that \( i \in V(t_i) \) and \( V(t_i) \cap R = \emptyset \) (see figure 4.2.1). Then, \( \text{cost}(R \cup \{i\}) > \text{cost}(R) \) implies that \( \text{cost}(R \cup V(t_i)) > \text{cost}(R) \).
Proof: Consider how the addition of $V(t_i)$ to $R$ changes costs. It decreases the read cost of each processor of $T_i$ by at most $|V(t_i)|$, and does not change the read cost of $T_R$'s processors. It also increases the write cost of each processor of $T_R$ by $|V(t_i)|$ (by lemma 2.1), and does not decrease the write cost of $T_i$'s processors. Note that $\text{cost}(R \cup V(t_i)) = \text{cost}(R) - \text{[decrease in read cost]} + \text{[increase in write cost]}$. Thus, $\text{cost}(R \cup V(t_i)) \geq \text{cost}(R) - \alpha \cdot |V(T_i)| - |V(T_i)| + |V(T_R)| + |V(t_i)| = \text{cost}(R) + |V(t_i)| - \alpha \cdot |V(T_i)|$. Since it is given that $\text{cost}(R \cup \{i\}) > \text{cost}(R)$, then by lemma 4.2.1, $\alpha > 0$, and the lemma follows.

Lemma 4.2.3: Let $R$ and $R'$ be two residence sets that induce connected subgraphs, and assume that $R \subseteq R'$. Furthermore, assume that processor $i$ is a neighbor of some processor in $R$, and $i \in R'$ (see fig. 4.2.2). Then $\text{cost}(R \cup \{i\}) \leq \text{cost}(R)$ if and only if $\text{cost}(R' \cup \{i\}) \leq \text{cost}(R')$.

Proof: Straightforward from Lemma 4.2.1.

The last lemma indicates that for each median there is an optimal residence set which contains it. It uses the following median property (see [Z]). A node $m$ is a median of a tree $T$, if for each neighbor, $w$, of $m$ it is true that: if the edge $(m, w)$ is removed, then in the connected component which contains $w$ there are at most half the number of nodes in $T$.

Lemma 4.2.4: Let $m$ be a median, and $R$ a residence set which induces a connected subgraph of the tree
network. If \( m \notin R \) then there is another residence set, \( R' \), such that \( R' \) induces a connected subgraph of the tree network, and \( m \in R' \), and \( |R'| = |R'| \), and \( \text{cost}(R') \leq \text{cost}(R) \).

**Proof**: Intuitively, the proof shows that if \( R \) is "shifted" towards \( m \) then the cost cannot increase. Formally, consider the unique path in the network from \( m \) to the closest member of \( R \). Denote by \( i \) the last node on this path which is not in \( R \) (see fig. 4.2.3). If \( R \) contains more than one member, then denote by \( j \) a leaf of the subtree induced by \( R \), which is not a neighbor of \( i \). Otherwise let \( j \) be the unique member of \( R \). Let \( R' = (R \cup \{i\}) - \{j\} \). We shall show that \( \text{cost}(R') \leq \text{cost}(R) \). Denote by \( T_i \) the subtree of the network which contains \( i \) and is obtained by removing the edge between \( i \) and its neighbor in \( R \). Similarly, \( T_j \) is defined (again, consult fig. 4.2.3). It is easy to verify that \( \text{cost}(R') = \text{cost}(R) + |V(T_j)| - |V(T_i)| + \alpha(|V(T_j)| - |V(T_i)|) \). The **median property** implies that \( |V(T_i)| \geq |V(T_j)| \), and therefore \( \text{cost}(R') \leq \text{cost}(R) \). The procedure can be repeated until \( m \) is contained in \( R' \). □

![figure 4.2.3.](image)

**Theorem 4.2.1**: For any tree-network and any read-write ratio, the residence set \( RS \) output by the algorithm \( TREE-RS \), is optimal.

**Proof**: We will prove that \( RS \) satisfies the following condition. It is a minimal cost residence set that contains \( m \), with a maximal number of processors. By lemma 4.2.4, a residence set which satisfies the condi-
tion is an optimal residence set. Assume that $RS$ does not satisfy the condition, and denote by $RS'$ a residence set which does so. This obviously implies that $RS \neq RS'$. The set $RS'$ induces a connected subgraph (by Lemma 4.2.0), and $RS$ induces a connected subgraph (by the way TREE-$_RS$ adds nodes to the residence set). We will analyze two cases.

**case 1:** $RS \not\subseteq RS'$. Observe that the graph induced in the network by $RS' - RS$ is a forest. Consider a tree, $t_i$, of this forest. It contains a processor, $i$, that has a neighbor in $RS$. Step 3 of TREE-$_RS$ must have been executed for $i$, but it did not add $i$ to $R$. In other words, adding $i$ would have increased the cost. From Lemmas 4.2.2 and 4.2.3 we can conclude that by removing the processors of $t_i$ from $RS'$ a lower cost residence set can be obtained. Contradiction to the minimality of $RS'$'s cost.

**case 2:** $RS \subseteq RS'$. Let $k$ be the first processor that the algorithm adds, such that $k \in RS$ and $k \not\in RS'$. Denote by $S$ the residence set that we have before adding $k$. Note that $S \neq \emptyset$ (because at least $m \in S$), and $S \subseteq RS'$. The algorithm adds $k$ to $S$, so $\text{cost}(S \cup \{k\}) \leq \text{cost}(S)$. The processor $k$ is a neighbor of some processor in $S$. By Lemma 4.2.3, $\text{cost}(RS' \cup \{k\}) \leq \text{cost}(RS')$. If $\text{cost}(RS' \cup \{k\}) < \text{cost}(RS')$ then $RS'$ is not an optimal residence set, and if $\text{cost}(RS' \cup \{k\}) = \text{cost}(RS')$ then $RS'$ does not have a maximal number of processors. □

Now consider the time complexity of TREE-$_RS$. A median can be found by using the median property. It necessitates establishing in advance, for each edge $e$ of the tree, how many nodes are in each connected component, if $e$ is removed. This can be done in linear time, and then finding the median simply involves scanning the nodes. In step 2 of the algorithm every processor of the tree-network is examined at most once. For the red processors step 3 is performed. By using Lemma 4.2.1, which indicates how the addition of a processor to the residence set changes costs, step 3 can be performed in constant time. Specifically, $\text{cost}(R \cup \{j\}) \leq \text{cost}(R)$ if and only if $\alpha \cdot |V(T_j)| \geq |V(T_R)|$. Therefore, the optimal residence set can be found in linear time.

### 4.3 Ring

The main result of this subsection is providing in theorem 4.3.1 a formula to compute the optimal residence set in constant time. First we prove 4 lemmas, enabling us to conclude that for any size ring, and for any
read-write ratio, there is an optimal residence set that induces a connected subgraph of a ring network.

We refer to a connected subgraph of a ring network as a string. The first lemma enable us to speak subsequently in more intuitive terms of strings, rather than the distance-graph.

Lemma 4.3.1: Let $G$ be a ring network, $R$ be a residence set, and $i$ be a processor of $G$. Then $w_i$ equals to the number of edges in the shortest string of $G$ that contains all the processors of $R \cup \{i\}$.

Proof: Obvious.

Given a residence set, $R$, a string without nodes in $R$ is a hole.

Lemma 4.3.2: For any residence set $R$ which induces three or more strings in the network, there is a residence set having a lower cost, for any read-write ratio.

Proof: Denote by $H$ the set of processors each of which is not in $R$, and is not in the two biggest holes of $R$. Since there are more than two holes, $H$ is not empty. Consider the residence set, $R'$, which is $R \cup H$ (we fill all but the two biggest holes). Since $R'$ is a proper superset of $R$, the total cost of reads for $R'$ is less than for $R$. Consider an arbitrary processor, $i$. If $i$ is not in the biggest hole of $R$, then the length of the shortest string of $G$ that contains all the processors of $R \cup \{i\}$ is the length of the ring except the biggest hole. If $i$ is in the biggest hole of $R$ than the length of the shortest string is the length of the ring except part of the biggest hole, or the length of the ring except the second biggest hole. In both cases $w_i$ for $R'$ equals to $w_i$ for $R$, thus the total cost of writes has not changed. Therefore $\text{cost}(R') < \text{cost}(R)$ for any $\alpha$.

Lemma 4.3.3: If the read-write ratio $\alpha > 1$, then for any residence set that induces two strings, there is a residence set that has a lower cost.

Proof: Let $R$ be a residence set that induces two strings (and two holes). Denote by $H_s$ and $H_b$, the sets of processors of the smaller hole and of the bigger hole, respectively. Consider the residence set, $R'$, which is $R \cup H_s$ (see fig. 4.3.1). We will show that $\text{cost}(R') < \text{cost}(R)$. 
For each processor in $R \cup H_b$ the read costs using $R$ and $R'$ are equal; the read cost of the processors of $H_s$ is zero using $R'$ and nonzero using $R$. First we compute the total decrease in the cost of reads, when substituting $R'$ for $R$.

case 1: $|H_s|$ is even.

Then there are two processors in $H_s$ at distance $i$ from $R$, for each $1 \leq i \leq \frac{|H_s|}{2}$. The total read cost decreases by
\[
2 \cdot \sum_{i=1}^{\frac{|H_s|}{2}} i = \frac{|H_s|}{2} \left(\frac{|H_s|}{2} + 1\right).
\]

(case 2: $|H_s|$ is odd.

Then there are two in $H_s$ processors at distance $i$ from $R$, for each $1 \leq i \leq \frac{|H_s| - 1}{2}$, and there is one processor at distance $\frac{|H_s| + 1}{2}$. The total read cost when using $R'$ instead of $R$ decreases by
\[
2 \cdot \sum_{i=1}^{\frac{|H_s| - 1}{2}} i + \frac{|H_s| - 1}{2} = \left(\frac{|H_s| + 1}{2}\right)^2.
\]

Now consider the write costs. The write cost of a processor in $R \cup H_s$ has not changed, because by Lemma 4.3.1 it equals to the length of the shortest string that contains $R \cup H_s$, and for any processor in $R \cup H_s$ it is the same string (i.e. the ring minus the biggest hole) for $R$ or $R'$. In $H_b$ however, there are
two types of processors (see fig. 4.3.2).

The first type (fig. 4.3.2a) are processors for which a shortest string that contains them and \( R \), contains the whole ring except some processors of \( H_b \). For these processors the shortest string has not changed, and by lemma 4.3.1 their write cost has not changed. Therefore, if \( H_b \) contains only processors of the first type, then the total cost is decreased for \( R' \) compared to \( R \), and the proof is completed.

The second type of processors in \( H_b \) (fig. 4.3.2b) are processors for which the shortest string contains the whole ring, except the processors of \( H_s \). For each such processor the shortest string that contains it and the processors of \( R' \) is longer. Let \( i \) be the distance of such a processor from \( R \). Then, a processor of \( H_b \) is of type two if \( i > |H_b| - 1|H_s| \); otherwise it is of type one. Denote by \( \Delta l \) the increase in the total write cost, when substituting \( R' \) for \( R \). The following two cases give an upper bound on \( \Delta l \). From this upper bound, the fact that \( \alpha > 1 \), and formulas (3) and (4), the lemma follows.

**case 1:** \( |H_b| \) is even.

Then there are two processors at distance \( i \) for each \( |H_b| - 1|H_s| < i < \frac{|H_b|}{2} \), and

\[
\Delta l = 2 \sum_{i=|H_b| - 1|H_s| + 1}^{\frac{|H_b|}{2}} \left( |H_s| + 1 - |H_b| \right) = (1 + |H_s| - \frac{|H_b|}{2})(|H_s| - \frac{|H_b|}{2}).
\]

Given that \( |H_b| \geq |H_s| \), if \( |H_s| \)
is even then \( \Delta I \leq \frac{|H_b|}{2} \left( 1 + \frac{|H_b|}{2} \right) \), and if \( |H_b| \) is odd then \( \Delta I \leq \left( \frac{|H_b|+1}{2} \right)^2 \).

case 2: \( |H_b| \) is odd.

Then there are two processors at distance \( i \), for each \( |H_b| - |H_s| < i \leq \frac{|H_b|+1}{2} \), and one processor at distance \( \frac{|H_b|+1}{2} \). It can be easily verified that \( \Delta I = (2 \cdot \sum_{i=|H_b|-|H_s|+1}^{\frac{|H_b|+1}{2}} (|H_s|+i-|H_b|))+(|H_s| \frac{|H_b|+1}{2}) = (\frac{|H_s|}{2} \frac{|H_b|-1}{2})^2 \). Again, given that \( |H_b| > |H_s| \), if \( |H_s| \) is even then \( \Delta I \leq \frac{|H_s|}{2} \left( 1 + \frac{|H_s|}{2} \right) \), and if \( |H_s| \) is odd then \( \Delta I \leq \left( \frac{|H_s|+1}{2} \right)^2 \). \( \square \)

**Lemma 4.3.4:** Let the read-write ratio \( \alpha \leq 1 \), and \( R \) be a residence set which induces two strings, and denote the smallest hole by \( H_s \). Then there is another residence set \( R' \) that induces at most two holes, and \( \text{cost}(R') \leq \text{cost}(R) \), and \( |R'| = |R| \), and the smallest hole it induces, \( H'_s \), is of a smaller size than \( H_s \).

**Proof:** We shall define \( R' \) to be the "shift" of the long string induced by \( R \), closer by one position to the short string (thus reducing \( |H_s| \) by one). Denote by \( S_b, S_s \) the big and small strings induced by \( R \), respectively. Denote by \( n_b \) the processor of \( H_s \) which is closest to \( S_b \), and by \( n_s \) the processor of \( S_b \) which is closest to \( H_s \) (see Fig. 4.3.3).

![figure 4.3.3.](image)

Let \( R' = S_s \cup (S_b-\{n_b\}) \cup \{n_s\} \). It is easy to calculate that the total read-cost is higher for \( R' \) than for \( R \), by \( \alpha \left( \frac{|H_b|}{2} \right) - \left( \frac{|H_s|}{2} \right) + 1 \). Consider now write costs. Since \( |H'_b| = |H_b| + 1 \) the write cost of a
processor of \( S_s \cup H_s \cup (S_b)-(n_b) \) is lower by one for \( R' \) than for \( R \). For \( n_b \), the write-cost is equal for the two residence sets. Left is only to evaluate the change in write costs for the processors of \( H_b \). If all the processors of \( H_b \) are of type one (see proof of Lemma 4.3.3), then the write cost of \( \left[ \frac{|H_b|}{2} \right] \) processors is lower by one for \( R' \) than for \( R \). Overall, the total write cost is lower for \( R' \) than for \( R \) by

\[
|S_s| + |H_s| + |S_b| + \left\lfloor \frac{|H_b|}{2} \right\rfloor - 1; \quad \text{since} \quad \alpha \leq 1
\]

\[
cost(R') - cost(R) \leq \left\lfloor \frac{|H_b|}{2} \right\rfloor - \left\lfloor \frac{|H_a|}{2} \right\rfloor + 1 - |S_s| - |H_s| - |S_b| + 1 - \left\lfloor \frac{|H_a|}{2} \right\rfloor < 0, \quad \text{and the lemma follows.}
\]

Suppose now that there are processors of type two in \( H_b \) (i.e. processors for which their distance from \( R \) is more than \( |H_b| - |H_s| \)). Then the write cost of the \( |H_b| - |H_s| \) processors of type one is lower by one for \( R' \) than for \( R \); the write cost of the \( |H_b| - 2 \cdot |H_s| \) processors of type two is higher by one for \( R' \). Then the total write cost is lower for \( R' \) than for \( R \) by

\[
|S_s| + |H_s| + |S_b| - 1 + |H_b| - |H_s| - |H_s| + |H_b| = |S_s| + |S_b| + |H_b| + 1 - 2 \cdot |H_s|.
\]

Overall, since \( \alpha \leq 1 \),

\[
cost(R') - cost(R) \leq \left\lfloor \frac{|H_b|}{2} \right\rfloor - \left\lfloor \frac{|H_a|}{2} \right\rfloor + 1 - |S_s| - |S_b| - 2 \cdot |H_b| - 1 + 2 \cdot |H_s| \leq 0.
\]

(Note that we assumed that processors do not change type when going from \( R \) to \( R' \). If they do, it is because their write cost can be decreased further, and our inequalities obviously continue to hold.)

\[ \square \]

**Corollary 4.3.1:** If the read-write ratio \( \alpha \leq 1 \), then for any residence set \( R \) that induces two strings, there is another residence set \( R' \) that induces only one string, and \( cost(R') \leq cost(R) \), and \( |R| \leq |R'| \).

**Proof:** Immediate from Lemma 4.3.4, since the size of the smaller hole can iteratively be reduced to zero.

\[ \square \]

By Lemma 4.3.3 and Corollary 4.3.1 we conclude that for each ring, and for each read-write ratio, there exists an optimal residence set with at most one hole; in other words, there exists an optimal residence-set that induces a connected subgraph of the network. Since the ring is symmetric, the residence set location is irrelevant. The next theorem provides a formula for computing the optimal residence set size.
Theorem 4.3.1: Let \( n \) be the size of the ring, and \( \alpha \) be the read-write ratio. Then the cardinality of the optimal residence set equals to \( \frac{n \cdot (\alpha - 1)}{\alpha + 1} + 1 \).

Proof: If \( R \) is a string of size \( k \), then for \( v \in R \) the cost \( w_v = k - 1 \) and the cost \( r_v = 0 \); for \( v \notin R \) at distance \( i \) from \( R \), \( w_v = i + k - 1 \) and \( r_v = i \). Summing up we obtain:

\[
\text{cost}(R) = n \cdot (k-1) + \frac{\alpha + 1}{4} \cdot [(n-k+1)^2 - c],
\]

where \( c = 1 \) if \( n - k \) is even, and \( c = 0 \) if \( n - k \) is odd. In both cases, the derivative of \( \text{cost}(R) \) with respect to \( k \) is \( n + \frac{1}{2} \cdot (\alpha + 1)(k-n-1) \). This indicates that the minimum of \( \text{cost}(R) \) is obtained when

\[
k = \frac{n \cdot (\alpha - 1)}{\alpha + 1} + 1.
\]

5. Reliability Considerations

As mentioned in the introduction, often the number of processors in the allocation scheme should not fall below some threshold, \( t \). Therefore, although the optimal allocation scheme contains, for example, one processor, this may be unacceptable for reliability reasons. Consequently, we are often interested in the \( t-RS \) problem: Given a network, \( G = (V,E) \) a read-write ration, \( \alpha \), and a reliability threshold, \( t < |V| \), what is the residence set \( R \subseteq V \) such that \( |R| \geq t \) and \( \text{cost}(R) \) is minimal? The problem remains NP-complete (we have proved it NP-complete for \( t=1 \)) but for the special topologies discussed in Section 4, it can be solved efficiently.

Consider first a completely connected network. If the cardinality of the optimal residence set is less than the threshold \( t \), it means that \( \alpha < |V| - 1 \), and that the optimal residence set is of size one. But then, based on formula (1) in subsection 4.1, it is easy to see that the solution to the \( t-RS \) problem is any subset of \( t \) processors.

Next, consider a ring network. Based on Lemma 4.3.3 and on Corollary 4.3.1, we can disregard residence sets which induce more than one connected component. Then the formula developed in the proof of Theorem 4.3.1 for \( \text{cost}(R) \) as a function of the size of the residence set, is a polynomial of rank two having a minimum value. Consequently, since \( t \) is bigger than the optimal residence set size (otherwise the problem is trivial), the solution to the \( t-RS \) problem is a string of size \( t \).
Finally, we consider a tree network. For this network topology the solution is slightly more complicated. We present a linear time algorithm, TREE-tRS, which provides a solution to the t-RS problem. Given a tree $T$, a residence set $R$ which induces a connected subgraph of $T$, and a node $j$ of $T$ which is not in $R$, but is a neighbor of some node $i$ in $R$ assume that we remove the edge $(i,j)$ from $T$; then remember that we denote by $T_j$ the connected component which includes $j$, and by $T_R$ the other one. The algorithm is formally given below, but intuitively, it starts with $RS$, the optimal residence set output by TREE-RS, and iteratively adds to it the neighbor $j$ for which $|T_j|$ is minimal. Obviously, each addition of a node to $RS$ increases the communication cost.

$$\text{TREE-tRS} [T(V,E), \alpha, j];$$

1. initialize $R_j$ to $RS$, the optimal residence set output by TREE-RS $[T(V,E), \alpha]$;
2. while $|R_j| < t$ do;
3. add to $R_j$ the neighbor $j$ for which $|T_j|$ is maximal;
4. end;
5. output $R_j$.

Next we shall prove that the set $R_j$ output by TREE-tRS indeed provides a solution to the t-RS problem. First we need some notation. Denote by $RS_t$ a solution to the $t$-RS problem which includes the median, $m$, used by TREE-RS (according to Lemma 4.2.4 there is such), and is of maximal size. Obviously, $|RS_t| \geq t$.

By Lemma 4.2.0, $RS_t$ induces a connected subtree of the network. Denote by $RS$ the optimal residence set obtained by the procedure TREE-RS, when starting from $m$. We obviously suppose that $|RS| < t$, otherwise $RS$ is a solution to the $t$-RS problem.

**Lemma 5.1:** $RS \subset RS_t$.

**Proof:** Suppose that $RS \subset RS_t$. Note that $RS \cap RS_t \neq \emptyset$, because $m$ belongs to both sets. Therefore, consider a processor $i \in RS - RS_t$, which is a neighbor of a processor in $RS_t$ (see Fig. 5.1). By Lemma 4.2.3, and the fact that $TREE-RS$ added $i$ to the residence set, it is clear that $\text{cost} (RS \cap RS_t \cup \{i\}) \leq \text{cost} (RS \cap RS_t)$. Then, again by Lemma 4.2.3, $\text{cost} (RS_t \cap \{i\}) \leq \text{cost} (RS_t)$. But if $\text{cost} (RS_t \cap \{i\}) < \text{cost} (RS_t)$ it contradicts the cost minimality of $RS_t$, and if $\text{cost} (RS_t \cap \{i\}) = \text{cost} (RS_t)$ it contradicts the maximality of the size of $RS_t$. □
Theorem 5.1: \( \text{cost} (RS_t) = \text{cost} (R_t) \).

Proof: Note that \( R_t \cap RS_t \neq \emptyset \), because \( m \) belongs to both sets. We shall analyze three cases.

Case 1: Suppose that \( RS_t \subset R_t \). But this is impossible, since then \( |RS_t| < t \).

Case 2: Suppose that \( R_t \subset RS_t \).

Consider some leaf \( l \) in the subtree induced by \( RS_t \), such that \( l \in R_t \). By Lemma 4.2.1, 
\[
\text{cost}(RS_t - \{l\}) = \text{cost}(RS_t) + \alpha|V(T_i) - |V(T_{RS_t - \{l\}})|.
\]
Since \( RS \subset R_t \) clearly \( l \in RS \). Consider the path from \( l \) to \( RS \), and denote by \( T_i \) the last node on this path which is not in \( RS \). By the way \( TREE-RS \) operates clearly \( \alpha|V(T_i)| - |V(T_{RS})| < 0 \). But since \( RS \) includes a median, \( |V(T_i)| \leq |V(T_i)| \), which in turn implies that \( \text{cost}(RS_t - \{l\}) < \text{cost}(RS_t) \). Since \( l \in R_t \), in \( RS_t - \{l\} \) there clearly are at least \( t \) nodes, contradiction to the cost minimality of \( RS_t \).

Case 3: \( RS_t \subset R_t \) and \( R_t \subset RS_t \).

Denote \( H = R_t \cap RS_t \). By Lemma 5.1, \( H \supseteq RS \). Let \( l \) be some leaf of the subgraph induced by \( RS_h \), such that \( l \in H \) (see Fig. 5.2).
By Lemma 4.2.1, \( \text{cost}(RS_{1}-(l)) = \alpha \cdot |V(T_{l})| - |V(T_{RS_{1}-(l)})| + \text{cost}(RS_{1}) \). Let \( k \) be some processor in \( R_{1}-RS_{1} \), which is a neighbor of \( H \) (see Fig. 5.2). If \( l \) is not a neighbor of \( H \), then \( |V(T_{l})| < |V(T_{k})| \), because by step 3 of \( \text{TREE-}tRS \), \( |V(T_{l})| \leq |V(T_{k})| \). (The definition of \( l_{f} \) is illustrated in Fig. 5.2.) Therefore, \( \text{cost}((RS_{1}-(l)) \cup \{k\}) < \text{cost}(RS_{1}) \), and this contradicts the cost minimality of \( RS_{1} \). Consequently, \( l = l_{f} \), i.e. \( l \) is a neighbor of \( H \); also any neighbor \( n \) of \( H \) in \( R_{1}-RS_{1} \), has in its subtree, \( T_{n} \), at least as many nodes as \( T_{l} \). If there is such a neighbor, \( k \), which has more nodes, then again \( \text{cost}((RS_{1}-(l)) \cup \{k\}) < \text{cost}(RS_{1}) \). Therefore, every neighbor \( n \in R_{1}-RS_{1} \), has in its subtree, \( T_{n} \), exactly as many nodes as \( T_{l} \). Therefore, \( R_{1}-H \) cannot have any nodes which are not neighbors of \( H \), because step 3 of \( \text{TREE-}tRS \) would have added \( l \) to \( R_{1} \), before any such node. Therefore, every member of \( (R_{1}-H) \cup (RS_{1}-H) \) is a neighbor of \( H \), and has an equal number of nodes in its subtree. The size of \( R_{1} \) is \( t \), therefore \( |R_{1}-H| \leq |RS_{1}-H| \). Any node of \( RS_{1}-H \) increases the cost communication of the residence set, thus, since \( RS_{1} \) is cost optimal \( |R_{1}-H| = |RS_{1}-H| \). Therefore, clearly \( \text{cost}(R_{1}) = \text{cost}(RS_{1}) \).

6. Discussion

In this paper we first proposed a new method by which a processor should write all replicas of a data item for minimizing communication. The processor should construct a minimum spanning tree of what we
called the distance graph, and then propagate the data item along its edges. The read, as usual, is carried out from the closest replica. Then we showed that determining an allocation scheme, or residence set, for minimizing overall communication is \textit{NP-Complete} in networks modeled by general graphs. However, we provided constant time algorithms for determining the optimal residence set in completely-connected networks and rings, and a linear time algorithm for determining the optimal residence in tree-networks. Extensions for the algorithms in case reliability constraints exist, were provided.

It turns out that for the special network topologies discussed in this paper, for any read-write ratio, the data-item should be placed in one connected component of the network. This is counter-intuitive because one would expect that as the read-write ratio grows, the replicas should be spread out throughout the network to allow a close read for each processor, rather than clustering them in one connected component. The reason for this apparent anomaly is the following. Our write policy favors clustering more than other policies. In order to require dispersion, the read-write ratio has to grow to such extent, as to make the whole set of processors the optimal residence set - again one connected component.

If the write cost of a single processor were simply the sum of distances to all replicas (namely, a naive write policy), then the optimal allocation scheme would have been different. For example, consider a ring with six processors, and a read write ratio of two (in other words, each processor performs two reads and one write per time unit). If the processors were conducting a naive write policy, then the optimal allocation scheme is 2-symmetric, i.e., place replicas at two processors at a distance of three edges from one another. The cost of this scheme is 26 messages per time unit. For the mst write policy, the optimal allocation scheme is a 3-string, i.e., place replicas on a string of three processors (24 messages). The 3-string is not optimal for the naive write policy (35 messages), and the 2-symmetric is not optimal for the mst write policy (26 messages).

Next, we would like to demonstrate that the communication cost improvement obtained by using the allocation schemes proposed in this paper is significant. Consider, for example, in a ring network of $n$ processors, how the proposed residence set compares with a trivial residence set consisting of all $n$ processors. Denote the number of messages per time unit for this trivial residence set by $\text{cost}_n$, and the number of messages per time unit for the optimal residence set (Theorem 4.3.1) by $\text{cost}_{opt}$. Then, if the read-write ratio is
\( \alpha, \text{cost}_{\text{opt}} = \frac{n^2 \alpha}{\alpha + 1} \), and \( \text{cost}_n = n(n-1) \). Furthermore, \( \frac{\text{cost}_n}{\text{cost}_{\text{opt}}} \rightarrow 1+\frac{1}{\alpha} \). Therefore, if for example \( \alpha = 2 \), then as the number of processors grows, our proposed residence set is 33\% better than the trivial one. Similarly, one can show that if we consider another trivial residence set, consisting of one processor, then \( \frac{\text{cost}_1}{\text{cost}_{\text{opt}}} \rightarrow \frac{(\alpha+1)^2}{4\alpha} \), and for \( \alpha = 4 \), a 36\% gain is realized.

An additional remark is that although the optimal set for the discussed topologies induces a connected graph, this is not necessarily the case in a general network. For example, for the network in fig 6.1 and for \( \alpha = 1.8 \) the unique optimal residence set is \( \{4,8\} \).

As far as future work is concerned, much remains to be done. First, it would be interesting to generalize our results to other network topologies, e.g. the hypercube. Second, bounded error approximations for general networks should be investigated. Third, networks without load balancing, i.e., different numbers of reads and writes for different processors, should be considered. Fourth, what happens when communication time in addition to communication cost is an issue (a similar analysis was carried out in [SW])? Finally, it would be interesting to generalize the results to majority-voting access schemes, as opposed to read-one-write-all.

References


Proof of Theorem 3.1 It is easy to see that $RS \in NP$. Guess a subset $R \subseteq V$, find $w_i$ and $r_i$ for each $i \in V$, and verify that $\sum_{i \in V} w_i + \alpha \cdot \sum_{i \in V} r_i < C$. Obviously, this can be done in polynomial time.

Next we show that $RS$ is $NP$-Hard. This is done by transforming the Steiner Tree (ST) problem to $RS$. In ST the input consists of a graph $G'=(V',E')$, a subset $X \subseteq V'$ and a positive integer $B < |V'|$. The question is whether there exists a subtree of $G'$ that includes all the nodes of $X$, and such that its number of edges is no more than $B$. We shall assume without loss of generality that $1 < |X| < B < |V'|$.

Given an instance of the ST problem we construct an instance of the RS problem as follows. The graph $G$ consists of $G'$, with every node $u \in X$ connected to a "crown" of $|V'|^3$ new nodes: $u_1, u_2, \ldots, u_{|V'|^3}$. For example, in fig. 3.1b there is graph constructed from the graph of fig. 3.1a, where $X=(a, b, c)$. Let $\alpha = B$ and $C = 2 \cdot |V'|^3 \cdot |X| - (B + 1)$.

We claim that there exists a solution to the ST problem if and only if there exists a solution to the RS problem. In the course of the proof we shall refer to the the nodes of $G$ in $G'$ as old nodes, and to the nodes of the "crowns" as new nodes.

(only if) Assume that there exists a Steiner tree, $T$, in $G'$ whose weight is no more than $B$. We choose as
the residence set $R$ all the nodes of $T$, and show that it constitutes a solution to the RS problem. Note that for every new node, say $k$, $r_k = 1$ and for every old node, $j$, $r_j \leq |V'|$. Therefore, the total cost for reading is $\sum_{i \in V} r_i \leq |V'|^3 \cdot |X| + |V'|^2$. By lemma 2.1, for every new node, $k$, $w_k \leq B + 1$. The distance of every old node from $R$ is at most $|V'| - B - 1$, and $|R| \leq B + 1$, so $w_j \leq |V'|$. Therefore, the total cost for writing is $\sum_{i \in V} w_i \leq |V'|^3 \cdot |X| \cdot (B + 1) + |V'|^2$.

Totaling, the cost is: $\sum_{i \in V} w_i + \alpha \sum_{i \in V} r_i \leq |V'|^3 \cdot |X| \cdot (B + 1) + |V'|^2 + \cdot (|V'|^3 \cdot |X| + |V'|^2) \leq (1)$

$|V'|^3 \cdot |X| \cdot (B + 1) + |V'|^3 < (2)$ $2 \cdot |V'|^3 \cdot |X| \cdot (B + 1) = C$.

(1) $|V'| > B$.

(2) $|X| > 1$.

Thus, the residence set $R$ is a solution to the RS problem.

(if) Assume now that the set $R$ constitutes a solution to the RS problem. We use 3 lemmas for this direction of the proof.

Lemma 3.0: There exists a residence set, $R'$, which does not contain any new nodes, and $\text{cost}(R') \leq \text{cost}(R)$.

Proof: To obtain $R'$ we will repeatedly perform one of the following two transformations, for each new node $u \in R$.

Case 1: The old node $v$, which neighbors $u$, belongs to $R$. Then drop $u$ from $R$. The read cost of every processor, except $u$, remains the same, and the read cost of $u$ increases by one. On the other hand, the write cost of every node of $G$, except $u$, decreases by one, and the write cost of $u$ remains the same. Since $\alpha < |V| - 1$ the cost of the new residence set is lower than $\text{cost}(R)$.

Case 2: The old node $v$, which neighbors $u$, does not belong to $R$. Then drop $u$ from $R$ and add $v$ to $R$. The read cost of any node, except $u$ and $v$, obviously does not increase due to the transformation, and the read cost of $u$ increases by one, and the read cost of $v$ increases by one. Overall, the total read cost does not increase. The total write cost also does not increase due to the transformation, because every path from a node to $u$ goes through $v$. □

To simplify notation we shall assume that $R$ does not contain any new nodes.
Lemma 3.1: If $R$ does not contain $X$, then there exists another residence set, $R'$, for which $\text{cost}(R') \leq \text{cost}(R)$, and $X \subseteq R'$, and $R'$ does not contain any new nodes.

Proof: Denote the nodes in $X-R$ by $\{1, 2, \ldots, k\}$. To obtain $R'$, first we choose for every node $h \in X-R$ a shortest path, $p_h$, from $h$ to a member of $R$. Then we add to $R$ all the nodes of all $p_h$'s, and denote this new residence set by $R'$. Obviously, $X \subseteq R'$ and $R'$ does not contain any new nodes. We denote by $d_h$ the length (in arcs) of $p_h$ for $h=1, 2, \ldots, k$. Let $d = d_1 + d_2 + \ldots + d_k$. The total-read-cost for $R'$ decreases by at least $B$ times $|V|^3 d_1 + |V|^3 d_2 + \ldots + |V|^3 d_k = |V|^3 d \cdot \text{(cost of reads by new nodes which are neighbors of X-R)}$, compared to the total-read-cost for $R$. We add no more than $d$ nodes to $R$ to create $R'$. Note that for each node $i$, $w_i$ increases by at most $d$. Since there are $|V| + |V|^3 |X|$ nodes in the graph, the total write cost of all the nodes increases by at most $d \cdot (|V| + |V|^3 |X|)$. 

Totaling, $\text{cost}(R') \leq \text{cost}(R) + (|V| + |V|^3 |X|) \cdot d - B \cdot |V|^3 d = \text{cost}(R) + |V|^3 d \cdot (|X| - B) + |V|^3 d \leq \text{cost}(R) + |V|^3 d \cdot (|X| - B + 1) \leq (1) \text{cost}(R)$

(1) $B > |X| - 1$. 

For the next lemma we need the following definition. Given an indirected graph, $G$, and a subset of the nodes, $X$, the minimal Steiner tree is a subgraph of $G$ which: 1) is a tree, and 2) contains the nodes in $X$, and 3) has a minimal number of edges among all subgraphs which satisfy the first two conditions.

Lemma 3.2: Let $G$ be a communication network graph, and $X$ a subset of its nodes. The weight of a minimal spanning tree (mst) of the distance graph on $X$ is not smaller than the weight of the minimal Steiner tree for $X$.

Proof: Obvious.

Assume that there is no Steiner tree with $B$ or less edges. Lemma 3.2 implies that the total weight of $\text{mst}(D_G(X))$ is at least $B + 1$. Thus, the cost of a write of a new node for $R'$ (Lemma 3.2), is at least $B + 2$ (one hop to the closest node in $X$, and then at least $B + 1$ hops to all the nodes of $X$ through the mst). Therefore the total write cost for $R'$ is at least $(B + 2) |X| \cdot |V|^3$. The total read cost for $R'$ is at least the read cost of the new nodes, i.e., $|X| \cdot |V|^3$. Totaling, $\text{cost}(R) \geq \text{cost}(R') = \sum_{i \in V} w_i + c \sum_{i \in V} r_i \geq (B + 2) |X| \cdot |V|^3 + B \cdot |X| \cdot |V|^3 = 2 \cdot |V|^3 |X| \cdot (B + 1) = C$

But this contradicts the fact that $R$ is a solution to the RS problem. □