An Analysis of the Average Message Overhead in Replica Control Protocols [1]

Debanjan Saha [2] Sampath Rangarajan [3] Satish K. Tripathi [2]

Abstract

Management of replicated data has received considerable attention in the last few years. Several replica control schemes have been proposed which work in the presence of both node and communication link failures. However, this resiliency to failure inflicts performance penalty in terms of communication overhead incurred. Though the issue of performance of these schemes from the stand point of availability of the system has been well addressed, the issue of message overhead has been limited to the analysis of worst case and best case message bounds. In this paper we derive expressions for computing the *average message overhead* of several well known replica control protocols and provide a comparison study of the different protocols with respect to both average message overhead and system availabilities.

Keywords: Replica Control, Quorum Consensus, Replicated Databases, Message Overhead, Availability, Update Synchronization.

[1] A preliminary version of this paper appeared in the Proceedings of the International Conference on Distributed Computing Systems, 1993.

[2] Department of Computer Science, University of Maryland at College Park, College Park, MD 20742.
[3] Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115.

1 Introduction

In a distributed system, replication of data improves system availability. Replication however burdens the system with added responsibility of maintaining consistency among different copies of the same data. A replica control protocol is essentially a protocol for "synchronizing" concurrent read and write operations on a replicated data object by different concurrent transactions. To ensure one-copy serializability [4], a read and a write operation to two different copies of the data (residing in two different "nodes" in the system) should not be allowed to execute concurrently. Also, two write requests to two different copies of the data should not be allowed to simultaneously update the copies.

The simplest replica control protocol is the *Majority Voting* protocol suggested by Thomas [11]. In this protocol a node can proceed with an operation only if it gets permission from a majority of other nodes in the system. This has been generalized by Gifford [6] to what is called the *Weighted Voting* protocol where different votes may be assigned to each node. A node needs a majority of the votes before it can proceed with an operation. *Availability* of a system running a replica control protocol is the probability that an operation initiated in the system will proceed in spite of node failures. For example, availability of a system using the majority voting protocol is given by the probability that at least a majority of the nodes in the system are functional (are not faulty). The communication overhead of these protocols is measured in terms of the number of messages that have to be exchanged before either permission is obtained for an operation to proceed or could be deemed unobtainable. For example, in the majority voting protocol, even if all the nodes are functional, the node initiating an operation has to send messages to $\lfloor \frac{N}{2} + 1 \rfloor$ nodes in the system requesting permission to proceed with an operation. Though the majority voting (and weighted voting) protocol has high availability, the communication overhead increases linearly with the number of nodes in the system, making this protocol unsuitable for large systems.

In replica control protocols based on *Quorum Consensus*, an operation initiated at a node in the system can proceed only if permission is granted by a group of other nodes. Such a group is called a quorum group. Collection of all quorum groups will constitute a quorum set. Permission from all the nodes in any one of the groups in the quorum set is sufficient for an operation to proceed, and to ensure synchronization, every pair of groups in the quorum set should have a non-empty intersection. Different quorum groups could be used for read and write operations and these are referred to as *read quorum* and *write quorum* respectively.

Several variants of the basic protocol have been suggested over the years including protocols for dynamically changing vote assignment and quorum sizes [7] [9].

There are some quorum consensus protocols that are based on logical organization of the nodes. Agrawal and Abbadi [1] proposed a protocol where the nodes are organized in a logical binary tree. Any set of nodes forming a path from the root to a leaf of this logical tree forms a quorum group. If a faulty node is encountered on this path, it can be substituted in the quorum group by nodes on two paths starting from the two children of the faulty node; and this is done recursively. Cheung, Ahamad and Ammar [2] suggest a logical two dimensional grid organization of the nodes. Each read quorum group contains at least one node from each column and a write quorum group consists of all nodes of one of the columns plus the nodes in a read quorum group. In the hierarchical quorum consensus protocol [8], the nodes are organized in a multilevel hierarchy with physical nodes at the lowest level of the hierarchy. Higher level "logical" nodes of the tree correspond to logical groups of the physical nodes. At any level except for the lowest one, a logical node needs permission from a majority of the nodes below it to call itself *available*. A quorum group consists of just the logical node at the root and thus a quorum is achieved when the logical node at the root is available.

Performance analysis of replica control protocols is a difficult problem involving trade-off between availability and communication overhead. Though the problem of availability analysis of systems using replica control protocols has been well addressed in the literature, analysis of communication overhead of these protocols has been limited to the best case and worst case situations. However, in a failure prone distributed system, the most useful and interesting measure of performance is the average case message overhead. In practical situations where there are failures in the system, the average or expected number of messages for obtaining a quorum may be very different from the worst case or the best case message overhead. Using the the best case and the worst case values in computing the overhead would be too optimistic and too pessimistic respectively. So, in order to obtain a realistic value for the expected message overhead of replica control protocols, it is essential to study their average case behavior in the presence of failures; and this is the focus of our paper. We compare the average case message overhead of some of the well known replica control protocols and also consider the trade-off between the average message overhead and availability.

In Section 2, we define the system model that we consider. In Section 3, we derive analytical expressions for the average message overhead and availability of five well known replica control protocols. Our results are discussed in Section 4 with numerical examples. Conclusions are presented in Section 5.

2 System Model

We consider a system of N nodes with each node maintaining a copy of the replicated object. A node that initiates a read or a write operation, requests permission from a set of other nodes by sending messages. If the initiating node does not receive a permission message from a node within a specified time-out period, it times-out on this request. We assume that a fault-free (available) node always gives permission (casts its vote) when it receives a request for permission. Further, we assume that the permission message reaches the initiating node before it times-out on the request message it sent. Thus, we ignore the issue of resource contention and do not take into consideration the queuing delay which may be incurred when several nodes compete for the same resource. Then, the system availability that we measure is based solely on the availability of the individual nodes. This assumption simplifies the analysis without being detrimental to our initial goal.

There is a basic difference in the failure models used in the best case and worst case calculations as compared to the one we use in our average case analysis. Previous work on worst case and best case overhead calculations have assumed that there are a bounded number of faults. With this assumption, the best possible pattern of failures and the worst possible pattern of failures are identified, and this information is used to compute the best case message overhead and the worst case message overhead respectively. Such a bounded fault-set model does not allow average case analysis. In this paper, in order to perform average case analysis, we assume a probabilistic model for failures. We assume that node failures are independent and use p to denote the steady-state probability that a node is non-faulty and is available to take part in the replica control procedure.

We consider the following well know replica control protocols.

- The majority voting protocol [11].
- The tree protocol [1].
- The grid protocol [2].

- The hierarchical quorum consensus (HQC) protocol [8].
- The RST protocol [10].

For each of the above protocols, we compute the average case message overhead and availability and consider the trade-off between the two. We further compare these performance parameters across the different protocols. The average number of messages exchanged as part of a replica control protocol is taken to be the number of messages exchanged on an average among the nodes in the system until *termination* of the protocol. A protocol may terminate successfully or unsuccessfully. A protocol terminates successfully when a node initiating a request to proceed with an operation gets permission to do so. For example, a quorum consensus based protocol terminates successfully if the initiating node gets the consensus of a quorum. When permission to proceed with the operation is not obtained by the node initiating a request, then the protocol is deemed to have terminated with failure.

We assume that the initiating node sends a minimum number of messages that will either lead to a successful termination of the protocol or will lead the initiating node to determine that the protocol will terminate unsuccessfully. For example, in a majority voting protocol, the initiating node will send request messages to only $\lfloor \frac{N}{2} + 1 \rfloor$ nodes first and only if permission is not obtained from some of them, will it send request messages to other nodes. Assume that at some point in time, the initiating node has received permission from $\lceil \frac{N}{4} \rceil$ nodes and it has timed-out on its other request messages. It needs permission from another $\lfloor \frac{N}{2} + 1 \rfloor - \lceil \frac{N}{4} \rceil$ nodes. So, it will send request messages to another $\lfloor \frac{N}{2} + 1 \rfloor - \lceil \frac{N}{4} \rceil$ nodes to whom it has not sent request messages before. Thus, the number of request messages that it would send at any time would be equal to the minimum number of votes that it needs at that time to obtain a majority. At some point, the initiating node may realize that it will not be able to obtain a majority even if it sent requests to all the nodes that it has not sent a request to yet; and at this point, it will abort the operation. A little thought shows that such a protocol would indeed minimize the average message overhead. It should also be noted that such a protocol may maximize the delay. That is, it may maximize the time from the initiation of the protocol to the time a majority is obtained or the operation is aborted. There are many different ways in which message exchange in a replica control protocol can be implemented. Depending on the implementation, there will be a trade-off between message overhead and delay. In this paper, we study the average message overhead (in the presence of failures) in an implementation where the number of messages used is minimum. Thus, we will assume such an implementation for all the protocols that we analyze.

3 Average Message Overhead Analysis

In this section, we derive analytical expressions for computing the average message overhead for the different protocols under consideration, as a function of the system size and the individual node availability. A brief description of each of the protocols is also provided.

3.1 Majority Voting Protocol

In majority voting, a node needs to receive permission from a majority, i.e. $\lfloor \frac{N}{2} + 1 \rfloor$ nodes to proceed with an operation. Even if all nodes in the system are functional, the node initiating a request needs to send messages to a minimum of $\lfloor \frac{N}{2} + 1 \rfloor$ other nodes in order to receive their permission. In a failure prone system, the average number of messages that needs to be sent by the initiating node will be more since not all the nodes from whom permission is sought will respond. As mentioned in the previous section, we assume a message exchange strategy which minimizes the number of messages.

The average number of messages that needs to be sent by the initiating node before either receiving permission from a majority of nodes or being able to determine that permission from a majority cannot be obtained is a function of both the system size and availability of a node. The following recursive expression where $m = \lfloor \frac{N}{2} + 1 \rfloor$ and $n = N$ provides an equation for computing this average message overhead.

$$M_{Maj}(p, m, n) = \begin{cases} 1 + (1-p)M_{Maj}(p, m, n-1) + \\ pM_{Maj}(p, m-1, n-1), \quad n \geq m > 0 \\ 0, \quad Otherwise. \end{cases} \tag{1}$$

$M_{Maj}(p, m, n)$ denotes the number of request messages to be sent by the initiating node in order to collect m votes from n other nodes when each node is available with a steady-state probability p. The derivation of the above relation is fairly simple. When $n \geq m > 0$ and the node executing the protocol sends a message to another node requesting its permission, with probability p the request is granted, in which case it needs to gather $m - 1$ votes from the remaining $n-1$ nodes. However, if the request is not acknowledged, probability of which is $1-p$, it needs to collect m votes from the remaining $n-1$ nodes. If $m > n$, no more messages need to be sent, since it is impossible to collect m votes from less than m nodes.

The *availability* of a system of N nodes, running *Majority Voting* protocol is nothing but the probability that at least $\lfloor \frac{N}{2} + 1 \rfloor$ out of the N nodes are functional. This can be evaluated combinatorially or could be evaluated using the following recurrence with $m = \lfloor \frac{N}{2} + 1 \rfloor$ and $n = N$.

$$A_{Maj}(p,m,n) = \begin{cases} (1-p)A_{Maj}(p,m,n-1)+ \\ pA_{Maj}(p,m-1,n-1), \quad n \geq m > 0 \\ 0, \quad Otherwise. \end{cases} \tag{2}$$

3.2 The Tree Protocol

In this protocol, nodes are organized as a logical binary tree of depth l. For convenience, we assume that $N = 2^{l+1} - 1$, so that the logical tree is a full binary tree. The initiating node that tries to form a quorum, sends request messages to all the nodes on a branch from the root down to one of the leaves. If any node on this branch fails to respond, in the next round, request messages are sent to all the nodes on a branch starting from a child of the failed node down to one of its leaves. This process is continued until permission messages are received from all the nodes to whom messages were sent in the previous round, or a situation is reached where it is determined by the initiating node that a quorum cannot be obtained. For a detailed description of this protocol, see [1]. The following recurrence holds for $M_{Tree}(p,d)$ which denotes the average number of request messages sent by the initiating node until the protocol terminates (either successfully or unsuccessfully), where the depth of the tree is d and p is the steady-state availability of a node.

$$M_{Tree}(d,p) = \begin{cases} 1 + p[2 - A_{Tree}(d-1,p)]M_{Tree}(d-1,p)+ \\ (1-p)[1 + A_{Tree}(d-1,p)]M_{Tree}(d-1,p), \\ \hfill d > 0 \\ 0, \quad Otherwise. \end{cases} \tag{3}$$

In the above equation, $A_{Tree}(d,p)$ is the availability of a tree of depth d when individual nodes are available with probability p. An expression for this parameter is derived later in this section.

The above recurrence can be derived as follows. Consider a tree of depth d. The first request message goes to the root. With probability p the response is positive (i.e. permission is granted by the root), in which case, $M_{Tree}(d-1,p)$ request messages need to be sent to one of the sub-trees. Let us assume that messages are sent to the nodes in the left sub-tree first. If a quorum is not obtained from the nodes in the left sub-tree, probability of which is $1 - A_{Tree}(d-1,p)$, $M_{Tree}(d-1,p)$ more messages have to be sent, this time to the nodes in the right sub-tree. Thus, the average or expected number of request messages sent by the initiating node when the root is available is given by

$$M_{Tree}(d-1,p) + (1 - A_{Tree}(d-1,p))M_{Tree}(d-1,p)$$

If the root does not respond, $M_{Tree}(d-1,p)$ messages are sent to the left sub-tree. Messages are sent to nodes in the right sub-tree only if the left sub-tree is available. Hence, the expected number of request messages sent when the root is unavailable is given by

$$M_{Tree}(d-1,p) + A_{Tree}(d-1,p)M_{Tree}(d-1,p)$$

The average message overhead for the protocol can be obtained by considering the above two mutually exclusive events and combining the above equations and simplifying the resulting equation as shown below.

$$1 + p[2 - A_{Tree}(d-1,p)]M_{Tree}(d-1,p)$$

$$+(1-p)[1 + A_{Tree}(d-1,p)]M_{Tree}(d-1,p)$$

The availability of the system running the tree protocol can be enumerated using the following recurrence.

$$A_{Tree}(d,p) = \begin{cases} (1-2p)A^2_{Tree}(d-1,p) \\ +2pA_{Tree}(d-1,p), \quad d > 1 \\ p, \quad Otherwise. \end{cases} \tag{4}$$

A tree whose depth d is greater than one ($d > 1$) is available if the root and one of the two sub-trees are available. If the root is not available, then the whole tree is available if and only if both the sub-trees are available. When the tree consists of a single node, the availability of the system is the same as that of the node and is given by p.

3.3 The Grid Protocol

In this protocol, nodes are organized in a logical rectangular grid with n_r rows and n_c columns such that $N = n_r \times n_c$, where N is the total number of nodes in the system. A read quorum consists of at least one node from each of the columns. That is, permission from at least one node in each column is required before a read operation can proceed. A write quorum consists of all the nodes in one of the columns and a read quorum. Again, p denotes the availability of a single node and $q = 1 - p$. Let $M_{GR}(p, n_r, n_c)$ and $M_{GW}(p, n_r, n_c)$ denote the average number of request messages sent by the node initiating a read/write request to obtain a read and a write quorum respectively. These parameters are computed as follows.

A node trying to obtain a write quorum uses the following strategy that minimizes the number of request messages sent. It first sends messages to nodes belonging to a randomly selected column one by one. There are three cases to consider.

- With probability p^{n_r} all of the nodes in the column respond; in this case, it only needs to achieve a read quorum from the remaining part of the grid and this would require on an average $M_{GR}(p, n_r, n_c - 1)$ request messages (an expression for $M_{GR}(p, n_r, n_c)$ is derived later).
- With probability q^{n_r}, none of the nodes in the column respond; in this case, it aborts the write operation since a write quorum can never be achieved in such a situation.
- Some (but not all) of the nodes in the column respond; in this case, it can proceed to achieve a write quorum from the sub-grid obtained by deleting the current column, and this would require an additional $M_{GW}(p, n_r, n_c - 1)$ request messages to be sent.

All that we need to consider now is the number of messages that is sent before one of the above three outcomes is confirmed. A jth message (to the jth node in the column) is sent only if the previous $j-1$ request messages resulted in permissions being obtained on all of them or permissions being obtained on none of them. This event can happen with probability $(p^{j-1}+q^{j-1})$. Thus, the average number of messages sent to nodes of one column before one of the above three outcomes is confirmed is give by

$$1+(p+q)+(p^2+q^2)+......+(p^{n_r-1}+q^{n_r-1})$$

After simplification, the above summation yields the following expression.

$$\frac{(1-p^{n_r+1}-q^{n_r+1})}{pq}-1$$

¿From the discussion above, we can write the following expression for the average number of request messages sent by a node trying to obtain a write quorum.

$$M_{GW}(p,n_r,n_c)=\begin{cases}\frac{(1-p^{n_r+1}-q^{n_r+1})}{pq}-1\\ +(1-p^{n_r}-q^{n_r})M_{GW}(p,n_r,n_c-1)\\ +p^{n_r}M_{GR}(p,n_r,n_c-1), \quad n_c,n_r>0\\ 0, \quad Otherwise.\end{cases} \tag{5}$$

A similar line of argument can be used to derive an expression for $M_{GR}(p,n_r,n_c)$. A node trying to obtain a read quorum sends messages to nodes belonging to the same column one by one until one of them responds with the permission, or there exists no more nodes to be probed. The probability that one of the nodes responds with the permission is given by $1-q^{n_r}$. In this case, the initiating node needs to achieve a read quorum from the remaining part of the grid and this would require on an average $M_{GR}(p,n_r,n_c-1)$ more request messages. The average number of request messages sent before receiving permission from a node in a a column evaluates to $(1-q^{n_r})/p$. When none of the nodes in a column respond, no more messages need to be sent since a read quorum cannot be achieved in such a situation. The following expression for $M_{GR}(p,n_r,n_c)$ can be derived from the above discussion.

$$M_{GR}(p,n_r,n_c) = \begin{cases} (1-q^{n_r})M_{GR}(p,nr,nc-1) \\ +(1-q^{n_r})/p, \quad n_r, n_c > 0 \\ 0, \ Otherwise. \end{cases} \tag{6}$$

The write availability (denoted by A_{GW}) of an $n_r \times n_c$ grid can be derived as follows. Let us consider the first column and the remaining portion of the grid separately. Now, there are three cases to consider.

- With probability p^{n_r}, all the nodes in the first column are available and in that case the write availability of the entire grid is $p^{n_r} A_{GR}(p, n_r, n_c - 1)$.
- With probability $(1 - q^{n_r})$, at least one node in the first column is available and in that case the write availability of the entire grid is $(1 - q^{n_r})A_{GW}(p, n_r, n_c - 1)$.
- When none of the nodes in the first column are available, the grid is not available.

Then the expression shown below for write availability follows from the theorem of total probability (where A_{GR} stands for the read availability and is derived later).

$$A_{GW}(p,n_r,n_c) = \begin{cases} (1-p^{n_r})A_{GW}(p,n_r,n_c-1)+ \\ p^{n_r}A_{GR}(p,n_r,n_c-1), \quad n_c > 0 \\ 0, \ Otherwise. \end{cases} \tag{7}$$

The expression for read availability can be derived similarly. Again, we break up the grid into two parts - the first column and the remaining portion of the grid. When none of the nodes in the first column are available, the entire grid is unavailable. When at least one node in the first column is available, probability of which is $1 - q^{n_r}$, the whole grid is available for a read if the remaining portion of the grid is read available. The expression shown below for the read availability follows immediately.

$$A_{GR}(p,n_r,n_c) = \begin{cases} (1-q^{n_r})A_{GR}(p,n_r,n_c-1), \quad n_c > 0 \\ 1, \ Otherwise. \end{cases} \tag{8}$$

3.4 Hierarchical Quorum Consensus (HQC)

The logical organization adopted in the HQC protocol [8] is a l level hierarchy with physical nodes at the lowest level. A node at the ith level of the hierarchy represents a logical subgroup consisting of n_i of $i-1$th level nodes. The system is available and a read or a write quorum is obtained when the root is available. The root is available when the majority of the nodes in the level below it are available. This definition of availability can be generalized to all the non-leaf nodes of the hierarchy. A leaf node is available when the corresponding physical node is available. A node initiating a read or a write operation needs permission from the root (the top logical level). This permission is obtained if the root is available.

For a read or a write quorum to be obtained, the root needs to collect votes from a majority of its children. We can visualize the root doing this by sending virtual messages to its children. In turn, each of the children of the root needs to collect votes from a majority of their children and so on. As stated earlier, the physical nodes in the system form the leaves of this logical tree. Given that a logical node at level i needs to collect votes from a majority of its children, the average number of virtual messages sent to $i-1$th level nodes from their parent at level i is given by $M_{Maj}(A_{HQC}(p,i-1), \frac{n_{i-1}}{2}+1, n_{i-1})$ where $A_{HQC}(p,i)$ is the availability of a ith level node. This expression holds for any logical node at any level of the hierarchy. Thus, for each virtual message sent to a node at the ith level, $M_{HQC}(A_{HQC}(p,i-1), \frac{n_{i-1}}{2}+1, n_{i-1})$ messages are sent to the next level. Now, the number of messages sent to the nodes at the lowest level are the actual messages sent and this, if enumerated, will give us the average message overhead for the HQC protocol.

Let us denote the average message overhead of the HQC protocol by $M_{HQC}(p,l)$. Then, based on the above discussion, we can write the following expression.

$$M_{HQC}(p,l) = \begin{cases} M_{Maj}(A_{HQC}(p,l-1), \frac{n_{l-1}}{2}+1, n_{l-1}) \\ \times M_{HQC}(p,l-1), \quad l > 1 \\ 1, \quad Otherwise. \end{cases} \tag{9}$$

Availability of the HQC protocol (denoted by $A_{HQC}(p,l)$) can be evaluated using the following recurrence.

$$A_{HQC}(p,l) = \begin{cases} A_{Maj}(A_{HQC}(p,l-1), \frac{n_{l-1}}{2}+1, n_{l-1}), & \\ \hfill l > 1 & \\ p, \quad Otherwise. \end{cases} \tag{10}$$

A node at level l, when $l > 1$, is available if the majority of its children are available. A node at level $l > 1$ has n_{l-1} children each of which are available with probability $A_{HQC}(p, l-1)$. Substituting this for p in $A_{Maj}(p, n_{l-1}/2+1, n_{l-1})$, we get the first part of the above recurrence. It is easy to observe that when l is 1 $A_{HQC}(p,l)$ is p.

3.5 RST protocol

In the RST protocol [10], nodes are grouped into $\frac{N}{G}$ groups of G nodes each. Each such group is called a *subgroup*. Then, $\frac{N}{G}$ quorum groups are constructed such that each quorum group is made up of $K = \sqrt{\frac{N}{G}}$ subgroups with each subgroup containing G nodes. Also, these quorum groups are constructed in such a way that any pair of quorum groups intersects in exactly one subgroup. This would ensure that read and write operations can be synchronized (no more than one node can obtain a quorum). The construction is such that each node appears the same number of times in the subgroups. That is, each node will appear in exactly $\sqrt{\frac{N}{G}}$ quorum groups. Now, each such quorum group will be associated with G nodes.

When a node initiates a read or a write operation, it asks for permission from all the subgroups in its associated group. If permission is granted by all the subgroups, then the initiating node proceeds with the operation. Permission from a subgroup is obtained if a *majority* of the nodes in that subgroup give their permission. Thus, this is a two level protocol with a requirement of majority at the bottom level inside a subgroup and a quorum at the top level.

Given that a subgroup is available if at least a majority of the nodes in that subgroup are available, the availability of a subgroup is given by $A_{Maj}(p, \lfloor\frac{G}{2}+1\rfloor, G)$. Further, a quorum is available if all the K subgroups are available. Thus, the availability of the RST protocol is given by

$$A_{RST}(p,N,G) = A_{Maj}(p, \lfloor\frac{G}{2}+1\rfloor, G)^K \tag{11}$$

Also, the following equation can be used to calculate the expected number of request messages sent by the initiating node before a quorum decision is made (either the quorum is obtained or the request is aborted).

$$M_{RST}(p,N,G) = M_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G) \times \frac{1 - A_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G)^K}{1 - A_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G).} \tag{12}$$

The above expression can be derived as follows. A quorum cannot be obtained if permission from any one of the K subgroups is not obtained. The expected number of request messages required in order to obtain votes from a majority of the nodes in one subgroup (of G nodes) or to conclude that this is not possible is $M_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G)$. If permission is obtained from the first subgroup, permission request is made to the second subgroup and if this succeeds, then to the third subgroup and so on. The expected number of messages for this operation is given by

$$M_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G) + A_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G) M_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G) +$$

$$+ \ldots + A_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G)^{K-1} M_{Maj}(p, \lfloor \frac{G}{2}+1 \rfloor, G).$$

This expression can be summed up to yield the right hand side of the above equation.

4 Results

In this section we will analyze numerical results that are derived using the above equations. These results illustrate how the average message overhead and the availability of each protocol changes with different system parameters. Also, the results enable us to compare the protocols against one another. They also provide information about the scalability of each protocol, i.e., information about how the average message overhead and availability values change with an increase in system size N.

The numerical results are plotted in Figures 1 through 9. The plots can be classified under three categories: those that plot the average message overhead against node availability; those that plot the system availability against node availability; and those that plot ratio of message overhead over system availability against node availability. While the first two categories of graphs compare the protocols from the stand point of average message overhead and availability respectively, the third category of graphs compare them in the light of both system availability and message overhead. Best case and worst case message overhead analysis for the respective protocols have been shown in [1][2][8] and can be compared with our average case message overhead results. The plots are provided for all the protocols that we have analyzed and for system sizes varying from $N = 32$ to $N = 1024$.

Figures 1, 2 and 3 plot the performance parameters for the protocols that we have considered for a system size of 32 and 64. For these systems, it is not possible to form complete structures for each of the protocols that we compare; for example, a grid of 32 nodes with equal number of rows and columns cannot be formed. So, for some of the protocols that we consider, small approximations to the actual system size are made. For example, for the 32 node system, the grid for the grid protocol is approximated to have 5 columns and 6 rows, the HQC protocol uses a 3 level tree with 3 children per node and the RST protocol uses a subgroup size of 4 with 9 such subgroups (that is, $K = 3$). Similarly, for the 64 node system, the grid structure is now complete and has 8 columns and 8 rows, the HQC protocol uses a 3 level tree with 4 children per node and the RST protocol uses a subgroup size of 4 with 16 such subgroups (that is, $K = 4$). It can be observed from the graph in Figure 1 that the Tree protocol has the lowest average message overhead when the availability of individual nodes exceeds 0.5 and 0.6 for the 32 and the 64 node systems respectively. As expected, the message overhead of Majority Voting protocol is the highest among all the protocols investigated. It should be noted that for some protocols, the message overhead does not monotonically decrease with increasing node availability. This is because of the fact that for some regions of increased node availability, in some protocols, more messages are sent before the initiating node determines that the protocol has to be be terminated unsuccessfully. Figure 2 shows that the availability of the Majority voting protocol is better than the rest of the protocols and for node availabilities greater than 0.95, all the protocols have comparable system availabilities.

It is also useful to compare the protocols based on a parameter that measures the relative behavior of both average message overhead and system availability. In Figure 3, the ratio of average message overhead to system availability is plotted for system sizes of 32 and 64. It is seen that for node availabilities greater than 0.75, it is seen that this ratio is the smallest for the tree protocol and the ratio for the HQC protocol is comparable to the tree protocol. Further, the ratios of the Grid and RST protocol are comparable to each other. The majority voting protocol has the worst relative ratio of all the protocols.

In order to study the scalability of these protocols, the performance parameters are further computed for larger system sizes of 128, 256, 512 and 1024. Figures 4, 5 and 6 provide these graphs for system sizes of 128 and 256. It can be observed from Figure 4 that the majority protocol has the worst average message overhead as expected, but now, for node availabilities greater than 0.9, the average message overhead of all the protocols become comparable. If we consider the whole range of node availabilities, the tree protocol still has the smallest average message overhead. As our analysis suggests, the gap between the average message overhead of the majority protocol and the other four protocols widens when the system size is increased. Figure 5 shows that with increased system sizes, the system availability of the grid protocol gradually becomes worse. It can be shown [10], that when the system size is asymptotically increased, all the protocols that we have considered except the grid protocol see an asymptotic increase in their system availabilities. For node availabilities greater than 0.9 all the protocols except the grid protocol, have comparable system availabilities. Figure 6 shows that although the tree protocol still has the best relative ratio between average message overhead and system availability over the whole range of node availabilities, with increasing system size, as seen in the graph for the 256 node system, the gap between the four protocols (other than the majority voting protocol) really quite small.

The effect of further increase in system size on the performance parameters can be studied using Figures 7, 8 and 9 which provide graphs for system sizes of 512 and 1024. The RST protocol is designed to give asymptotically better performance when the system size is increased and this effect can be seen in these graphs. Figure 7 shows that for a system size of 1024, the RST protocol is second only to the tree protocol, and the Grid and HQC protocols have comparable performance for high node availabilities. Also, from Figure 8, it can be seen that for a system size of 1024, the RST protocol has better availability than the Grid and HQC protocols for node availabilities greater than 0.8. The system availability of the grid protocol has further worsened. Figure 9 that provides the relative ratio between average number of messages and system availability further solidifies the scalability comparison. It shows that when the system size is increased, the relative order of performance exhibited varies from tree being the best, to RST, HQC, grid and majority in that order. Interestingly enough, it is seen that for node availabilities smaller than 0.9 or so, the grid seems to have a relative ratio value worse than that of the majority protocol. This, as mentioned earlier, is due to the fact that there is an asymptotic increase in the system availabilities of all the protocols except the grid protocol, when the system size is asymptotically increased.

The main goal of our paper is to point out that average message overhead of replica control protocols is a practical parameter that should be used to discriminate among the different protocols, and to provide expressions for computing this parameter. Given that the protocols that we have compared have very many possible configurations, it is difficult to exhaustively explore all such possibilities. The expressions that we have derived can be used to study design spaces that are of particular interest.

5 Conclusions

In this paper, we compared the *average* message overhead of some well known replica control protocols in the presence of node failures. Previous work has discussed the performance of these protocols from the stand point of availability of the system. But the issue of message overhead has largely been limited to the analysis of worst case and best case message bounds. We derived expressions for the *average* message overhead of these replica control protocols and used numerical examples to study their performance in terms of average message overhead and availability. Our analysis is expected to help system designers in selecting an appropriate protocol for replica control suitable to their requirements.

References

[1] D. Agarawal and A. El Abbadi, "An Efficient Fault-tolerant Solution to the Mutual Exclusion Problem," *ACM Transaction on Computer Systems*, vol. 9, no. 1, pp. 1-20, Feb. 1991.

[2] S. Y. Cheung, M. H. Ammar and M. Ahamad, "The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data," *Proceedings of the Sixth International Conference on Data Engineering*, pp. 438-445, 1990.

[3] M. Ahamad, M. H. Ammar and S. Y. Cheung, "Multi-Dimensional Voting," *ACM Trans. on Computer Systems*, pp. 399-431, Nov. 1991.

[4] P. A. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[5] A. El Abbadi, D. Skeen and F. Cristian, "An Efficient Fault Tolerant Protocol for Replicated Data Management," *Proceedings of the Symposium on Principles of Database Systems*, pp. 215-228, 1985.

[6] D. K. Gifford, "Weighted Voting for Replicated Data," *Proceedings of the Seventh Symposium on Operating System Principles*, pp. 150-159, 1979.

[7] S. Jajodia and D. Mutchler, "Dynamic Voting," *Proceedings of SIGMOD*, pp. 227-238, May 1987.

[8] A. Kumar, "Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data," *IEEE Transactions on Computers*, vol. 40, no. 9, pp. 996-1004, September 1991.

[9] J. F. Paris and D. Long, "Efficient Dynamic Voting," *Proceedings of the International Conference on Data Engineering*, January 1988.

[10] S. Rangarajan, S. Setia and S. K. Tripathi, "A Fault-Tolerant Algorithm for Replicated Data Management," *Proceedings of the Eighth International Conference on Data Engineering*, pp. 230-237, February 1992. Also, to appear in the *IEEE Transactions on Parallel and Distributed Systems.*

[11] R. H. Thomas, "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," *ACM Transactions on Database Systems*, vol. 4, no. 2, pp. 180-209, June 1979.

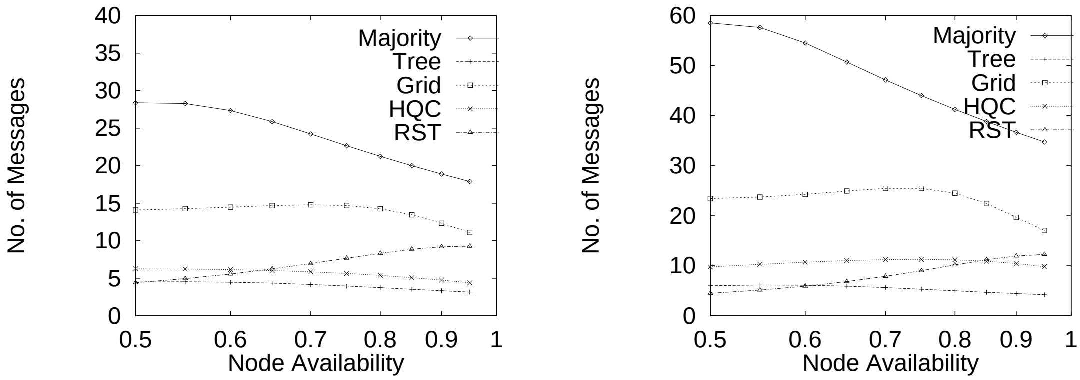

Figure 1: Average Message Overhead in a 32-node System (left) and in a 64-node system (right)

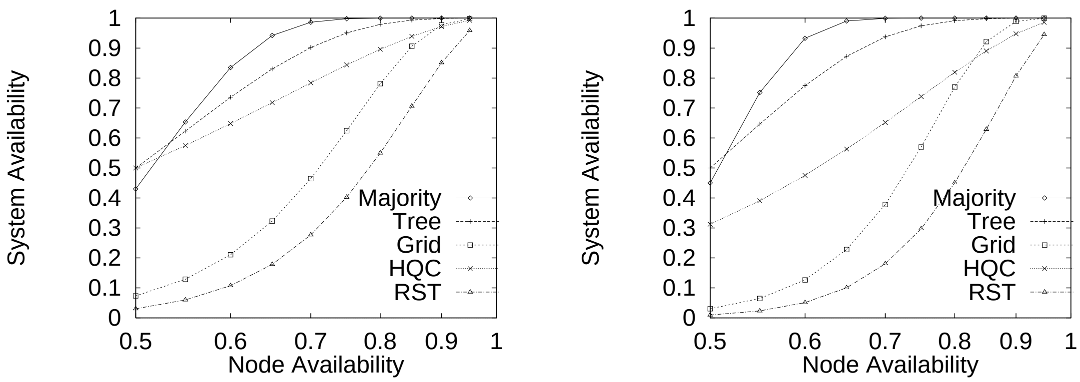

Figure 2: System Availability in a 32-node System (left) and in a 64-node system (right)

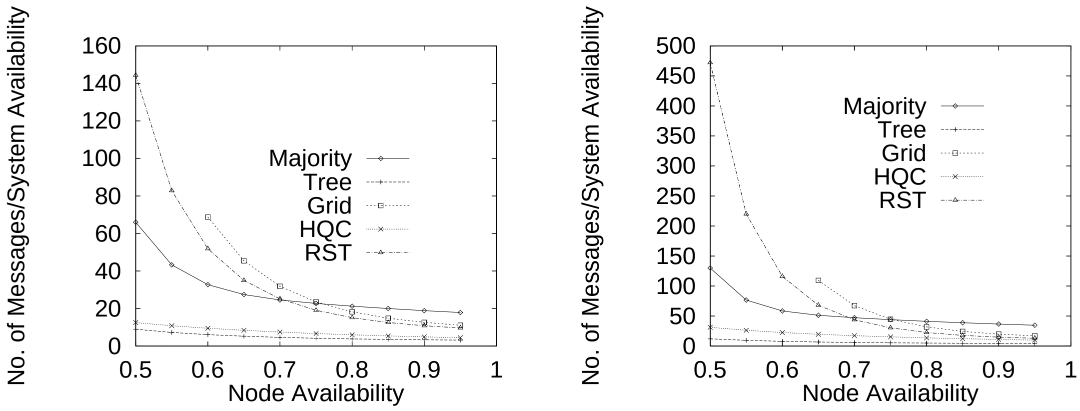

Figure 3: Ratio of Average Message Overhead/System Availability in a 32-node System (left) and in a 64-node system (right)

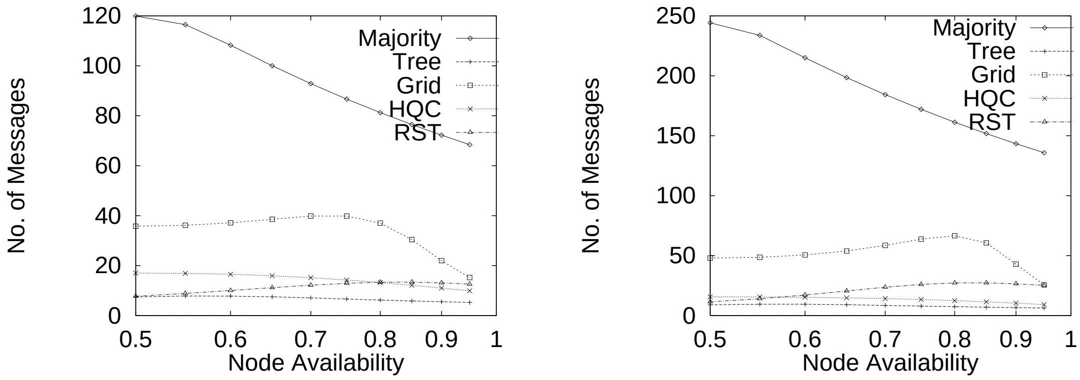

Figure 4: Average Message Overhead in a 128-node System (left) and in a 256-node system (right)

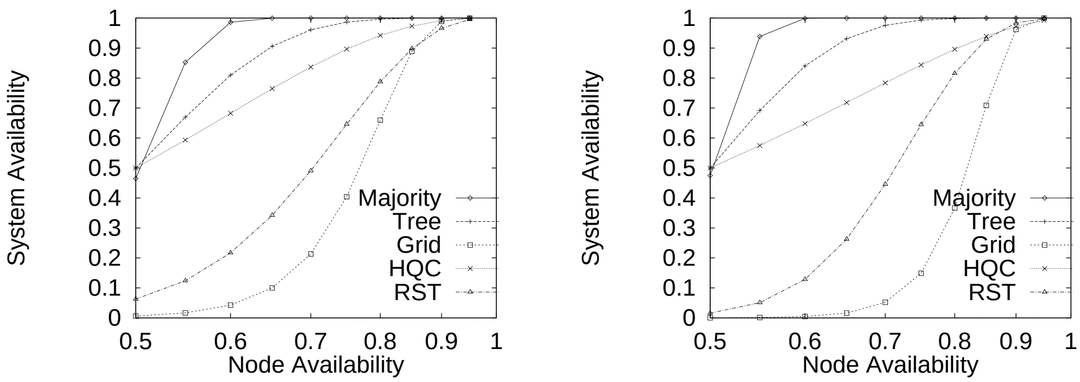

Figure 5: System Availability in a 128-node System (left) and in a 256-node system (right)

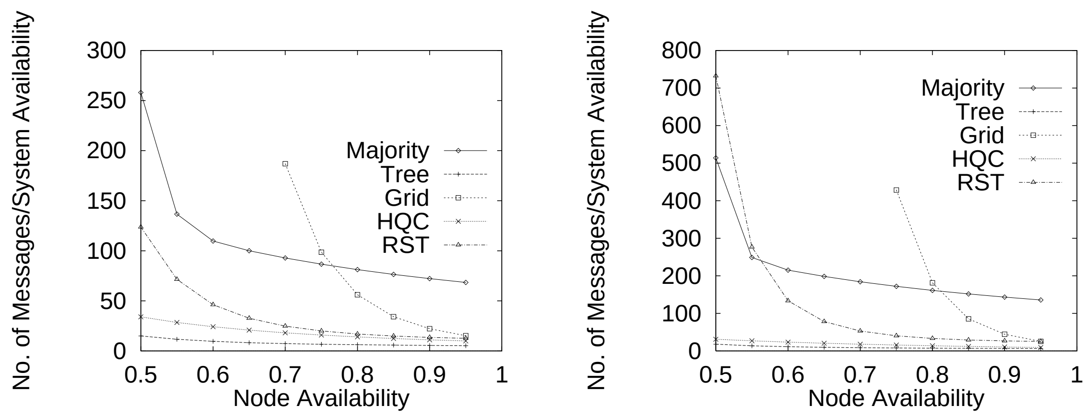

Figure 6: Ratio of Average Message Overhead/System Availability in a 128-node System (left) and in a 256-node system (right)

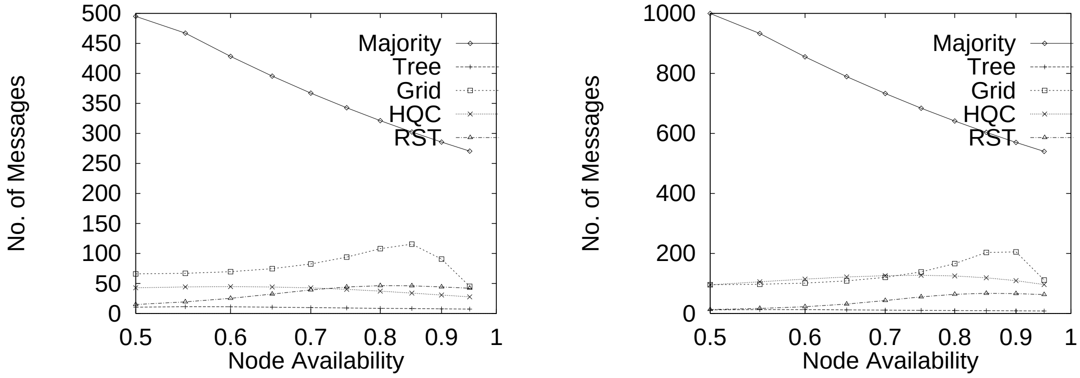

Figure 7: Average Message Overhead in a 512-node System (left) and in a 1024-node system (right)

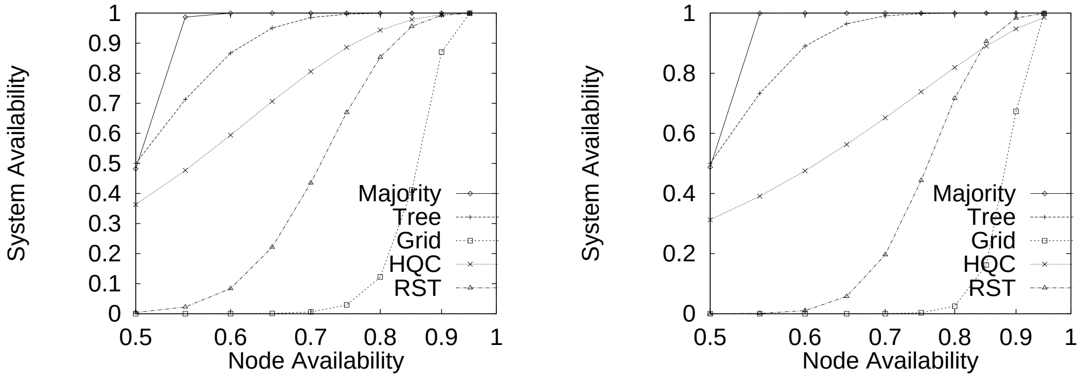

Figure 8: System Availability in a 512-node System (left) and in a 1024-node system (right)

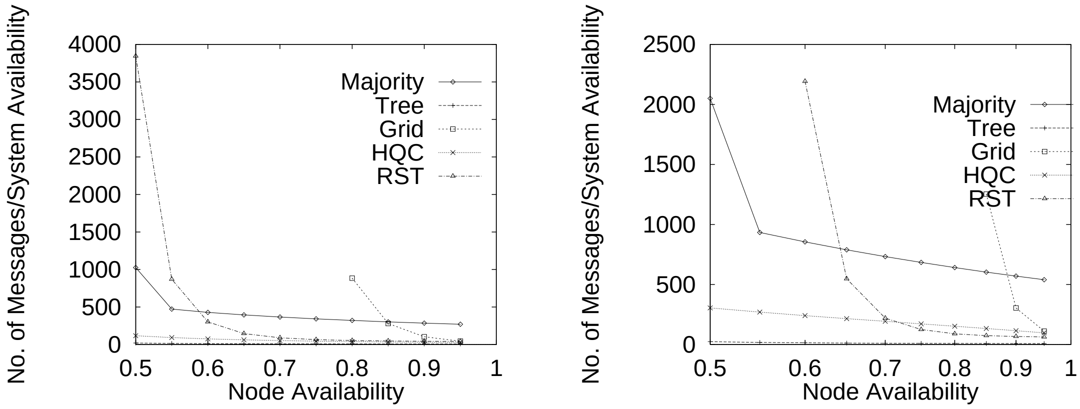

Figure 9: Ratio of Average Message Overhead/System Availability in a 512-node System (left) and in a 1024-node system (right)