A Robust Distributed Mutual Exclusion Algorithm
Sampath Rangarajan* Satish K. Tripathi†
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

Abstract
Correct operation of a distributed system with replicated resources requires that mutual exclusion be maintained among independent requests to these resources at different sites in the system. In this paper, we propose "asymptotically high resiliency" as a performance measure for fault-tolerant mutual exclusion algorithms for systems where sites can fail. We then present an efficient highly fault-tolerant algorithm for mutual exclusion. The algorithm is totally distributed in nature and is shown to have a message overhead of $O(\sqrt{N \log N})$ where $N$ is the number of sites. The algorithm provides mutual exclusion with a resiliency (that is, probability that permission for a mutual exclusion request is obtained in spite of site failures) approaching 1 asymptotically with an increase in $N$.

1 Introduction
Reliability of a distributed system can be increased by replicating a resource at different sites. For example, a file can be replicated at different sites to improve its availability in case of site failures. Correct operation of a distributed system with such replicated data requires that mutual exclusion be maintained among independent requests to copies of the same data in different sites. There are a number of other applications in distributed

*E-mail: sampath@umiacs.umd.edu
†Also with the Department of Computer Science, University of Maryland at College Park, College Park, MD 20742. E-mail: tripathi@cs.umd.edu
systems with the requirement that exclusive access to a resource be given to a single process in the system. These include atomic commitment and other synchronization problems.

All mutual exclusion algorithms are instances of what are called quorum consensus protocols. In a quorum consensus protocol, an operation (or a process performing that operation) can proceed only if it gets permission from a group of other processes. Such a group is called a quorum group. Collection of all quorum groups will constitute a quorum set. Permission from all processes from any one of the groups in the quorum set is sufficient for an operation to proceed, and to ensure mutual exclusion, every pair of groups in the quorum set should have a non-empty intersection [4]. In the rest of the paper, without loss of generality, we assume that each process runs on a different processor or site. Henceforth, we will use the terms process and site interchangeably. A distributed algorithm for mutual exclusion is one where each site bears equal responsibility in controlling mutual exclusion. Each site in a distributed algorithm serves as the arbitrator for the same number of sites. In a non-distributed algorithm, some sites will bear more responsibility than some others in controlling mutual exclusion.

There are two main issues in the design of fault-tolerant quorum consensus algorithms for mutual exclusion. The first issue is the message overhead involved per mutual exclusion request and the second issue is the resiliency of the algorithm. Given the possibility of site failures, one measure of resiliency is the probability that if a mutual exclusion request is initiated by a site, then a quorum group can be found by the algorithm (which of course, includes the initiating site) such that all sites in the quorum group are available (a site is available, if it has not failed) to control mutual exclusion. We will call this measure site resiliency and denote it by $R_{site}$. A weaker measure of resiliency is the probability that at least one quorum group (in the whole system) can be found such that all sites in that group are available. We will call this measure system resiliency and denote it by $R_{sys}$. System resiliency in some sense gives us the probability that mutual exclusion requests initiated through at least one site in the system will succeed (in spite of site failures). Note that resiliency is dependent not only on how the quorum groups are formed, but also how the quorum groups are used by an algorithm. Hence, the term resiliency of an algorithm (rather than resiliency of a quorum grouping procedure). In the rest of the paper, we will use the term resiliency when we want to refer collectively to both the above measures. It is to be noted that we are concerned only with the fault tolerance of mutual exclusion algorithms. So, as far as we are concerned, a mutual exclusion request may not be satisfied only due to site failures. We do not consider the situation where a mutual exclusion request is not granted because some other site has already been granted permission and is using the replicated resource.

The algorithm presented in [13] requires $O(N)$ messages per mutual exclusion request