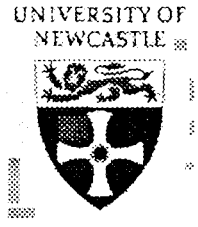
THE UNIVERSITY OF NEWCASTLE UPON TYNE

DEPARTMENT OF COMPUTING SCIENCE






# Performance Modelling of Replication Protocols

by

Manoj Misra

PhD Thesis

October, 1997

## Abstract

This thesis is concerned with the performance modelling of data replication protocols. Data replication is used to provide fault tolerance and to improve the performance of a distributed system. Replication not only needs extra storage but also has an extra cost associated with it when performing an update. It is not always clear which algorithm will give best performance in a given scenario, how many copies should be maintained or where these copies should be located to yield the best performance. The consistency requirements also change with application. One has to choose these parameters to maximize reliability and speed and minimize cost. A study showing the effect of change in different parameters on the performance of these protocols would be helpful in making these decisions. With the use of data replication techniques in wide-area systems where hundreds or even thousands of sites may be involved, it has become important to evaluate the performance of the schemes maintaining copies of data.

This thesis evaluates the performance of replication protocols that provide different levels of data consistency ranging from strong to weak consistency. The protocols that try to integrate strong and weak consistency are also examined. Queueing theory techniques are used to evaluate the performance of these protocols. The performance measures of interest are the response times of read and write jobs. These times are evaluated both when replicas are reliable and when they are subject to random breakdowns and repairs.

**Acknowledgements**

First of all I would like to thank my supervisor Prof. Isi Mitrani. Without his help and guidance this thesis would never have been completed. I would also like to thank Prof. Santosh Shrivastava for his support and guidance over the past three years. Several other people at Newcastle University have also been of considerable help to me, of these I would like to specially mention Dr. M. Little, Mr. Trevor Kirby, Dr. Ram Chakka and Dr. Paul Ezhilchelvan for their advice and help.

Personal thanks go to my family and friends and especially to Chris Angus. This work was supported by a Commonwealth Scholarship.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Statement of problem

The motivation behind data replication is to improve both the availability and the speed of retrieval of data objects. These objectives have become increasingly important in recent years as the provision of on-line information and the demand for it have grown exponentially. The idea is simple: keeping several copies of an object on different servers would facilitate (a) its survival in the event of hardware crashes and (b) its accessibility to several users in parallel. These advantages are of course bought not only at the price of extra storage needed for storing more than one copy of the data (replicas) but also at the price of the overhead incurred in maintaining consistency between the replicas. Different applications require different levels of consistency. In some cases it is important that all copies should be identical all the time whereas other applications may tolerate intermediate inconsistencies among different copies of the data provided that all copies eventually become the same. The cost of the protocol to maintain these copies depends on the type of consistency needed. It is therefore important to be able to evaluate the effect which a given consistency

protocol has on the performance of the system.

There is no shortage of proposed algorithms to manage replicated data. These protocols usually fall into two different categories : (a) *strong consistency protocols* guarantee that all replicas are identical at all times and ensure that a read always gets most recent value of the data and (b) *weak consistency protocols* allow replicas to differ in order to improve performance. Some protocols that try to integrate both of these approaches into the same framework have also been proposed [1]. These protocols try to organize replicas in levels with each level providing a different type of consistency. We call these protocols *multi-level protocols.*

Replication is not cheap. It not only needs extra storage but also extra cost to perform an update. It is still not clear which algorithm will give best performance in a given scenario, how many copies should be maintained or where these copies should be the located to yield the best performance. The consistency requirements also change with application. One has to choose these parameters to maximize reliability and speed and minimize cost. A study showing the effect of change in different parameters on the performance of these protocols would be helpful in making these decisions.

In this thesis we study and compare the performance of data replication protocols using analytical modelling methods. The models used for performance analysis make certain simplified assumptions so that they can be solved mathematically. The results obtained from such analysis depend on the assumptions made to reach a suitable model. These assumptions generally are about the probabilistic distributions of job arrivals, service times, failure and repair characteristics. Even in the presence of simplifying assumptions the results obtained from such analysis are quite helpful to understand the behaviour of actual physical phenomenon. Moreover these methods

have the advantage of computational efficiency over other techniques. At some places where the exact analysis of the model was intractable, we provide approximate solutions. We then compare the results obtained from these approximate solutions with simulation results.

Different replication protocols are being designed to work efficiently for different type of applications. It is not wise to compare the performance of protocols that fall into one category with the protocols that fall in some other category. Protocols that maintain strong consistency are suitable for the case when the number of replicas are not very large. Our analysis shows that for these protocols there is an optimal degree of replication which depends on arrival and service rates of jobs. Increasing number of copies beyond this limit degrades the performance of the protocol instead of improving it. On the other hand weak consistency protocols are being designed for the applications that can tolerate some inconsistency in data to get better performance. In this thesis we first study the performance of quorum based protocols that provide strong consistency. Then we study the performance of a hierarchical replication protocol with two levels of hierarchy. The protocol keeps some replicas strongly consistent while allowing others to differ providing different levels of consistency at different levels of hierarchy. We finally study and compare the performance of protocols that allow updates and queries to occur asynchronously on any replica.

## 1.2 Summary of Previous Work

**Weighted Voting algorithm**

The first voting approach was the majority consensus algorithm proposed by R. H. Thomas [67]. The algorithm assumes the existence of two processes, a database

managing process (DBMP) and an application process (AP). The database copy at each site is accessible only through DBMP residing at that site. Query and update accesses to the database are initiated by AP. An AP while submitting an update also supplies the base value of the data to which update should be applied. A DBMP on receiving an update compares the base value of the data item with current value. If there is no change and this request does not conflict with some pending request it votes OK otherwise votes PASS. If the value has been changed it votes REJECT. An update can only be executed after getting a majority of OK votes. This algorithm was then generalized by D. K. Gifford in [25]. He named his algorithm the Weighted Voting Algorithm. The algorithm proposed by D. K. Gifford allows a write (read) request to execute only after collecting a write (read) quorum of votes. Several variants of this algorithm also exist in the literature [60, 33, 5, 8] that try to improve the performance of the protocol for a particular situation at the expense of running some expensive solution for some other case. We explain Weighted Voting Algorithm and its variants in detail in chapter 2. There are several modelling and simulation studies of this algorithm and its variants, and of related issues concerning readers and writers in database systems. Various aspects of performance and availability have been addressed by means of finite-state models. Some of them are:

In [64] W. Smith and P. Decitre give two procedures, based on probabilistic analysis, to determine the availability of a replicated object and the probability that a read or write request fails.

In [6] M. Ahamad and M. H. Ammar present an analysis of the algorithm assuming that if an appropriate quorum is available for a transaction, its service time is negligible and thus service is completed instantaneously. This enables them to model the system as a birth and death process representing failure and repair of sites. They

use this model to find the optimal degree of replication, optimal quorum assignment and mean transaction response time which is the first passage time for an arriving transaction from the state when it arrives to the state when a quorum is available for its execution. This model does not consider the effect of queueing or the service times of read and write jobs on the performance.

In [61] D. Saha and *et al.* compare the performance of the Weighted Voting protocol and its variants based on average message overhead.

These studies ignore the effect of queueing and congestion. Studies that use the queueing theory approach for studying the performance of replication are [16, 11, 58, 35, 36].

Systems where the queue of read access is saturated, or where any number of reads can be processed in parallel, were examined in [16] by E. G. Coffman et al. In [11] Baccelli and Coffman have analysed a data replication model with stable queues for both read and write access (the latter having preemptive priority over the former), by treating the write requests as interruptions. The analysis assumes that a read needs only one copy for execution whereas a write needs all copies.

Nelson and Iyer [58] have applied the matrix-geometric solution method to a different model where read and write accesses wait in a common queue and are served in order of arrival. They present two models for the synchronous and the non-synchronous cases. A read request needs only one copy whereas a write is performed on all copies. In the synchronous case a write operation in progress blocks all the requests that arrived after it until its completion of service. An unblocked read request can start processing when it reaches the head of the request queue and when there are no outstanding write requests that arrived before it. In the asynchronous case a write operation releases each copy as soon as it is updated and released copies are available

for service to any waiting read request at the head of the request queue.

In [35] V. G. Kulkarni and L. C. Puryear consider a reader-writer queue with readers having non-preemptive priority over writers. The system can process an unlimited number of readers simultaneously. However, writers have to be processed one at a time. The analysis uses an $M/G/\infty$ queue busy period to model readers, followed by a modified $M/G/1$ queue to model the entire system.

In [36] V. G. Kulkarni and L. C. Puryear analyse the stability and queuing time of the reader-writer queue with alternating exhaustive priorities and no breakdowns. Again the system can process an unlimited number of readers simultaneously and writers have to be processed one at a time. There is infinite waiting room for both. The alternating exhaustive priority policy operates as follows. Assume that the system is initially idle. The first arriving customer initiates service for the class (readers or writers) to which it belongs. Once processing begins for a given class of customers, this class is served exhaustively. At this point, if the customer of the other class are in the queue, priority switches to this class, and it is served exhaustively. This is a variant of polling.

### Multi-level Replication Protocols

Users are sometimes willing to accept slightly out-of-date information, if they can access it much faster. This is the basis of the *universal name service* proposed by C. Ma in [46] and hierarchical asynchronous replication protocol, or HARP, proposed by N. Adly *et al.* HARP organizes replicas into a multi-level hierarchy. Replicas at level 0 are always strongly consistent but replicas at other levels may become out of date. In [2], N. Adly *et al.* have evaluated the performance of their protocol. That evaluation is based on a separable queueing network model, which precludes simultaneous occupation of several servers by one request, priority scheduling for

requests of different types and server breakdowns.

In *quasi-copies* schemes proposed by D. Barbara and H. Gracia-Molina in [9] there is a central location where all the updates are processed, and several copies are located throughout the network. A predicate is associated with each copy, establishing the degree of inconsistency that can be tolerated. The scheme was mainly proposed to use the user's local storage capabilities to cache data at the user's site in an information retrieval system. In [9] authors present a performance model for their scheme. As there is only one central location, they model the central and other nodes as M/G/1 servers. This is different from the two-level replication analysed in this thesis where there are more than one copies which are strongly consistent and we have to consider the simultaneous occupation of several servers by one request.

Another scheme that deals with two-tier replication has been presented in [31] by Gray *et al.* for mobile systems. They also compare the performance of their scheme with eager replication (strong consistency protocols) and lazy replication schemes. As in the case of [2] their analysis also precludes simultaneous occupation of several servers by one request, priority scheduling for requests of different types and server breakdowns.

Our analysis of these protocols assumes the existence of separate queues for read and write jobs and shows the effect of simultaneous occupation of several servers by one request on the response time of read and write jobs. For reliable replication the analysis is based on a similar analysis done by I. Mitrani and P. J. B. King [51] for multiprocessor systems with preemptive priority.

For the case of breakdowns and repairs we use the Spectral expansion method described in [50]. This method can be used to solve a class of two-dimensional Markov models whose state space is a lattice strip. As the Markov model with two queues for

read and write and with breakdowns become three dimensional we use approximations to solve the system.

### Gossip and Timestamped anti-entropy protocols

In [39] R. Ladin proposes a protocol that allows an operation (an update or a query) to happen at a single replica. The effects of the call are then propagated to other replicas by lazy exchange of gossip messages between replicas. The method allows three kinds of operations: client ordered, server ordered and globally ordered operations. We describe them in detail in chapter 2. Results showing the performance of the protocol based on experiments performed have also been presented in [39].

In [26, 27] Golding presented his Timestamped anti-entropy protocol (TSAE) which provides weak consistency. Like [39] TSAE also allows an operation (an update or a query) to happen at a single replica. In the TSAE protocol each replica at random intervals selects some other replica and instead of sending a gossip message it exchanges information with the selected replica. Once this exchange is complete both replicas have seen the same set of messages. In [28, 29] Golding presents the results from his simulation analysis of the protocol giving probability of successfully delivering a message to all sites, expected data age, probability of getting old values etc. The paper also shows the effects of partner selection policy and the number of sites on the performance. The analytical model for spreading an update to all replicas has also been given in [28]. He used Monte Carlo simulation to get the results for the analytical model.

In this thesis we present an analytical solution to evaluate and compare the performance of the schemes proposed in [39] and [26] for spreading the updates. In our model, replicas execute updates in the order in which they arrive in the system. We take average response time of an update as the performance parameter which we

define as the difference between times when an update arrives at a replica and the time when it can be executed on that replica. To the best of our knowledge this problem has not been analysed before. We show that this time is the same as the time taken to spread an update to all replicas in the dual system (we define dual system in chapter 7). We show that this time depends on the connectivity of the network. We also derive upper and lower bounds on the time when an arriving update can be executed by all replicas.

## 1.3 Overview of Thesis

The aim of this thesis is to study and compare the performance of various data replication protocols. First we describe different aspects that should be considered while evaluating the performance of these protocols and give a classification of these protocols in chapter 2.

In chapter 3 the analysis of weighted voting protocol is presented both in case of reliable replicas and when breakdowns may occur. We use generating function approach to solve our model when all replicas are reliable. We then present the analysis of our model for the case when replicas may fail but join the service again within a finite time after being repaired. As the exact analysis of this model with both read and write queues unbounded is, at present, intractable, we provide an approximate solution using the spectral expansion method. We also compare the approximate analytical results with the simulation results. The comparison shows that results obtained from such an approximation are very close to exact results when write arrival rate is low.

Chapter 4 shows the effect of scheduling strategies on the performance of the Weighted Voting protocol. We compare the results for the case when write jobs have

higher priority with the case when FIFO scheduling is being used.

In chapter 5 we give an exact solution for the system with breakdowns when there is a single server, two type of jobs and type 1 jobs have preemptive priority over type 2 jobs.

The analysis of a data replication protocol with two levels of consistency is given in chapter 6. In many applications the user does not require the most recent information and he may be satisfied with slightly out-of-date information if it can be accessed quickly. This is the basis of this type of protocols. This chapter not only presents the analysis of the protocol both for reliable and unreliable replication but also gives an analytical method to find out the probability of getting out-of-date information. The results presented in this chapter have been published in the form of a paper in the *2nd annual IEEE International Computer Performance and Dependability Symposium at Urbana-Champaign, Illinois, 1996* [53].

Chapter 7 first presents the analysis of gossip scheme for spreading updates when updates are being executed in the order they arrive into the system. It then compares its performance with exchange-gossip scheme where instead of sending gossip messages at random intervals replicas exchange information with each other. Upper and lower bounds on the time when all replicas have executed a given update have also been given. We also show the effect of network topology on the performance of the protocol. Most of the results given in this chapter have been presented in *3rd CaberNet Plenary Workshop held in Rennes in April 1997* [54].

Chapter 8 concludes the thesis and gives directions for further research work in this and related areas.

# Chapter 2

# Data Replication

## 2.1 Introduction

The replication of data objects on several sites has been advocated as an approach for improving both the availability and performance of distributed systems. By storing copies of shared data on processors where they are frequently accessed, the need for expensive remote read access is decreased. By storing copies of critical data on processors with independent failure modes, we can increase the probability that at least one copy of the data will be accessible even if some of the processors fail. However, these benefits are achieved at the cost of maintaining correctness of data across several copies [21]. Earlier approaches for maintaining replicated data attempt to keep all copies identical all the time. The correctness requirements depend on the application. For example, the USENET system maintains replicas of items posted to electronic bulletin boards across the Internet, the replicas being held within or close to the various organizations that provide access to it. The DNS naming service, maintains copies of name-to-address mappings for computers and other resources and is relied on for day-to-day access to services across the Internet [19]. Both of these

applications can cope with intermediate inconsistencies of data and by using this fact the performance of the data replication scheme can be improved.

The performance of a data replication scheme also depends on the granularity of data replication which may vary from in processor caches to replicating a whole database or file system. In this chapter we first explain various terms related with data replication and describe parameters used to evaluate the performance of different data replication schemes. We then classify and describe replication protocols. Finally we discuss the future of data replication.

## 2.2 Consistency and message ordering

An important criterion in the design of a data replication protocol is the type of consistency needed. This choice significantly affects the efficiency of data replication scheme. In [32] J. N. Gray *et al.* describe four degrees of consistency for database systems. P. A. Bernstein *et al.*, in [13], describe the consistency preservation as the concept of producing database states that are meaningful. He further states that *one copy serializability* can be assumed as the correctness criterion for replicated data. This requires that interleaved execution of the transactions on a replicated database should be equivalent to a serial execution of those transactions on a *one-copy* database. All these definitions of consistency are based on how execution of transactions affects the state of database.

There is some work that tries to define consistency of replicated data depending on how far different copies of data may differ. A. Sheth and M. Rusinkiewicz in [62] define consistency based on the difference of replicated copies in time and space. As pointed out the consistency requirements for replicated data may vary depending on the applications. Not all applications require that all copies should be identical all

the time. Many applications may tolerate intermediate inconsistencies which may arise based on the way updates are being implemented on different replicas. An update may or may not require synchronization among a group of replicas. It may complete after updating all replicas or it may complete after updating only a small set of replicas, possibly one, and then it may be propagated to other replicas in the background allowing replicas to differ at any time. Our definition of consistency is based on the difference in the states of the replicas at any time (how far the replicas may differ from each other).

### Strong or Immediate consistency

Strong consistency or immediate consistency guarantee that all replicas are identical at all time. Some of the possible ways of doing this are by means of quorums or with the help of some centralized control. If using quorums an update operation completes after updating all replicas in quorum and during this time no other operation (other update or read) can be in progress. This requires synchronization among a large number of replicas but ensures that replicas are mutually consistent and a read access always gets the most up-to-date version of the data. Protocols that provide strong consistency with the help of a centralized control depend on the reliability and speed of the centralized control.

### Weak consistency

Weak consistency does not guarantee that replicas are identical at all time. Weak consistency protocols provide higher availability and better response time by allowing updates and queries to occur asynchronously at any replica. The updates are then propagated to other replicas using some reliable or unreliable technique. This may create temporary inconsistencies among the replicas. Reads may read older versions

of data. This approach is based on the assumption that the applications can tolerate temporary inconsistencies. The system guarantees to resolve these inconsistencies and return the replicas to mutual consistency [23]. Reconciliation methods are available to resolve conflicts.

### Multi-level consistency

Some protocols that try to integrate both of these approaches into the same framework are also being proposed [1]. Protocols that fall into this category try to organize replicas in levels or groups with each level providing a different type of consistency. For example the protocol proposed in [1] by Noha Adley organizes replicas into a logical hierarchical structure and supports three different type of write operations and two different type of read operations (see section 2.5.3). By carefully choosing the type of read and write operations at each level an application designer may provide different type of consistencies for each level. We call these protocols *multi-level protocols.*

The order in which different updates are implemented is another important issue that affects the design of the protocol. Replicas may implement operations in a totally ordered, causally ordered or unordered way. The cost of implementing an update depends on the ordering imposed. Following are some of the most common orderings:

### Total

Updates are implemented in the same order at every replica. This order may be different from the order in which the update operations arrive in the system. Total order ensures that if update $a$ is being implemented before update $b$ at any replica then all other replicas will implement $a$ before $b$.

**Causal**

Update operations are implemented in an order that respects their causal relationship. Event $b$ causally depends on event $a$ if it occurred after $a$ on the same process or if $a$ is sending of a message by one process and $b$ is receiving of the same message by another process. If update $b$ causally depends on update $a$ then every replica implements $a$ before $b$. On the other hand if two updates are not causally dependent then they can be implemented in any order.

**Sync-ordering**

If a system supports several different type of ordering for its update operations then a sync-ordered operation ensures that all other operations are consistently implemented before it or after it [19]. For example, if any site implements an operation $a$ before implementing a sync-ordered operation $b$ then all sites will implement $a$ before $b$. This is true regardless of the type of order of $a$.

**Unordered**

There is no restriction on the order in which different update operations are implemented. They can be implemented in any order at any replica.

## 2.3 The Environment

Data replication protocols are designed to work in an environment in which individual computers, or nodes, are connected by a communication network. These protocols make certain assumptions about the type of failures that may occur in the network and certain services that should be available for the protocols to behave correctly. We first present a classification of the failure types and then describe some of the services these protocols may need to work correctly.

**Failure Types**

Failures may occur in the individual components of the network and due to these faults the network may partition. We first describe the faults that may occur in the individual components of the network. P. Jalote in [34] gives a classification of these faults based on how the faulty component behaves when it fails. He classifies the faults into four categories:

- *crash fault* The fault that causes the component to halt. With this type of fault, a component never undergoes any incorrect state transition when it fails. The processors that behave in this way in the event of any failure are termed as fail-stop processors.

- *Omission fault* This type of fault causes a component not to respond to some inputs.

- *Timing fault* When a component responds too early or too late, the fault is called as timing fault.

- *Byzantine fault* An arbitrary fault which causes the component to behave in a totally arbitrary manner during failure.

These faults form a hierarchy with the crash faults being the simplest to deal with and Byzantine faults being the most difficult. This hierarchy is shown in Figure 2.1

When the failure in nodes and communication links of the network fragments the network into isolated subnetworks in a way that nodes in one subnetwork can not communicate with the nodes in other subnetwork, it is called a *partition failure.* These subnetworks are then called partitions of the network.

Almost all replication protocols ensure the correct behaviour of the system when the failure in the components of the network is a crash failure. Many of these either

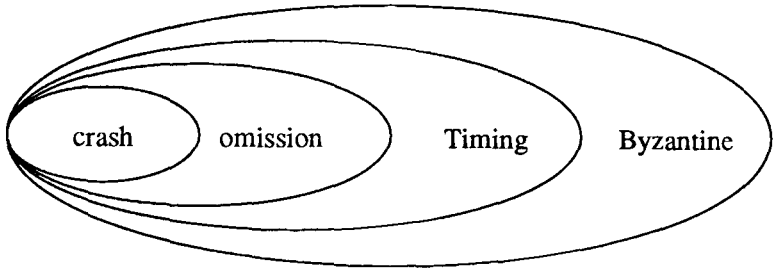



Figure 2.1: Failure Types

do not work or their performance degrades significantly when the network partitions.

### Stable Storage

Many replication protocols assume the existence of some stable storage for the correct functioning in the event of failure of network components. The contents of a stable storage are not destroyed or corrupted by a failure. P. Jalote in [34] defines an ideal stable storage as one where a read always returns good data (which is also the most recently written data), and a write always succeeds. He also describes methods by which approximations to stable storage can be implemented using disk storage system.

### Network Topology

Replication protocols generally do not make any assumption about the underlying network topology and guarantee to work correctly for all network topologies. But their performance may vary with network topology. We show this in chapter 7.

### Clocks

Each computer uses its own physical clock. These clocks are electronic devices that

count oscillations occurring in a crystal at a definite frequency, and which typically divide this count and store the result in a counter register. This can be read by software and scaled into a suitable time unit. This value can be used to timestamp some event or message on that computer [19]. As the crystal based clocks used in computers count time at different rates, they may diverge. Clock synchronization protocols like Berkeley algorithm and Network Time Protocol (NTP) try to ensure that clocks at different computers do not differ by more than a specified amount. Some data replication protocols that use such clock values in their timestamps may require the existence of a clock synchronization protocol to provide a certain level of consistency.

## 2.4 The Parameters

The performance of a data replication protocol usually depends on many parameters. In [55] H. Gracia-Molina categorizes these parameters into four different groups: base parameters, control parameters, failure parameters, and performance parameters. We discuss some of these parameters that closely affect the performance of a replication protocol.

- *Type of consistency* This is the most important factor that decides the design of the replication protocol. In section 2.2 we defined consistency based on the way replicas implement updates. Section 2.5 categorizes protocols depending on the type of consistency they support. The consistency requirements depends on the application which in turn decides the choice for a protocol.

- *Number of copies* The choice about the number of copies depends on the type of consistency and performance requirements of the application. Increasing the

number of copies should increase the availability and performance. But if the application needs stronger consistency which needs synchronization among the copies, increasing the number of copies more than a specified value may well decrease the performance. This is due to the overheads needed for synchronization among large number of copies. Even in case of weaker consistency requirements by the application the growth in performance may not increase linearly with number of copies.

- *Location of copies* In [44] M. C. Little and D. L. McCue show that the placement of replicas plays an important role in deciding the performance of the replication protocol. The placement of replicas should be chosen based on the reliability of nodes and links as well as the bandwidth of the network and the geographical distribution of requests. Little et al. in [43] describe a Replica Management System (RMS) that dynamically computes the level and placement of replicas to take into account the changing conditions in a distributed system. They show that the performance of such a dynamic system is far better than static one.

- *granularity of data* Size or granularity of data not only affects storage cost but also maintenance cost. For example if the replication protocol treats each entry in a table as a separate entity for replication the cost of replication may be too high. On the other hand if the complete table is being treated as a single object for replication, multiple requests trying to access different items of the table, not related with each other, will not be successful.

- *Failure model* The type of failures may vary from Byzantine to fail-stop. Most of the replication protocols assume that processors are fail-stop and there is no

partition in the network.

## 2.5 Replication Protocols

### 2.5.1 Strong Consistency Protocols

Strong consistency protocols always provide most current version of data to a user. These protocols may use primary copy (a centralized control) or quorum based approach to provide strong consistency. An approach to consistent replication based on quorums that has gained acceptance in the literature is provided by the weighted voting algorithm [25]. We describe the primary copy, weighted voting and some of the variants of weighted voting algorithm.

#### Primary copy

The primary copy approach has been used at many places, not just for data replication. M. Stonebraker describes the primary copy approach as used in INGRES in [65]. The basic approach can be described as having a primary site and some secondary (backup) sites. The number of secondary sites depends on the level of fault tolerance needed. If the operation is a read then it can go to any site that performs the operation and returns the result. A write operation first goes to the primary site. Before performing the write operation the primary site sends the write request to all backups. When all these backups have received the request, then the primary performs the operation and returns the result. This ensures that a read always gets the most recent version of the data. If a primary fails then a new primary has to be elected. There are various ways of electing the new primary as described in [56] by H. Garcia-Molina.

### Weighted Voting algorithm

In this algorithm [25] every copy of replicated data is assigned some number of votes. The algorithm uses two integers, $R$ and $W$, referred to as read quorum size and write quorum size, respectively. The execution of a read access requires the simultaneous holding of copies having a sum of $R$ votes, while that of a write access requires copies with sum of votes equal to $W$ votes. These numbers satisfy $R+W = N+1$, where $N$ is the total number of votes. Hence a read access and a write access cannot execute in parallel. The protocol also prohibits the parallel execution of two write access even if write quorums do not intersect. It does so by forcing a write access to first collect a read quorum and then collect the write quorum.

Every copy maintains a version number that reflects the number of updates that have been performed on this copy. Copies with highest version number are current copies. A write operation always updates current copies so there is always a subset of copies whose votes total to $W$ that are current. Because read and write quorums intersect, a read quorum always has a current copy.

The number of votes assigned to a copy depends on its importance. The performance and reliability characteristics of the protocol depends on the choice of $R$, $W$ and the voting structure. If all the copies have only one vote a read can tolerate up to $(N - R)$ faulty copies and a write can tolerate up to $(N - W)$ faulty copies.

### Voting with Witnesses

In [60] J. F. Paris proposes to replace some of the replicas by mere records of the current state of the file containing the data. Although not containing any data themselves, these records called *witnesses* can testify about the current state of the replicated file/data and can vote like conventional copies. Paris claims that because of their very small sizes, witnesses have practically negligible storage cost. Bringing a

witness up to date becomes also a trivial operation since it only involves the update of the version number. Witnesses can thus be created much more freely than conventional copies. He also shows that under very general assumptions, the reliability of a replicated file consisting of $n$ copies and $m$ witnesses is the same as the reliability of a replicated file consisting of $n + m$ copies.

### Dynamic Voting

In a voting-based scheme if there does not exist a partition containing a majority of sites, no updates can occur anywhere in the system. S. Jajodia and D. Mutchler in [33] propose an extension of voting algorithm which permits a file to be updated in a partition provided it contains a majority of up-to-date copies. Each copy along with the *version number* also contains an integer called the *update sites cardinality* which always reflects the number of sites participating in the most recent update. Whenever an update is made, it must be made to all sites in the partition. Thus if in the last update only $m$ out of total $N$ copies participated, the current update requires only a majority of $m/2 + 1$ copies (in contrast to Voting which needs a majority of $N/2 + 1$ copies).

### Hierarchical Quorum Consensus

A major problem with the quorum consensus method is that it does not scale well. The Hierarchical Quorum Consensus algorithm proposed by Akhil Kumar in [8] generalizes the quorum consensus scheme into a multilevel algorithm that requires a smaller quorum size of $N^{0.63}$ copies only. The algorithm logically organizes the set of copies of an object into a multilevel tree (of depth m) with the root at level 0. The physical copies of an object are stored only in the leaves of this tree, while the higher level nodes of the tree correspond to logical groups. The algorithm works as follows:

A read (write) quorum at level $i$ is defined as the number of subgroups of a level $i-1$ group that must be locked by a read (write) operation to obtain read (write) access to the group. This is a recursive definition.

For example Figure 2.2 shows how nine copies can be organized into three subgroups. If they are numbered as $c_{11}, c_{12}, c_{13}$ (subgroup 1), $c_{21}, c_{22}, c_{23}$ (subgroup 2) and $c_{31}, c_{32}, c_{33}$ (subgroup 3) a possible quorum is $c_{11}, c_{12}, c_{21}, c_{22}$ when write quorum is 2 for level 0 and 1.

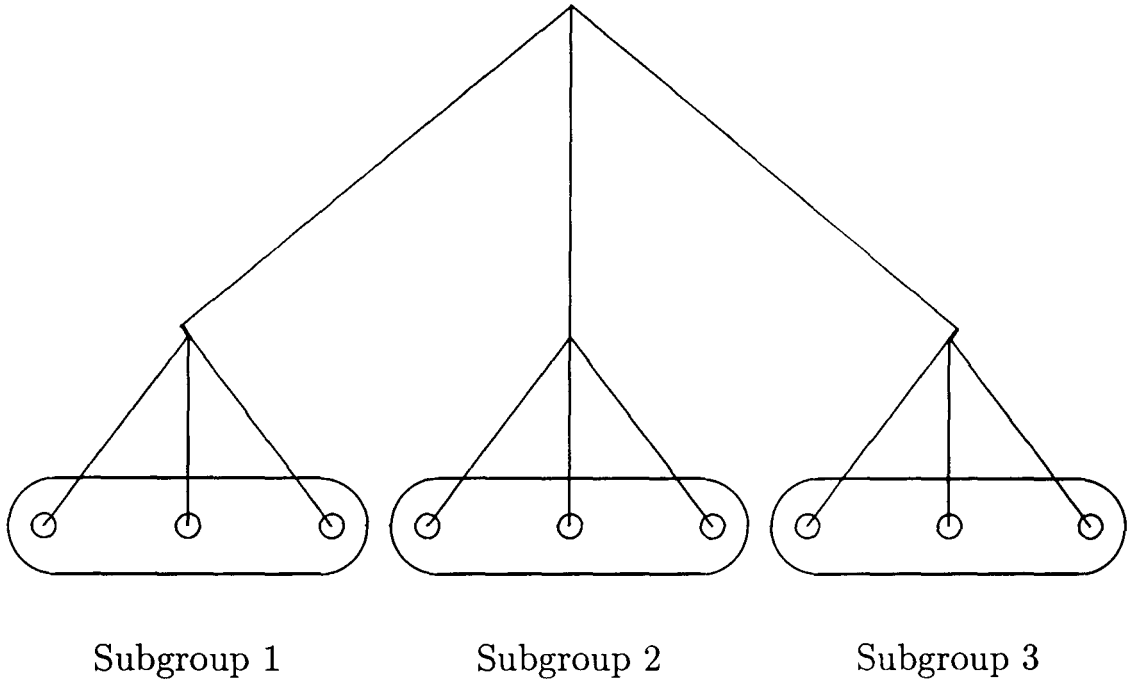



Figure 2.2: Hierarchical Quorum Consensus: Nine copies organized in three subgroups

### Tree Quorum Protocol

In [4] D. Agrawal and A. El Abbadi discuss how the synchronization cost of the quorum based algorithms can be reduced by exploiting the structural information of the underlying system. They describe their tree quorum protocol that organizes

replicas into a logical tree structure. A write quorum is constructed by selecting the root and a majority of its children. For each selected child, the protocol adds a majority of its children to the quorum. This process continues until the leaves are reached. A read quorum is constructed by selecting the root of the tree. If successful, this node constitutes the read quorum. If it fails, it tries to access a majority of the root's children. Again if successful this set constitutes the read quorum, otherwise, for each copy, which is inaccessible, the protocol tries to replace it with a majority of its children. This process is repeated recursively until a set of copies is included in the read quorum, or no such copies are accessible. For example in Figure 2.3 a write quorum may be {1,2,3,5,6,8,9} or {1,2,4,6,7,11,12} etc. A read quorum may be {1} or {3,4} etc. They also give upper and lower bounds on quorum sizes which depend on the height of the tree and logical connectivity of nodes of the tree.

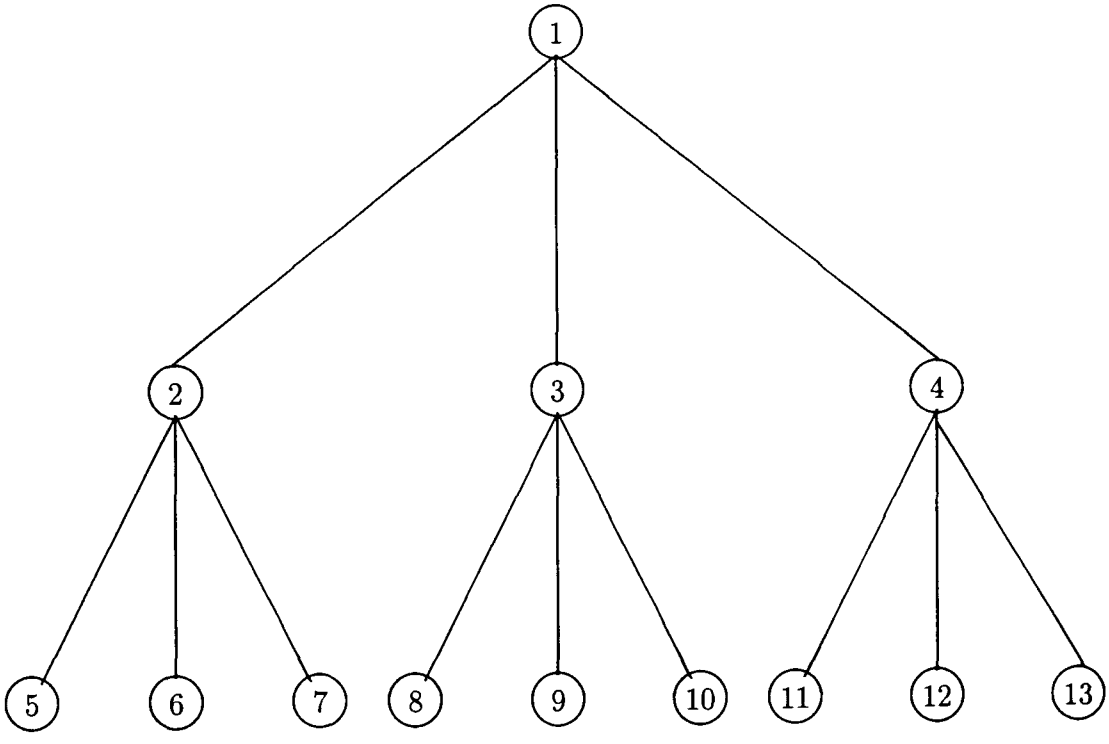



Figure 2.3: A tree organization of 13 copies

### Grid Protocol

In the grid protocol [15], presented by S. Y. Cheung et al. nodes that store replicas are arranged in a logical grid and read and write transactions are required to lock replicas in rows and columns of the grid so that conflicting transactions need to obtain locks from at least one common node. For example, a read transaction may lock all nodes of one column and a write locks all nodes of one row and one node in each of the other rows. In this scheme, only $O(\sqrt{N})$ of the $N$ nodes need to participate in a transaction.

### Delay-Optimal Quorum Consensus

This scheme suggested by Ada Waichee Fu in [70] takes into account the network topology and finds a quorum with minimum communication delay. Given an operation at a node $s$ it chooses a quorum such that its virtual distance $y$ from the furthest node in this quorum is minimized.

## 2.5.2 Weak Consistency Protocols

We call protocols that allow updates and queries to occur asynchronously on any replica, weak consistency protocols. Reads are allowed to see older versions of data. These protocols allow replicas to differ and generally provide a set of algorithms that support different level of consistencies based on the ordering imposed on implementation of updates.

### Epidemic Replication

Alan Demers *et al.* in [22] describe several randomized algorithms for distributing updates and driving the replicas toward consistency. These are:

- *Direct Mail*: each update is immediately mailed from its entry site to all other

sites. This is not entirely reliable as a site may not know about all other sites and mail is sometimes lost.

- *Anti-entropy*: every site regularly chooses another site at random and by exchanging database contents with it resolves any differences between the two.

- *Rumor mongering*: sites are initially "ignorant"; when a site receives a new update it becomes a "hot rumor"; while a site holds a hot rumor, it periodically chooses another site at random and ensures that the other site has seen the update. After trying to share an update with too many sites that have already seen this update the site stops spreading it.

### Grapevine and Clearinghouse

Grapevine [14] and Clearinghouse [59] are early examples of using a replication scheme that supports weaker consistencies. An update can be submitted at one replica and is later propagated to other replicas. During this period different copies of the replicas may differ from each other. Each update has a unique timestamp associated with it which is produced from the server's internet address and clock.

### Global Name Service

B. W. Lampson describes a Global Name Service in [42] that uses replication to provide high availability. The copies are kept approximately, but not exactly, the same. The update originates at one copy and is initially recorded there. The basic method for spreading updates to all copies is a *sweep* operation, which visits every copy, collecting a complete set of updates and writes this set back to every copy. All the copies are linked into a logical ring. The sweep starts at any copy and then goes through the complete ring returning back to starting point.

### OSCAR

OSCAR (Open System for Consistency and Replication) [23] provides a variety of message orderings. It is based on two cooperating agents called replicators and mediators which work together to provide replication and consistency for a set of database replicas. Each replicator is uniquely paired with a mediator and at least one mediator must be active in each network partition.

When a replicator receives an update from its database server it uses an unreliable multicast to send the update to all other replicators responsible for copies of the database. On receiving an update a replicator stores the update in its log and then delivers the update to its associated database server according to the consistency method associated with the data item.

A mediator periodically polls the replicators to get the information about the updates that have been received by each replicator. Once a round is complete a mediator summarizes the information and sends the summaries back to the replicators. The replicators may use this information to push and pull the missing updates.

### Lazy Replication

Rivka Ladin in [39] proposes a set of algorithms to implement three different type of orderings for the operations. These are:

- *Client ordered* The operations for which the clients define the required order dynamically during the execution.
- *Server ordered* these operations are totally ordered with respect to one another even when no dependency relationship is defined by the client.
- *Globally ordered* These operations are totally ordered with respect to all other operations.

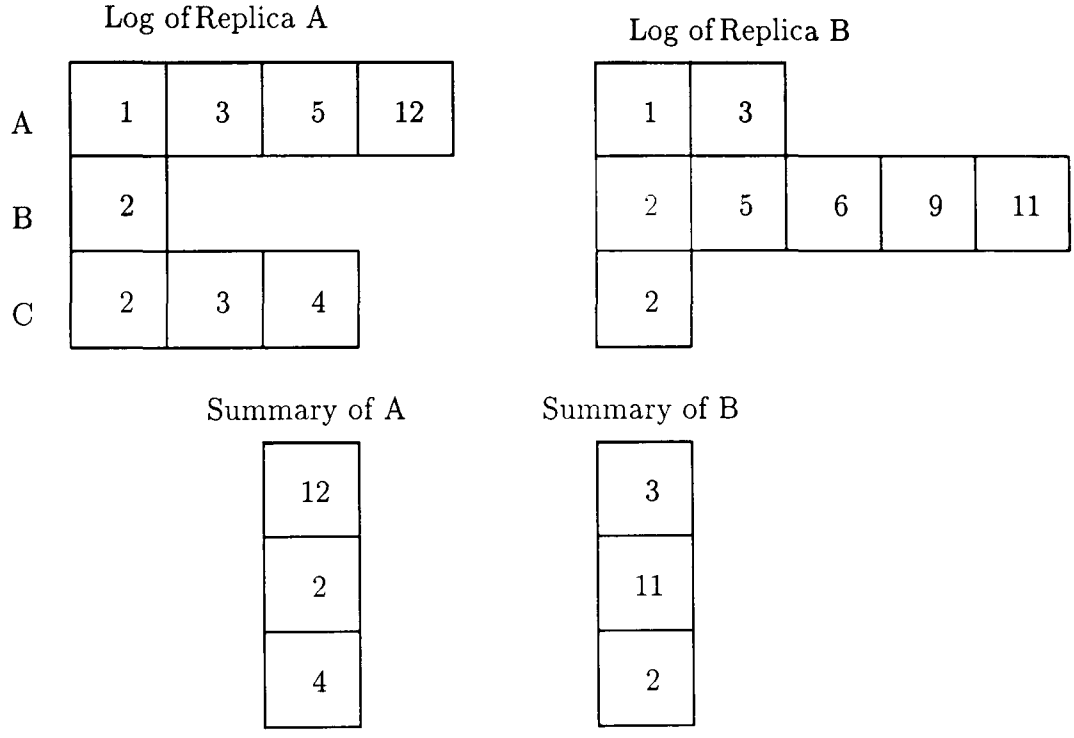
Each update is assigned a unique timestamp called its uid. In the first case when a client submits an update U it also tells about its dependency on other updates by passing a label along with the update. This label contains uids of all updates U depends on. An update is ready for implementing when the server has already implemented all updates it depends on. Server ordered updates also take an input label. The label identifies the client ordered updates and server ordered updates that must precede the server ordered update. Unlike other operations, a globally-ordered update U does not take a label as an argument; instead, the system decides what operations precede U.

**Timestamped anti-entropy**

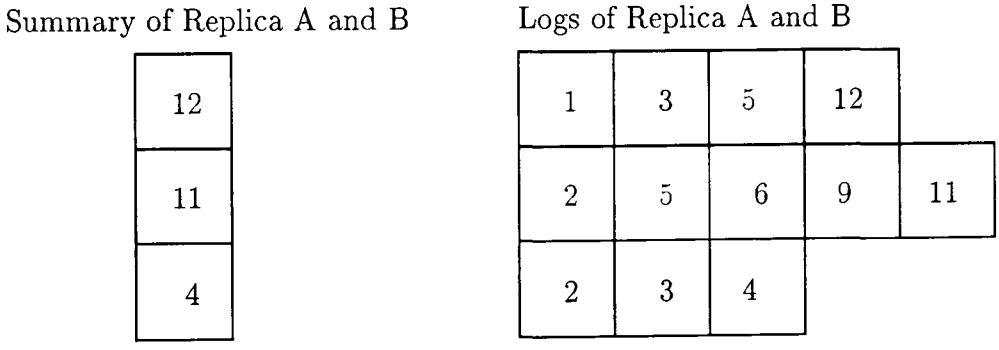
Timestamped anti-entropy protocols can provide several different message delivery orderings, including total, per-process, or no ordering. Causal orderings are possible if the process clocks meet Lamport's happens-before condition [72]. The algorithm can be described in short as follows:

Timestamped anti-entropy protocols maintain three data structures: a message log and two timestamp vectors. The *message log* contains messages that have been received by a process. Processes maintain a *summary timestamp vector* that records the timestamp of last update for each replica as all updates before this have been received by the process. The third data structure is the *acknowledgement timestamp vector* that records what messages have been acknowledged by other processes. From time to time, a process selects another process and initiates an anti-entropy session. During this session the two processes first exchange their summary and acknowledgement vectors. Based on these vectors the two processes determine if one of them has messages that the other has not yet observed. The messages are then exchanged using a reliable stream protocol. To explain how the protocol works we reproduce

here an example of anti-entropy session given in [29] (see Figure 2.4).




(a) Before Exchange




(b) After Exchange

Figure 2.4: An example of anti-entropy session

### 2.5.3 Multilevel Consistency Protocols

Use of replication techniques in distributed environments with thousands or even more nodes connected through a wide area network motivated the need for algorithms to manage replicated data that are scalable and also ensure properties like availability and speed of retrieval of data. Distributed systems that scale are organized hierarchically to exploit locality of reference [24]. Based on this fact many researchers have proposed algorithms where replicas are organized in a hierarchy of two or more levels. These protocols maintain some replicas in strongly consistent state while allowing others to become out-of-date. We describe some protocols that fall into this category.

#### Quasi-copy

The quasi-copies algorithm was proposed by Daniel Barbara and H. Gracia-Molina in [12]. Quasi-copies are replicated copies that may be somewhat out of date but are guaranteed to meet a certain consistency predicate. With quasi-copies, it is assumed that a central location exists, where all the updates are processed, and several copies are located throughout the network. A predicate is associated with each copy, establishing the degree of inconsistency that can be tolerated. For instance, the copy must not be more than ten minutes old. The system guarantees that this predicate is not violated when updates occur. This can be done in two ways depending on who is responsible for the consistency, central node or client.

#### Universal Name Service

C. Ma in [46] proposed Universal Name Service that tries to integrate both strong and weak consistencies. Replicas are grouped into first class servers and secondary servers. The first class replicas use quorum based scheme to implement strong consistency whereas the secondary replicas use anti-entropy method described in [22].

Only the first class servers carry out updates. These updates are then propagated to secondary servers using *push-pull* techniques described in [22]. Secondary servers are used for read only operations that do not necessarily require most recent version of the data.

### Hierarchical Asynchronous Replication Protocol

HARP also tries to integrate strong and weak consistency into the same framework by supporting a set of operations that need different level of consistency. The protocol takes the advantage of the physical hierarchy present in the large networks to organize replicas into a logical multilevel hierarchy [2]. In this hierarchy nodes are grouped into clusters (normally all the nodes belonging to the same LAN) and clusters are organized into a tree, such that each cluster is assigned a father node in its parent cluster. The replicas in the root or top cluster maintain strong consistency by the use of quorums whereas the replicas at other levels are weakly consistent. The algorithm propagates a message in the following way: A node $i$, originating a message, sends it to its neighbours, parent and children. This works recursively and a message originated at any site is propagated everywhere. The protocol supports the following set of operations: A fast read and a fast write that can be initiated and completed at any replica. The value returned by a fast read may not be the most recent value of the data. For applications that need strong consistency the protocol supports operations slow read and slow write. A slow read (slow write) can be initiated at any replica but it is implemented only after collecting a read (write) quorum from the replicas at the root level. The protocol also provides Opt-Write which is similar to slow write, but it is applied to the database of the site of origin and, optionally, to some other selected replicas.

Fast write can create temporary inconsistencies in the database. The reconciliation

methods supported by the protocol are based on the delivery order mechanisms. The logical hierarchy of replicas can also be reorganized to cope up with the actual physical changes in the network.

### Two-tier Replication

Use of replication techniques in an environment where users and services are mobile need an entirely new approach for maintaining replicas. Most of the nodes are disconnected most of the time and can not communicate with each other. In [31] J. Gray et al. show that update anywhere anytime anyway transactional replication has unstable behaviour as the workload scales up. A ten-fold increase in nodes and traffic gives a thousand fold increase in deadlocks and reconciliations. They suggest a two-tier approach for replication in mobile systems. This approach allows mobile applications to propose tentative update transactions that are later applied to a master copy. There are two kind of nodes. *mobile nodes* are disconnected most of the time whereas *base nodes* are always connected. Replicated data items have two versions at mobile nodes. The most recent value received from the object master which is called *master version* and the most recent value due to local updates called *tentative version.* Similarly there are two kind of transactions. *Base Transactions* that work only on master data and they produce new master data. *Tentative transactions* that work on local tentative data to produce new tentative versions. The basic idea behind the scheme can be explained as follows: Each object has a master node. Mobile nodes accumulate tentative transactions that run against the tentative database stored at the node. They are reprocessed as base transactions when the mobile node reconnects to the base. Tentative transactions may fail when reprocessed.

## 2.6 Models studied in this thesis

This thesis evaluates the performance of some of the protocols described in previous section. Chapter 3 evaluates the performance of Weighted Voting protocol that maintains strong consistency. Both the cases of reliable and unreliable replication have been analyzed. The model assumes that all replicas are identical and write jobs have priority over read jobs. Chapter 4 studies the performance of Weighted Voting protocol when both read and write jobs share the same queue and get the service on first-in-first-out basis. A comparison of both scheduling strategies has also been presented. In chapter 5 we present the analysis of the model that contains a single unreliable server and two type of jobs with type 1 jobs having priority over type 2 jobs. Chapter 6 presents the analysis of a two level consistency protocol. As in case of HARP replicas at level 0 maintain strong consistency with the help of quorums but replicas at level 1 may contain out-of-date information. There are three type of operations that arrive in the system: fast read that may read older versions of data, slow reads that need an up to date copy of data and write. Finally in chapter 7 we study the performance of schemes where updates are allowed to occur asynchronously on any replica which then propagates these updates to other replicas. We study the performance of the schemes for propagating updates described in Lazy replication and timestamp anti-entropy protocols.

# Chapter 3

# Weighted Voting Protocol

## 3.1 Introduction

In the previous chapter we describe protocols for maintaining strong consistency by means of quorums. Almost all these protocols are variants of the Weighted Voting protocol. In this chapter we present the performance analysis of the Weighted Voting protocol. We first describe a model for the protocol and then present its analysis. There are many studies evaluating the performance of quorum based protocols with availability as the performance measure. These do not take congestion and queueing of the jobs into account. However, poor performance can be caused both by breakdowns and by congestion. If the response time of an operation increases over a certain value (the maximum time for which the user can wait for the response) the data may be considered as unavailable. This may be because the server is down or the queue is too long. We therefore choose a modelling approach based on queueing theory and use the response time of the operations as the performance measure. This approach takes both breakdowns and congestion and queueing into account. Chapter 1 mentions some work that evaluates the performance of data replication based on the queueing

theory approach. This either assumes that any number of reads can be executed in parallel or treats write requests as interruptions whereas the models presented in this chapter have two separate read and write queues. We first consider the case of $N$ reliable replicas with two separate unbounded queues for read and write jobs and present an exact analysis of this model. We study the effect of increasing the number of replicas and the effect of changing read and write quorum sizes on the response time of operations (read and write). We then evaluate the performance of the protocol when replicas are subject to random breakdowns and repairs. The analysis presented in case of latter is approximate as it considers that the queue for write jobs is bounded. We finally compare the approximate analytical results for the second case with simulation results.

## 3.2 Reliable Replicas

### 3.2.1 Model

The model presented in this section considers that replicas are fully reliable. We extend this model to consider the case of breakdowns and repairs in section 3.3. There are $N$ servers, each managing a copy of the data. We assume that these servers are identical and each contains only one vote. Two types of jobs, write and read, arrive into the system in independent Poisson streams with rates $\lambda_1$ and $\lambda_2$ respectively.

The service of a write job requires the simultaneous possession of $W$ servers (write quorum), which are held for an exponentially distributed interval with mean $1/\nu_1$ and then released. A read service requires $R$ servers (read quorum) and is exponentially distributed with mean $1/\nu_2$. As a read and a write service should never take place in parallel the read and write quorums should intersect each other. To ensure this

$R + W = N + 1$. Moreover, at most 1 write job can be in service at any moment, regardless of the value of $W$. We assume that some concurrency control mechanism exists to ensure this.

There are two separate unbounded queues for read and write jobs and write jobs have preemptive priority over reads. If an arriving write job finds that a write service is in progress it joins the write queue otherwise it preempts all the read services in progress. The read jobs preempted by the arrival of a write service join the read queue. A read service can start only if write queue is empty and a read quorum is available. The maximum number of read jobs that can be served in parallel is $r = \lfloor N/R \rfloor$, where $\lfloor x \rfloor$ is the integer part of $x$.

The parameters $\nu_1$ and $\nu_2$ depend, in general, on the quorum sizes. An access that engages a larger number of servers can be expected to take longer. Thus, the average write service times usually increase with $W$, and the read ones increase with $R$. The nature of that increase depends on the way read and write operations are implemented. If all replicas in a quorum are accessed in parallel, then it is reasonable to assume that

$$\frac{1}{\nu_1} = \frac{1}{\mu_1} \sum_{k=1}^{W} \frac{1}{k} \quad ; \quad \frac{1}{\nu_2} = \frac{1}{\mu_2} \sum_{k=1}^{R} \frac{1}{k} \,, \tag{3.1}$$

for some fixed $\mu_1$ and $\mu_2$. Those would be the averages of *the largest* of $W$ (respectively $R$) i.i.d. random variables, each distributed exponentially with mean $1/\mu_1$ (respectively $1/\mu_2$,). If, on the other hand, the operations are performed sequentially on all replicas, then average service time for read and write are given by:

$$\frac{1}{\nu_1} = \frac{W}{\mu_1} \quad ; \quad \frac{1}{\nu_2} = \frac{R}{\mu_2} \,, \tag{3.2}$$

(that, together with $W = N$ and $R = 1$, was the assumption in [11]).

The model is illustrated in Figure 3.1.

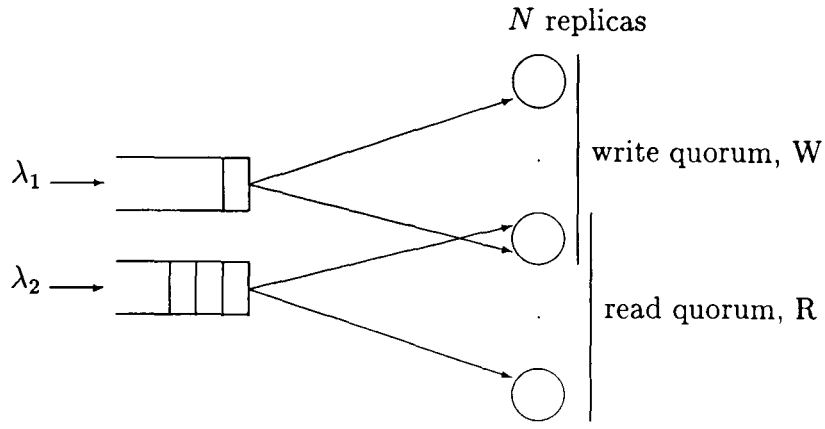



Figure 3.1: Data replication with read and write accesses

### 3.2.2 Analysis

Let $I(t)$ and $J(t)$ be the numbers of write and read jobs in the system at time $t$. Under the above assumptions, the pair $[I(t), J(t)]$ is an irreducible Markov process on the state space $\{0, 1, \ldots\} \times \{0, 1, \ldots\}$. Since the largest number of services in progress at any time can be either 1 write or $r$ read (but not both), the ergodicity condition is

$$\frac{\lambda_1}{\nu_1} + \frac{\lambda_2}{r\nu_2} < 1 \,. \tag{3.3}$$

That condition will be assumed to hold. The object of the analysis is to determine the steady-state joint distribution of $I$ and $J$, denoted by $p_{i,j}$:

$$p_{i,j} = \lim_{t\to\infty} P[I(t) = i, J(t) = j] \ ; \ i,j = 0,1,\ldots .$$

These probability satisfy the following set of balance equations:

$$p_{i,j}[\lambda_1 + \lambda_2 + \nu_1\delta(i>0) + min(j,r)\nu_2\delta(i=0)] = \lambda_1 p_{i-1,j} + \lambda_2 p_{i,j-1}$$

$$+\nu_1 p_{i+1,j} + min(j+1,r)\nu_2\delta(i=0)p_{i,j+1} \ ; \ i,j = 0,1\ldots , \tag{3.4}$$

where $p_{-1,j} = 0$ and $p_{i,-1} = 0$ by definition, and $\delta(B)$ is the indicator function: 1 if $B$ is true, 0 otherwise.

To solve these equations we define the generating function

$$g(x,y) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} p_{i,j}x^i y^j \ . \tag{3.5}$$

Multiplying (3.4) by $x^i y^j$ and summing over all $i$ and $j$ yields

$$(\lambda_1+\lambda_2)g(x,y) + \nu_1[g(x,y) - g(0,y)] + \nu_2[rg(0,y) - \sum_{j=0}^{r-1}(r-j)p_{0,j}y^j]$$

$$= \lambda_1 x g(x,y) + \lambda_2 y g(x,y) + \frac{\nu_1}{x}[g(x,y) - g(0,y)]$$

$$+\frac{\nu_2}{y}[rg(0,y) - \sum_{j=0}^{r-1}(r-j)p_{0,j}y^j] \ .$$

After some manipulations we get:

$$ya(x,y)g(x,y) = [\nu_1 y(x-1) + r\nu_2 x(1-y)]g(0,y)$$

$$+\nu_2 x(y-1)\sum_{j=0}^{r-1}(r-j)p_{0,j}y^j \ , \tag{3.6}$$

where $a(x,y) = \lambda_1 x(1-x) + \lambda_2 x(1-y) + \nu_1(x-1)$, and

$$g(0,y) = \sum_{j=0}^{\infty} p_{0,j}y^j \ . \tag{3.7}$$

The bivariate function $g(x,y)$ is thus expressed in terms of a single-variable unknown function, $g(0,y)$, and $r$ unknown constants, $p_{0,j}(j=0,1,\ldots,r-1)$. The latter are the first $r$ coefficients in the expansion of the former.

To eliminate $g(0,y)$, note that whenever $a(x,y)=0$ and $g(x,y)$ is finite, the right-hand side of (3.6) must vanish. Fix an arbitrary real $y \in (0,1)$, and consider $a(x,y)$ as a polynomial in $x$. This is a quadratic which satisfies $a(0,y)<0$, $a(1,y)>0$ and $a(\infty,y)<0$. It therefore has exactly one zero in the interval $(0,1)$ and one zero in the interval $(1,\infty)$. Denote the smaller of these by $\alpha(y)$. At the point $[\alpha(y),y]$, the generating function $g(.,.)$ is finite and hence the right-hand side of (3.6) is 0. This gives

$$g(0,y)=\frac{\nu_2\alpha(y)(1-y)\sum_{j=0}^{r-1}(r-j)p_{0,j}y^j}{r\nu_2\alpha(y)(1-y)-\nu_1 y[1-\alpha(y)]} \quad . \tag{3.8}$$

The only remaining unknowns are now the $r$ probabilities $p_{0,0},p_{0,1},\ldots,p_{0,r-1}$. To determine them, rewrite (3.8) in the following form:

$$\begin{aligned} &r\nu_2\alpha(y)(y-1)y^{-1}[g(0,y)-\sum_{j=0}^{r-1}p_{0,j}y^j]+\nu_1[1-\alpha(y)]g(0,y) \\ &\quad=\nu_2\alpha(y)(1-y)\sum_{j=1}^{r-1}jp_{0,j}y^{j-1} \quad . \end{aligned} \tag{3.9}$$

The definition of $g(0,y)$ implies that the first term in the left-hand side of (3.9) has a factor $y^{r-1}$. Therefore, that term and its first $r-2$ derivatives vanish at $y=0$.

Setting $y=0$ in (3.9) yields

$$\nu_1[1-\alpha(0)]p_{0,0}=\nu_2\alpha(0)p_{0,1} \quad . \tag{3.10}$$

Differentiating (3.9) once with respect to $y$ and setting $y=0$, gives

$$\nu_1\{-\alpha'(0)p_{0,0}+[1-\alpha(0)]p_{0,1}\}=\nu_2\{[\alpha'(0)-\alpha(0)]p_{0,1}+2\alpha(0)p_{0,2}\} \quad . \tag{3.11}$$

Continuing in this way, differentiating (3.9) $i$ times and setting $y = 0$, we get

$$\sum_{j=0}^{i} C_{i,j}[(j+1)!A^{i-j}(0)p_{0,j+1} - j!B^{i-j}(0)p_{0,j}] = 0 \tag{3.12}$$

where

$$C_{i,j} = \begin{cases} 1 & ; \ j = 0, i \\ C_{i-1,j-1} + C_{i-1,j} & \text{otherwise} \end{cases} .$$

$A^j(0)$ is the $jth$ derivative of $\{\nu_2\alpha(y)(1-y)\}$ at $y = 0$ and $B^j(0)$ is the $jth$ derivative of $\{\nu_1(1-\alpha(y))\}$ at $y = 0$. Taking derivatives up to order $r-2$ inclusive and setting $y = 0$, provides a set of $r-1$ homogeneous linear equations for the unknown probabilities. The derivatives of $\alpha(y)$ at $y = 0$ are obtained by differentiating the equation $a[\alpha(y), y] = 0$ and setting $y = 0$.

To the above equations we add a non-homogeneous normalizing equation. A simple form of the latter is obtained by noting that the marginal distribution of the number of write jobs in the system is that of an $M/M/1$ queue with parameters $\lambda_1$ and $\nu_1$. Therefore, we can set $y = 1$ in (3.8) and use the fact that

$$g(0,1) = 1 - \frac{\lambda_1}{\nu_1} \ . \tag{3.13}$$

All unknowns are now determined. From the generating functions one can compute various performance measures. In fact, the write response time can be calculated directly using the results for $M/M/1$ queue and is given by

$$W_1 = \frac{1}{\mu_1 - \lambda_1}$$

The average response time for low priority read jobs $W_2$, is given by

$$\lambda_2 W_2 = \frac{\partial}{\partial y} g(1,1) \tag{3.14}$$

In evaluating $g(1,1)$ and the corresponding derivatives, L'Hospital's rule is used to resolve indeterminacies of type 0/0.

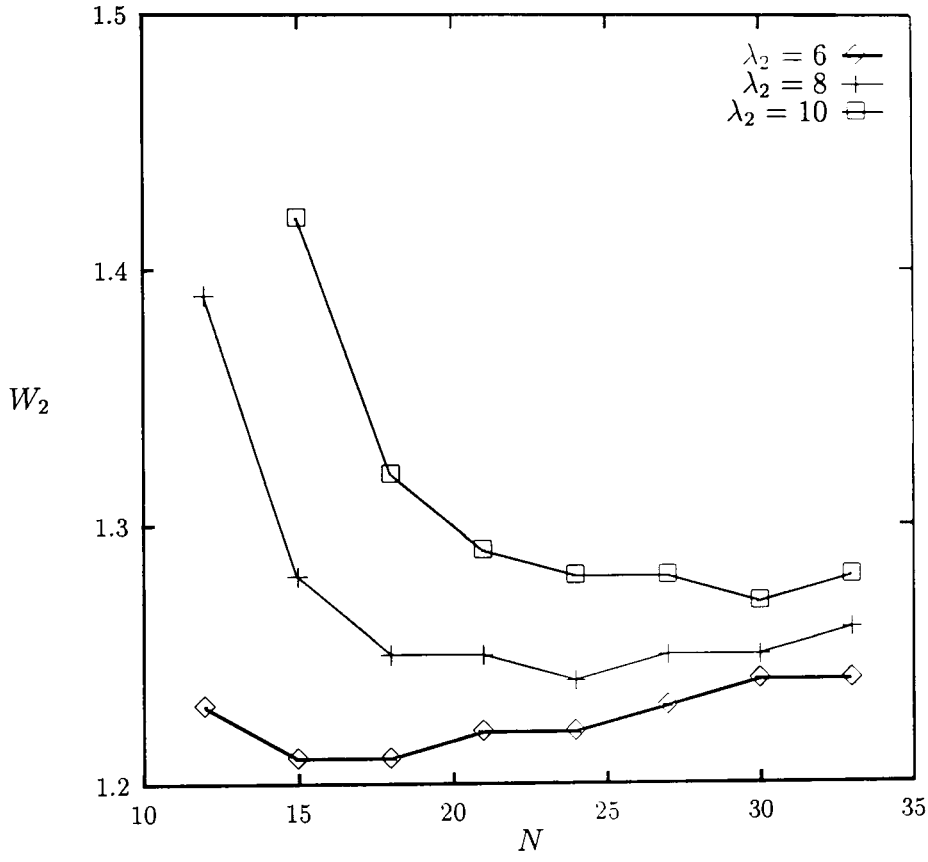
### 3.2.3 Results of Numerical experiments

We first examine the effect of increasing number of replicas when (fully reliable) read and write quorum sizes are $R = 1$, $W = N$. As the response time of write jobs is the response time of jobs for an $M/M/1$ queue, it is of little interest. The performance measure of interest is the average response time for read jobs, $W_2$ (presumably they constitute the bulk of the demand). The trade-off here is between the advantage of increased parallelism for read jobs and the disadvantage of longer service times for write jobs. Figure 3.2 shows some results for different parameter values. The response time of read jobs first decreases and then increases. In all cases, there is an optimal degree of replication which is lower when the read job arrival rate is lower. This behaviour can be explained as follows. The increase in number of replicas causes:

- more reads to execute in parallel which decreases overall read response time.
- write service time increases. This reduces the time for which the system is available for read service and increases read response time.

In the beginning the effect of former is more than the increase in write service time which decreases read response time. After a certain degree of replication the latter dominates and read service time increases. However, the curves are quite shallow in the regions of their minima. The convexity would increase if the replicas were updated sequentially rather than in parallel (see [11]).

The effect of changing the quorum sizes, with the number of replicas fixed (again all fully reliable), is illustrated in Figure 3.3. This time the behaviour is much less




$\lambda_1 = 0.01,\ \mu_1 = 1,\ \mu_2 = 1$

Figure 3.2: Read response time as function of $N$ ; no breakdowns

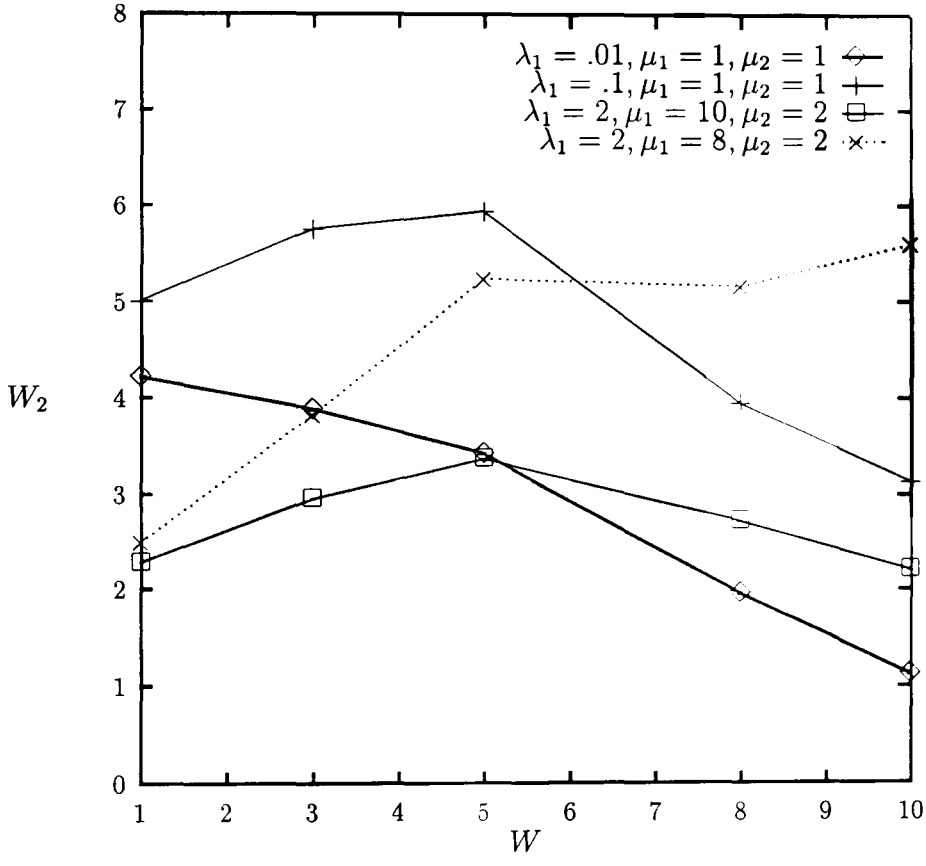
predictable. Increasing the read quorum causes the performance measure sometimes to increase, sometimes to decrease, and sometimes to increase and then decrease. The only reasonably general and intuitive observation that can be made is that when most of the offered load consists of read jobs, the allocation $R = 1$, $W = N$ is best, whereas $R = N$, $W = 1$ is preferable if most of the load consists of write jobs. It should be pointed out that if the performance measure is the overall average response time (including write jobs), rather than $W_2$, the situation is similar.

## 3.3 Unreliable Replicas

### 3.3.1 Model

Let us relax the assumption that the servers are fully reliable. There are $N$ identical servers. Each server goes through alternating periods of being operative and inoperative, independently of the others. Those periods are exponentially distributed with means $1/\xi$ and $1/\eta$, respectively. The arrival streams for write (type 1) and read (type 2) jobs are Poisson with rates $\lambda_1$ and $\lambda_2$, respectively. Each write service requires $W$ operative servers and at most one of them can be in progress at any time. A read service requires $R$ operative servers with $(R = N + 1 - W)$.

Write jobs have preemptive priority over reads whenever possible. A new arrival of a write job begins service immediately if there is no write job in the system and $W$ servers are operative. Otherwise it joins the write queue. Reads preempted by a write service join the read queue. A read job begins service if both a read quorum of $R$ servers is available and either there is no write job in the system, or a write quorum is not available. Thus, if there are both write and read jobs in the system, and the number of operative servers are $m$ then




$N = 10,\ \lambda_2 = 0.1$

Figure 3.3: Changing the write quorum size, fixed $N$; no breakdowns

- if $m \geq W$, one write job is served else
- if $R \leq m < W$, then $\lfloor m/R \rfloor$ read jobs are served in parallel

The average write and read service times, $1/\nu_1$ and $1/\nu_2$, are given by (3.1) or (3.2) depending on whether all replicas in quorum are accessed in parallel or sequentially. Services interrupted by either breakdowns or preemptions are eventually resumed from the point of interruption.

### 3.3.2 Analysis

The exact analysis of this model with both queues unbounded is, at present, intractable. We shall provide an approximate solution by assuming that queue 1 cannot exceed size $S$. Write jobs arriving when there are already $S$ of them in the system are lost. The accuracy of this approximation clearly increases with $S$, but so does its numerical complexity. However, it is possible to obtain accurate results with small values of $S$ when the offered load due to the write jobs, $\lambda_1/\nu_1$, is small compared to the processing capacity available to them, $c_1$. The latter is equal to the probability that there are at least $W$ operative servers:

$$c_1 = \sum_{j=W}^{N} \binom{N}{j} \frac{\eta^j \xi^{N-j}}{(\xi+\eta)^N} . \tag{3.15}$$

The system state at time $t$ is described by three integers, $K(t)$, $I(t)$ and $J(t)$, denoting the numbers of operative servers, write jobs present and read jobs present, respectively. The first two of these have finite ranges and it is convenient to replace them by a single integer, $U(t) = (N+1)I(t)+K(t)$, which takes values $0, 1, \ldots, NS+N+S$. When $U(t) < N+1$, there are $U(t)$ operative servers and the write queue is empty; if $N+1 \leq U(t) < 2(N+1)$, there are $U(t) - N - 1$ operative servers and 1

write job; etc. That random integer can be thought of as a Markovian environment which controls the behaviour of queue 2.

The parameters governing the transitions of the Markov process $[U(t), J(t)]$ can be classified according to whether $J(t)$ remains the same, jumps up by 1 or jumps down by 1. They are:

- The matrix $A = [a_{i,k}]_{i,k=0}^{NS+N+S}$, where $a_{i,k}$ is the instantaneous rate at which the environment $U$ jumps from state $i$ to state $k$. The diagonal elements of $A$ are equal to 0.
- The read job arrival rate, $\lambda_2$.
- The row vector $\boldsymbol{\sigma}_j = (\sigma_{0,j}, \sigma_{1,j}, \ldots, \sigma_{NS+N+S,j})$, where $\sigma_{i,j}$ is the rate at which read jobs are served when the environment is in state $i$ and their number is $j$.

The elements of matrix $A$ are given by

$$a_{i,k} = \begin{cases} \lambda_1 & ;\ k = i+N+1\ ;\ k \leq NS+N+S \\ \nu_1 & ;\ i \bmod (N+1) \geq W\ ;\ k = i-N-1\ ;\ k \geq 0 \\ m\xi & ;\ i \bmod (N+1) = m > 0\ ;\ k = i-1 \\ (N-m)\eta & ;\ i \bmod (N+1) = m < N\ ;\ k = i+1 \\ 0 & \text{otherwise} \end{cases},$$

while those of $\boldsymbol{\sigma}_j$ are

$$\sigma_{i,j} = \begin{cases} \min(\lfloor i/R \rfloor, j)\nu_2 & ;\ i < N+1 \\ \min(\lfloor m/R \rfloor, j)\nu_2 & ;\ i \bmod (N+1) = m < W \\ 0 & \text{otherwise} \end{cases}.$$

Note that $\boldsymbol{\sigma}_j$ is $\mathbf{0}$ when $j = 0$, and is independent of $j$ when $j \geq r = \lfloor N/R \rfloor$.

Let $p_{i,j}$ be the steady-state probability that the environment is in state $i$ and the number of read jobs in the system are $j$:

$$p_{i,j} = \lim_{t\to\infty} P[U(t) = i, J(t) = j] \ ; \ \ i = 0, 1, \ldots, NS + N + S \ ; \ \ j = 0, 1, \ldots \ .$$

Define the row vectors

$$\mathbf{v}_j = (p_{0,j}, p_{1,j}, \ldots, p_{NS+N+S,j}) \ ; \ j = 0, 1, \ldots \ .$$

These vectors satisfy the following balance equations:

$$\mathbf{v}_j[\lambda_2 I + D^A + C_j] = \mathbf{v}_{j-1}\lambda_2 I + \mathbf{v}_j A + \mathbf{v}_{j+1}C_{j+1} \ ; \ j = 0, 1, \ldots \ , \tag{3.16}$$

where $I$ is the unit matrix of order $(N+1)(S+1)$, $D^A$ is the diagonal matrix whose $i$'th diagonal element is the $i$'th row-sum of $A$, and $C_j$ is the diagonal matrix whose diagonal is $\boldsymbol{\sigma}_j$. In addition, we have the normalizing equation

$$\sum_{j=0}^{\infty} \mathbf{v}_j \mathbf{e} = 1 \ , \tag{3.17}$$

where $\mathbf{e}$ is the column vector with $(N+1)(S+1)$ elements equal to 1.

The solution of (3.16) and (3.17) can be obtained by spectral expansion (for more details, see [50]). When $j \geq r$, the coefficients in (3.16) do not depend on $j$. Those equations can then be rewritten in the form

$$\mathbf{v}_j Q_0 + \mathbf{v}_{j+1} Q_1 + \mathbf{v}_{j+2} Q_2 = \mathbf{0} \ ; \ j = r-1, r, \ldots \ , \tag{3.18}$$

where $Q_0 = \lambda_2 I$, $Q_1 = A - D^A - \lambda_2 I - C_r$ and $Q_2 = C_r$. Associated with this homogeneous vector difference equation of order 2 is the characteristic matrix polynomial, $Q(z)$, defined as

$$Q(z) = Q_0 + Q_1 z + Q_2 z^2 \,. \tag{3.19}$$

Denote by $z_\ell$ and $\psi_\ell$ the eigenvalues and corresponding left eigenvectors of $Q(z)$. In other words, these are quantities which satisfy

$$\psi_\ell Q(z_\ell) = 0 \;;\; \ell = 1, 2, \ldots, d \,, \tag{3.20}$$

where $d = degree\{det[Q(z)]\}$.

When the process is ergodic, $(N+1)(S+1)$ of the eigenvalues of $Q(z)$ are strictly inside the unit disk (each counted according to its multiplicity), while the others are on the circumference or outside (see [50]). Indeed, verifying this condition is the way to establish ergodicity for this model, since we no longer have a simple inequality like (3.3). Let the numbering be such that $|z_\ell| < 1$ for $\ell = 1, 2, \ldots, (N+1)(S+1)$. The corresponding independent eigenvectors are $\psi_1$, $\psi_2$, ..., $\psi_{(N+1)(S+1)}$. Then any solution of equation (3.18) which can be normalised to a probability distribution is of the form

$$\mathbf{v}_j = \sum_{\ell=1}^{(N+1)(S+1)} x_\ell \psi_\ell z_\ell^j \;\;;\;\; j = r-1, r, \ldots \,, \tag{3.21}$$

where $x_\ell$ $(\ell = 1, 2, \ldots, (N+1)(S+1))$, are arbitrary (complex) constants.

It remains to determine the coefficients $x_\ell$ and the vectors $\mathbf{v}_j$ for $j < r-1$, which is a total of $r(N+1)(S+1)$ unknown constants. The balance equations (3.16) for $j < r$, and (3.17), provide exactly the required number of independent linear constraints.

For computational purposes, the quadratic eigenvalue-eigenvector problem (3.20) can be reduced to the common linear one of the form $\psi V = z\psi$ (see appendix). However, the order of the matrix $V$ is double that of $Q$ (see [50]). Routines for solving the latter problem are available in most numerical packages.

Once all probabilities are known the response time of read jobs can be computed as following where $\varphi_\ell$ is the sum of all elements of $\psi_\ell$:

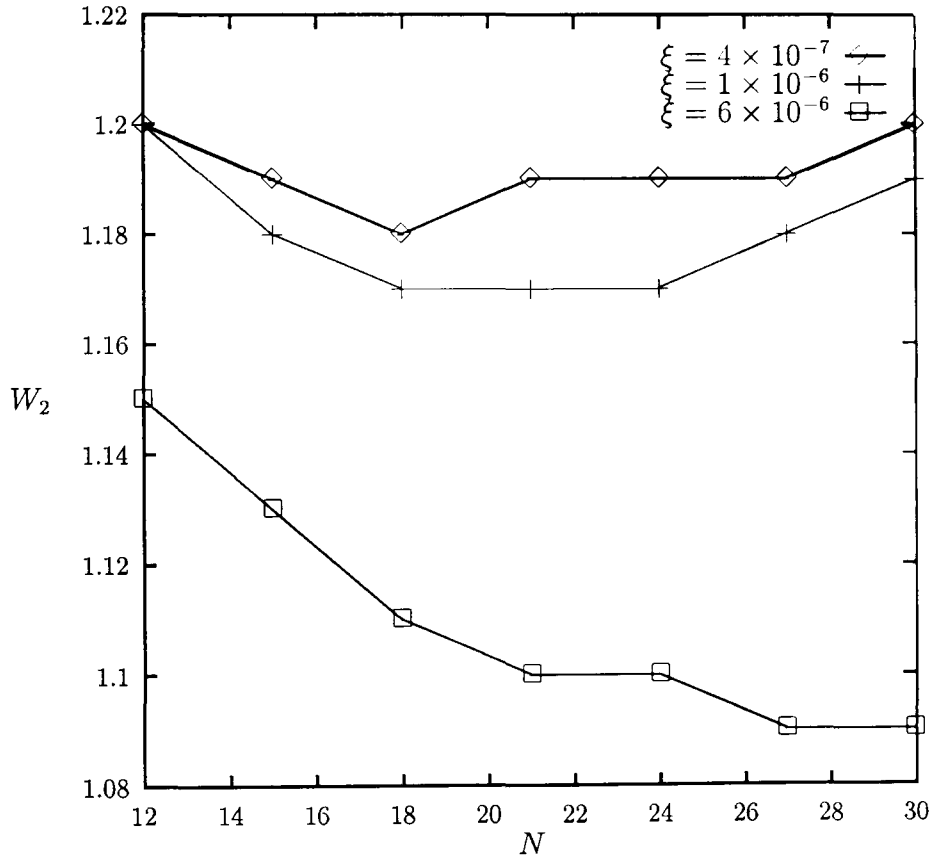
$$\lambda_2 W_2 = \sum_{i=0}^{NS+N+S} \sum_{j=1}^{r-1} j p_{i,j} + \sum_{\ell=1}^{(N+1)(S+1)} \frac{(r-(r-1)z_\ell)x_\ell \varphi_\ell}{(1-z_\ell)^2} \tag{3.22}$$

### 3.3.3 Results of Numerical experiments

A system with breakdowns and repairs is modeled in Figure 3.4. The arrival and service rates are the same as for one of the curves in Figure 3.2. We study the effect of different breakdown rates on response time of read jobs. The average read response time is plotted against the number of replicas. The quorum sizes are $R = 1$ and $W = N$. A notable feature of the results is that increasing the rate of breakdowns, for fixed $N$, leads to a reduction in the read response time. This seems counter-intuitive, but is not: breakdowns deny write jobs a quorum and allow read jobs to be served, thus in effect relaxing the strict priority rule.

Again, the trade-off between more parallelism for reads and longer service time for writes implies that there is an optimal degree of replication. Moreover, our intuition tells us that the presence of breakdowns should generally make that optimal degree larger; that is confirmed by the experiments.

The last set of results deal with the role of quorum size in a model with breakdowns. The experiment illustrated in Figure 3.5 mirrors the one in Figure 3.3, as far as arrival and service parameters are concerned. It can be seen that even a slight unreliability of the servers (each of them is operative more than 99% of the time) can have a considerable effect on the shape of the curves. Now the quorum sizes $W = N$, $R = 1$ are optimal for all parameter values in the figure. However, if the performance measure is the overall average response time (including the write jobs), then it is




$\lambda_1 = 0.01,\ \lambda_2 = 6,\ \mu_1 = 1,\ \mu_2 = 1,\ \eta = .0002$

Figure 3.4: Read response time as function of $N$; breakdowns

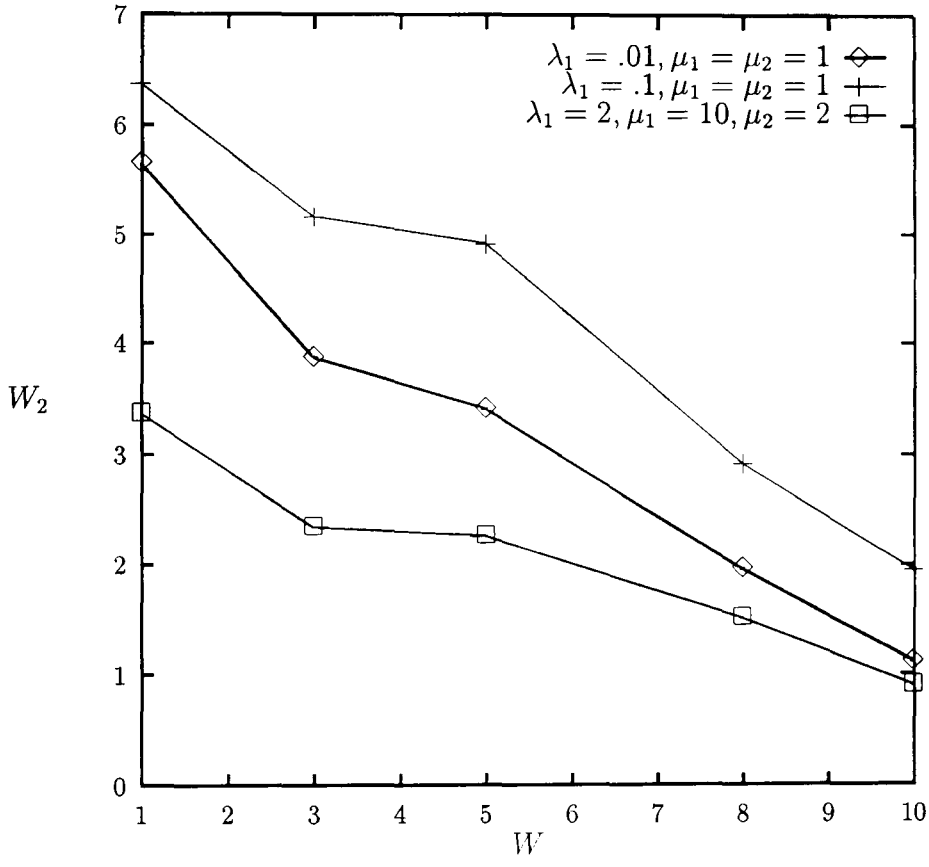
again possible for the allocation $W = 1$, $R = N$ to be optimal.

Figure 3.6 shows a comparison of approximate analytical results with simulation results. It can be seen that when ratio $\lambda_1/\mu_1$ is fairly low the simulation results are almost same as analytical results but when ratio $\lambda_1/\mu_1$ is comparable to the processing capacity available to them then the difference in two is significant. This is because in the former case the probability that at any time there will be more than one write in the system is negligible which is not true in case of latter. The analytical results shown in Figure 3.6 are with $S = 1$. They can be further improved by increasing the size of $S$.

## 3.4 Generalizations

Several modifications and generalizations of the models presented in section 3.2.1 and 3.3.1 may be considered. In the Weighted Voting algorithm any number of votes may be assigned to a server. A write (read) service collects a write (read) quorum of $W(R)$ votes. If the number of votes assigned to different servers are different, each read or write service may engage different number of servers for its service depending on the number of votes assigned to them. Due to this the service time of different write (read) jobs may be different even if the service requirement at each server is same. This is because the service time depends on the number of servers in read or write quorums.

The servers themselves may not be identical. The service time of a read or write job may be different for each server. For example communication delay in contacting a server may be included in service time which will vary depending on the location of servers. In this case even if the service of a read or write job takes the same time on each server the total time (service time + communication delay) to get the service




$\xi = 4 \times 10^{-7},\ \eta = 2 \times 10^{-3}, \lambda_2 = 0.1$

Figure 3.5: Changing the write quorum size, fixed $N$; breakdowns

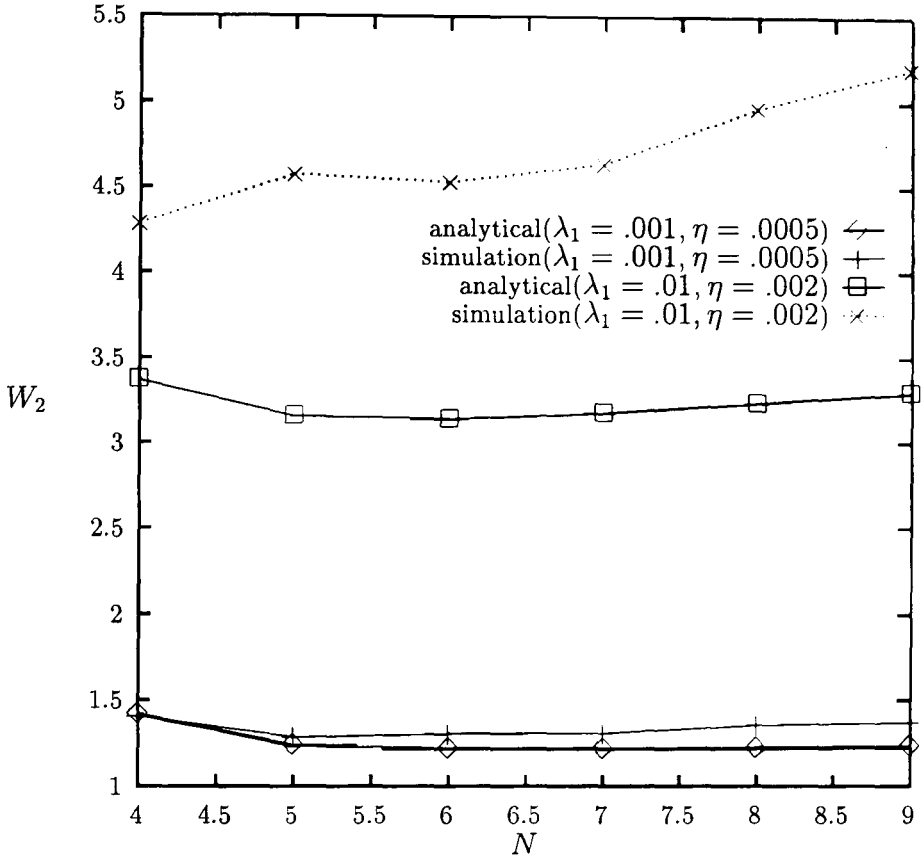



Figure 3.6: Analytical ($S = 1$) vs. simulation results

$\xi = 1 \times 10^{-5}$, $\lambda_2 = 2$, $\mu_1 = .2$, $\mu_2 = 1$

will be different for each server. So the read (write) service time will be different each time depending on which servers are in the quorum.

The above generalizations can be handled, approximately, by the methods presented in sections 3.2.2 and 3.3.2 provided that the following is true:

- read and write quorums are one and all respectively and
- read service time is same for all read services. This can be safely assumed when a user reads from its local server only.

Let $F_i(x)$ be the distribution of the write service time on $ith$ server. Then the average write service time can be calculated using following equation:

$$\frac{1}{\nu_1} = \int_0^\infty [1 - \prod_{i=1}^{N} F_i(x)]dx \ , \tag{3.23}$$

One would then make the approximating assumption that the write service times are distributed exponentially with parameter $\nu_1$. When read and write quorums are different from one and all, analysis becomes complicated.

Some other generalizations that may be considered:

- In practice the priority given to write jobs may be non-preemptive, rather than preemptive. While this is unlikely to make a big difference to the performance of the system, the analysis would become considerably more complicated.
- Some replication protocols require an update operation to be performed after a breakdown and the subsequent repair, with priority over any read jobs that may be in service. In other words, a write job is injected into the system at the end of a repair period. This modification can be handled by the method described in section 3.3.2; the state space of the environment variable $U(t)$ is enlarged.

- Recent proposals have introduced a hierarchy of replicas — primary, secondary etc. — with different scheduling policies at each level. This is a substantial generalization which we study in chapter 6.

## 3.5 Conclusion

The models presented here provide useful insights into the behaviour of replicated data systems. The effects of different parameters can be evaluated and optimal decisions concerning the degree of replication and quorum sizes can be taken. The solution of the model without breakdowns is exact; its numerical complexity is on the order of $O(N^3)$ (solving a set of $2N$ simultaneous linear equations). The model with breakdowns is solved approximately but as accurately as desired, subject to constraints on computing resources. That solution involves finding the eigenvalues and eigenvectors of a matrix polynomial, and solving a set of simultaneous linear equations; its complexity is on the order of $O[(NS + N + S)^3]$, where $S$ is the imposed bound on the number of write jobs in the system. We also compare the approximate analytical results obtained with breakdowns and simulation results to show that when offered load due to the write jobs is small compared to the processing capacity available to them, the two results are almost same. In chapter 5 we present an exact model for one server with breakdowns and two unbounded queues and for this special case of one server we compare exact and approximate results obtained for the two cases of unbounded and bounded queues.

# Chapter 4

# Effect of scheduling strategies

## 4.1 Introduction

In quorum based protocols, described in chapter 2, every read (write) job collects a read quorum (write quorum) before execution. The procedures that collect quorum use facilities provided by the language / operating system such as setting read or write locks on replicas. Gifford, in [25], describes procedures to implement the Weighted Voting Protocol. These procedures are in language Mesa and use the monitor facility provided by Mesa for manipulating shared data. There are several studies that examine the effect of scheduling strategies on the performance of the system for classical readers/writers problem that uses these facilities for mutual exclusion. These studies do not consider the issue of replication and so do not study the effect of quorums on performance. In chapter 2 we mentioned some work of E. G. Coffman *et al.* [16] and F. Baccelli and E.G. Coffman [11] that studies the performance of replication schemes using priority scheduling for read one write all policy (described in chapter 2). In [58] Nelson and Iyer present an analysis of data replication for read one write all policy when reads and writes are served in a first come first to serve discipline. None of them

has analyzed the performance of replication schemes when read and write jobs engage arbitrary number of servers for their service. Moreover no one has compared the performance of quorum based schemes for different scheduling strategies in the same scenario. Our analysis of the weighted voting protocol in chapter 3 assumed that write jobs have higher priority than read jobs. This chapter presents a model for quorum based protocols where jobs are served in FIFO order. We extend the model presented by Nelson and Iyer in [58] to study the performance of quorum based schemes with arbitrary read and write quorums. We then compare the results of both FIFO and priority scheduling strategy. In each case the performance measures of interest are the response times of read and write jobs. The comparison of results for the two cases show that in many situations we can improve the performance of the system by assigning priorities to different type of jobs. We explain the reason behind this with the help of an example in section 4.4. A discussion of when to use which scheduling strategy concludes the chapter.

## 4.2 FIFO scheduling

We present two models to evaluate the performance of quorum based protocols when read and write jobs are being served using FIFO scheduling strategy. Our models are extensions of the models given by Nelson and Iyer for evaluating the performance of replication with read one write all policy. We first discuss the modifications required in the models presented by Nelson and Iyer [58] to deal with read and write quorums other than 1 and $N$ ($N$ is the total number of servers in the system). We then present the analysis and use the Spectral Expansion method to solve our models.

### 4.2.1 Optimistic Scheduling

There are $N$ identical servers. We assume that these servers do not fail. Two type of jobs, read and write, arrive in the system as independent Poisson streams. This can be modelled by assuming that all jobs arrive in a single Poisson stream with rate $\lambda$, and any job is a read with probability $r$. Both read and write jobs join the same queue and are served in FIFO order. A read job requires $R$ servers (read quorum) for its service whereas a write job requires $W$ servers (write quorum) for its service. We assume that $R+W = N+1$. There is no preemption and the service of a read (write) job can be started only if a quorum is available. In optimistic scheduling strategy a read or a write operation releases each copy as soon as its service on that copy is complete. An operation (read or write) at the head of the queue can use this copy to collect a quorum. This is same as the non-synchronous case in [58]. The condition $R + W = N + 1$ ensures that if read and write are executing in parallel at least one copy in the read quorum is written by the current write. This is needed for the read to always read the latest version of the data. A new write may also start before the completion of the current write if a write quorum becomes available as current write releases copies. Therefore we further assume that $W > N/2$ and therefore $R < W$. This ensures that even if two writes are executing in parallel the one that started service later always modifies a copy which is the latest version . This is required to maintain consistency. In optimistic scheduling at least $R$ servers are always busy if there is a job in the queue. We assume that the time to complete a read or a write service on each server is exponentially distributed with mean $1/\mu$. As all replicas in quorum are accessed in parallel the time to complete a read(write) service will be the maximum of $R(W)$ exponentially distributed random variables. The model is illustrated in Figure 4.1.

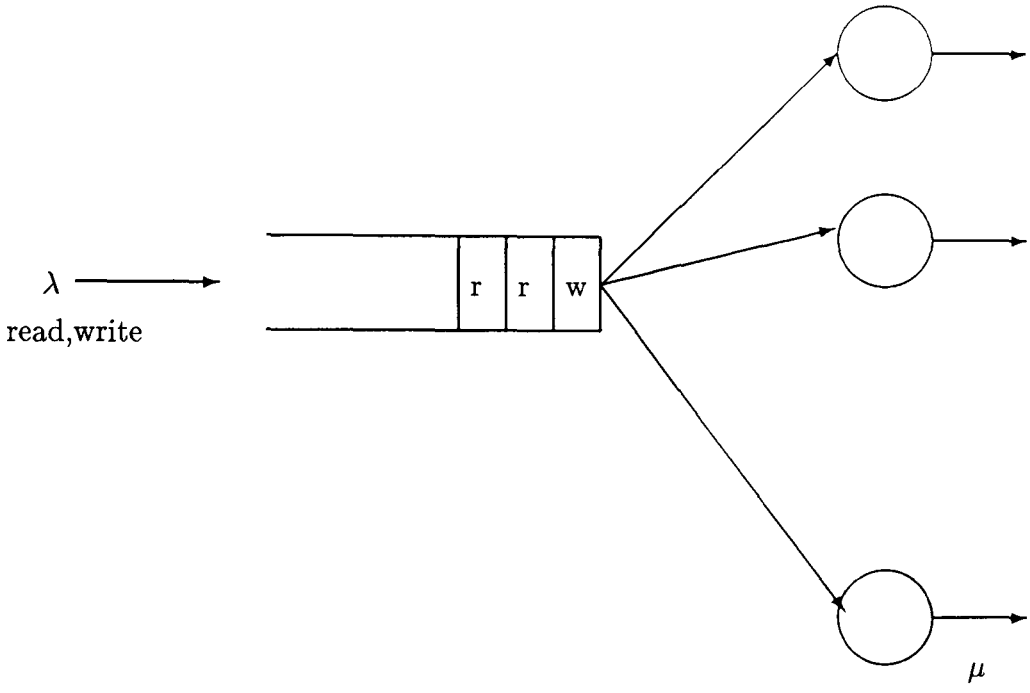



Figure 4.1: FIFO Scheduling

Let $I(t)$ and $J(t)$ be the number of busy servers and number of jobs in the queue at time $t$ excluding the jobs that are in service. Then the pair $[I(t), J(t)]$ is an irreducible Markov process on the state space $\{0, 1, \ldots, N\} \times \{0, 1, \ldots\}$. We draw the state transition diagram for this process in Fig. 4.2, for $N = 5$, $W = 4$ and $R = 2$.

The object of our analysis is to determine the response time of read and write jobs. The transitions possible in the Markov process shown in Figure 4.2 are:

(a) From state $(i, j)$ to state $(k, j)$ where $0 \leq i, k \leq N$ if $j = 0$; $R \leq i, k \leq N$ otherwise.

(b) From state $(i, j)$ to state $(i, j + 1)$ when $R \leq i \leq N$ and a read or a write job arrives.

(c) From state $(i, j)$ to state $(N, j-1)$ when $i = W, R$. The transition when $i = W$ takes place if the job at the head of the queue is a read because a read quorum is now

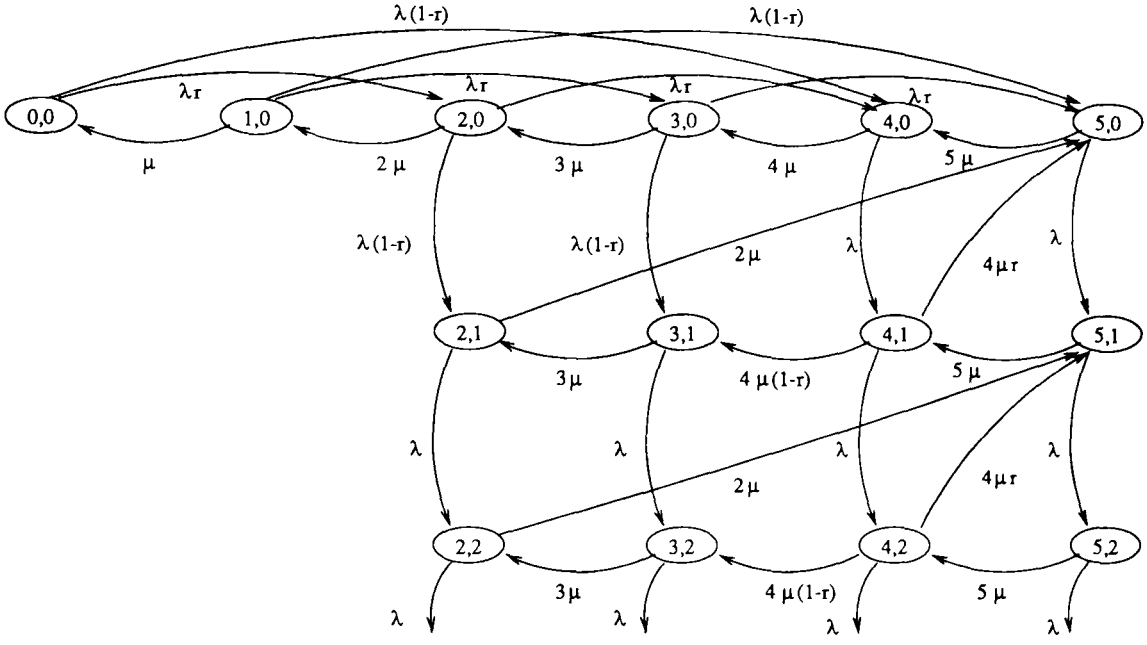



Figure 4.2: State Transition Diagram for Optimistic FIFO Scheduling

quorum sizes: $R = 2, W = 4$

available. The transition when $i = R$ takes place because the job at the head of the queue is a write and a write quorum is available now.

For $j = 0$ transitions of type (c) are not possible. Let us denote the transition rate matrices associated with (a) and (b) when $j = 0$ by $A_0$ and $B_0$ respectively. These matrices are of size $N \times N$. The elements of these matrices are given as

$$a_0(i,k) = \begin{cases} (1-r)\lambda & ;\ i < R,\ k = i + W \\ r\lambda & ;\ i < W,\ k = i + R \\ i\mu & ;\ i > 0,\ k = i - 1 \end{cases}$$

$$b_0(i,i) = \begin{cases} (1-r)\lambda & ;\ R \leq i < W \\ \lambda & ;\ i \geq W \\ 0 & otherwise \end{cases},$$

For $j \geq 1$ these transitions do not depend on $j$. Let us denote the transitions rate matrices associated with (a), (b) and (c) by $A, B$ and $C$ for $j \geq 1$. These matrices are of size $W \times W$ because the state variable $I$ can only take values $R, R+1, \ldots, N$. The matrix $B$ is given by:

$$B = \lambda I \quad \text{for} \quad j \geq 1 \tag{4.1}$$

where $I$ is the identity matrix of order $W$. The elements of matrices $A$ and $C$ equal to:

$$a(i,i-1) = \begin{cases} i\mu(1-r) & ;\ i = W \\ i\mu & ;\ i > R,\ \ i \neq W \end{cases},$$

$$c(i,N) = \begin{cases} i\mu r & ;\ i = W \\ i\mu & ;\ i = R \\ 0 & \text{otherwise} \end{cases},$$

Let $p_{i,j}$ denote the steady state probability that there are $i$ busy servers and $j$ jobs are in the queue:

$$p_{i,j} = \lim_{t\to\infty} P[I(t) = i, J(t) = j] \ ; \ i = 0, 1, \ldots, N \ ; \ j = 0, 1, \ldots \ .$$

We define the row vectors $\mathbf{v}_j$ of probabilities with $j$ jobs in queue as:

$$\mathbf{v}_0 = (p_{0,0}, p_{1,0}, \ldots, p_{N,0})$$

and

$$\mathbf{v}_j = (p_{R,j}, p_{R+1,j}, \ldots, p_{N,j}) \ ; \ j = 1, 2, \ldots \ .$$

This is because if queue is nonempty at least $R$ servers are always busy and the system never enters into states $(0, j)$ to $(R-1, j)$ for $j > 0$. The probabilities in vectors $\mathbf{v}_j$ for $j = 1, 2, \ldots$ can be determined using the balance equation

$$\mathbf{v}_j[\lambda I + D^A + D^C] = \mathbf{v}_{j-1}\lambda I + \mathbf{v}_j A + \mathbf{v}_{j+1}C \ ; \ j = 2, 3, \ldots \ , \tag{4.2}$$

where $I$ is the identity matrix, $D^A$ and $D^C$ are the diagonal matrices of size $W \times W$ whose $i$'th diagonal elements are the $i$'th row-sum of $A$ and $C$ respectively.

We use Spectral Expansion method described in section 3.3.2. to get an expression for $\mathbf{v}_1$ and $\mathbf{v}_2$ in terms of unknown $x$, eigenvalues $z$ and row eigenvectors $\psi$.

$$\mathbf{v}_j = \sum_{\ell=0}^{W-1} x_\ell \psi_\ell z_\ell^j \ , \ \ j \geq 1 \tag{4.3}$$

There are $W$ unknown coefficients. For determining $\mathbf{v}_0$ we use the following balance equation to express $\mathbf{v}_0$ in terms of $\mathbf{v}_1$:

$$\mathbf{v}_0[D_0^A + D_0^B - A_0] = \mathbf{v}_1' C' \ ; \ \ , \tag{4.4}$$

$D_0^A$ and $D_0^B$ are the diagonal matrices of size $N \times N$ whose $i$'th diagonal elements are the $i$'th row-sum of $A_0$ and $B_0$ respectively. Elements of vector $\mathbf{v}_1'$ and matrix $C'$ are given by:

$$\mathbf{v}_1'(i) = \begin{cases} \mathbf{v}_1(i) & ; \ i \geq R \\ 0 & \text{otherwise} \end{cases} ,$$

$$c'(i,k) = \begin{cases} c(i,k) & ; \ i \geq R, j \geq R \\ 0 & \text{otherwise} \end{cases} ,$$

The unknown coefficients of the spectral expansion, $x_\ell$ can now be determined by using following equation along with the normalizing equation

$$\mathbf{v}_1[\lambda I + D^A + D^C - A] = \mathbf{v}_0' B_0' + \mathbf{v}_2 C \ ; \ , \tag{4.5}$$

and vector $\mathbf{v}_0'$ and matrix $B_0'$ are of size $W$ and $W \times W$ respectively. Their elements are given by:

$$\mathbf{v}_0'(i) = \begin{cases} \mathbf{v}_0(i+R) & ; \ i = 0, 1, \ldots W-1 \end{cases} ,$$

$$B_0'(i,k) = \begin{cases} B_0(i+R, k+R) & ; \ i,k = 0, 1, \ldots W-1 \end{cases} ,$$

### 4.2.2 Pessimistic Scheduling

In Pessimistic scheduling strategy we assume that servers engaged by a read or write service are only available at the completion of the service. This is same as the synchronous case described in [58]. This decreases the utilization of servers. As earlier we assume that there are $N$ servers that can not fail. Read and write jobs arrive in the system as Poisson streams. Arrival rate of jobs is $\lambda$ and the probability that the arrival is a read is $r$. A read (write) service require a quorum of $R(W)$ replicas for its

service. Both read and write share the same queue and get the service in first come first serve basis. There is no priority associated with any type of job. A read or write service may start when it reaches at the head of the queue and a quorum to start the service is available. This ensures that read and write or two write services can not execute in parallel. At any time either one write service or a maximum of $\lfloor N/R \rfloor$ read services can be in progress. There is no preemption. We assume that the time to complete a write or a read service is exponentially distributed with mean $1/\nu_1$ and $1/\nu_2$ respectively. If all replicas in quorum are accessed in parallel we assume that this time is the maximum of R(W) exponentially distributed random variables and is given by (3.1) for some fixed $\mu_1$ and $\mu_2$. If operations are performed sequentially on all replicas this time is given by (3.2).

Let $I(t)$ represent the system state and $J(t)$ be the number of jobs in the system at time $t$, including the jobs that are in service. The system can be in one of the following states: idle (no job in service), a write service in progress or a maximum of $\lfloor N/R \rfloor$) read jobs in service. At any time $t$, $I(t)$ can take values between 0 and $\min(J(t), \lfloor N/R \rfloor)$. If $I(t)$ is 0 and $J(t) > 0$ a write service is in progress, otherwise $I(t)$ read services are in progress. The system is idle when $I(t) = J(t) = 0$. The pair $[I(t), J(t)]$ is an irreducible Markov process on the state space $\{0, 1, \ldots, \lfloor N/R \rfloor\} \times \{0, 1, \ldots\}$. We draw the state transition diagram for this process in Figure 4.3 for $\lfloor N/R \rfloor = 3$.

It is clear from the diagram that only possible transitions are

(a) from state $(i, j)$ to state $(k, j+1)$ and $i, k \leq min(j, \lfloor N/R \rfloor)$ when a job arrives

(b) from state $(i, j)$ to state $(k, j-1)$ and $i, k \leq min(j, \lfloor N/R \rfloor)$ when a service completes

The transition rate matrices associated with (a) and (b) for $j > \lfloor N/R \rfloor$ are denoted

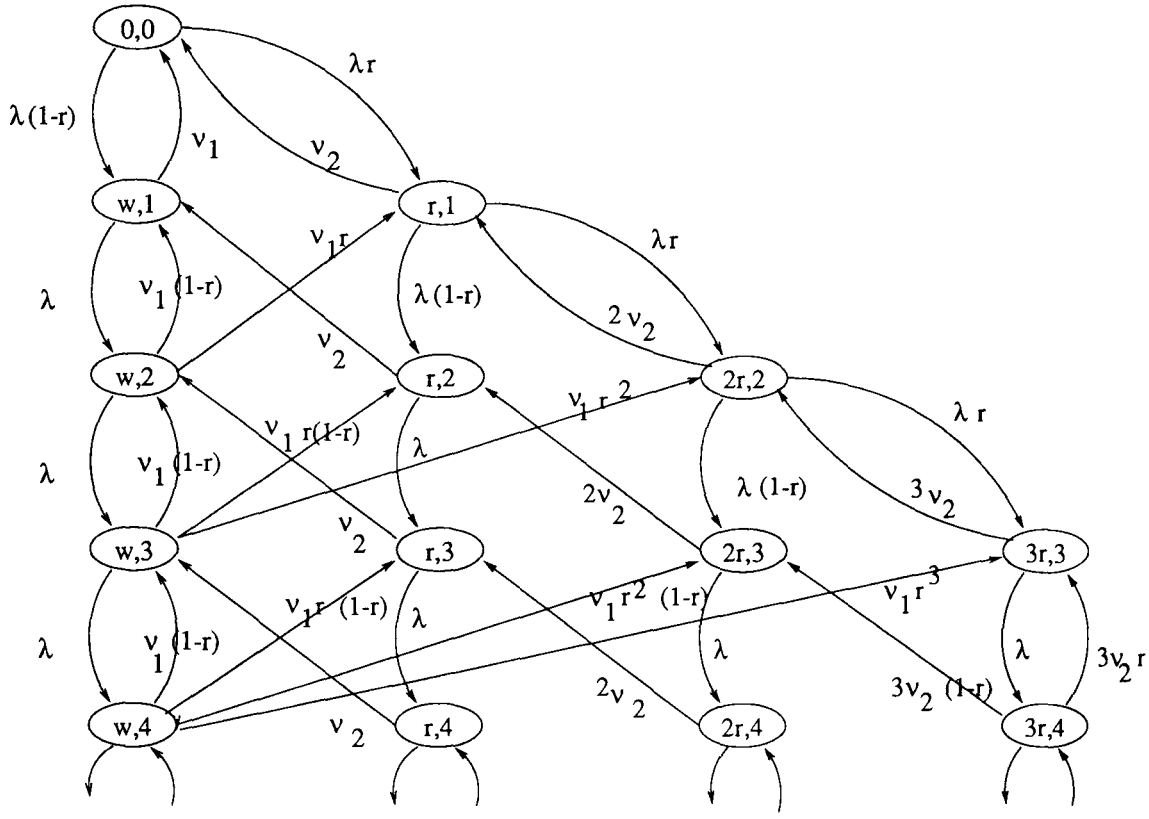



Figure 4.3: State Transition Diagram for pessimistic FIFO scheduling

by $B$ and $C$ and are $j$ independent. We define $M = \lfloor N/R \rfloor + 1$. The elements of the matrices $B$ and $C$ are given by

$$b(i,i) = \lambda$$

$$c(i,k) = \begin{cases} \nu_1 r^k(1-r) & ; \ i=0, \ 0 \le k \le M-2 \\ \nu_1 r^k & ; \ i=0, \ k=M-1 \\ i\nu_2 & ; \ i<(M-1), \ k=i-1, \ ; \ i \neq 0 \\ i\nu_2(1-r) & ; \ i=(M-1), \ k=M-2 \\ i\nu_2 r & ; \ i=(M-1), \ k=(M-1) \end{cases} ,$$

Let $p_{i,j}$ denote the steady state probability when system is in state $i$ and the number of jobs in the system are $j$:

$$p_{i,j} = \lim_{t \to \infty} P[I(t)=i, J(t)=j] \ ; \ i = 0,1,\ldots,min\{j, M-1\} \ ; \ j = 0,1,\ldots .$$

As before we can define vectors $\mathbf{v}_j$ for $j \ge M$ as:

$$\mathbf{v}_j = (p_{0,j}, p_{1,j}, \cdots, p_{M-1,j})$$

The balance equations for $j \ge M$ are given as:

$$\mathbf{v}_j[\lambda I + D^C] = \mathbf{v}_{j-1}\lambda I + \mathbf{v}_{j+1}C \quad , \tag{4.6}$$

where $D^C$ is the diagonal matrix whose $i$'th diagonal elements are the $i$'th row-sum of matrix $C$. Solution for $\mathbf{v}_{M-1}$ and $\mathbf{v}_M$ can be obtained using Spectral Expansion Method.

For $j < M$, $i$ takes values between 0 and $j$ only. The states $(i,j)$ for $i > j$ do not exist. We first obtain the expressions for vectors $\mathbf{v}_{M-1}, \mathbf{v}_M, \ldots$ in terms of unknowns $x_\ell$, eigenvalues $z_\ell$ and row eigenvectors $\psi_\ell$ given by

$$\mathbf{v}_j = \sum_{\ell=0}^{M-1} x_\ell \psi_\ell \mathbf{z}_\ell^j \,, \tag{4.7}$$

$M$ unknowns $x_\ell$ and $M(M-1)/2$ unknown probabilities can be evaluated by solving a set of $M(M+1)/2$ balance equations along with the normalizing equation.

## 4.3 Results of the numerical experiments

In this section we present the results of our investigations into the effect of changing the quorum sizes, $W$ and $R$ with $W + R = N + 1$ fixed, on the response time of read and write jobs. We compare these results with the results that we obtained when write jobs have preemptive priority over read jobs. Nelson and Iyer studied in [58] the effect of changing level of replication on the performance of read and write jobs when read quorum is one and write quorum is all. They have shown that for a given set of input parameters there is an optimal degree of replication for which the response time of read and write jobs is minimum. We observed the same effect in chapter 3 when write jobs have priority over read jobs. For the set of input parameters used in Figure 4.4 this optimal degree of replication is same for both pessimistic FIFO and priority scheduling. For optimistic FIFO scheduling the optimal degree of replication is larger. With write jobs having priority write service time will always be less. But for the same input parameters the response time of low priority read jobs is also less with priority scheduling (for $N \geq 2$) than pessimistic FIFO scheduling. This is due to the better utilization of the capacity of servers.

Figure 4.5 shows the read response time for pessimistic FIFO scheduling when read and write quorum changes. It can be seen that changing read and write quorums has almost same effect as in the case of priority scheduling. When write arrival rate is much less in comparison to read arrival rate, read response time continuously

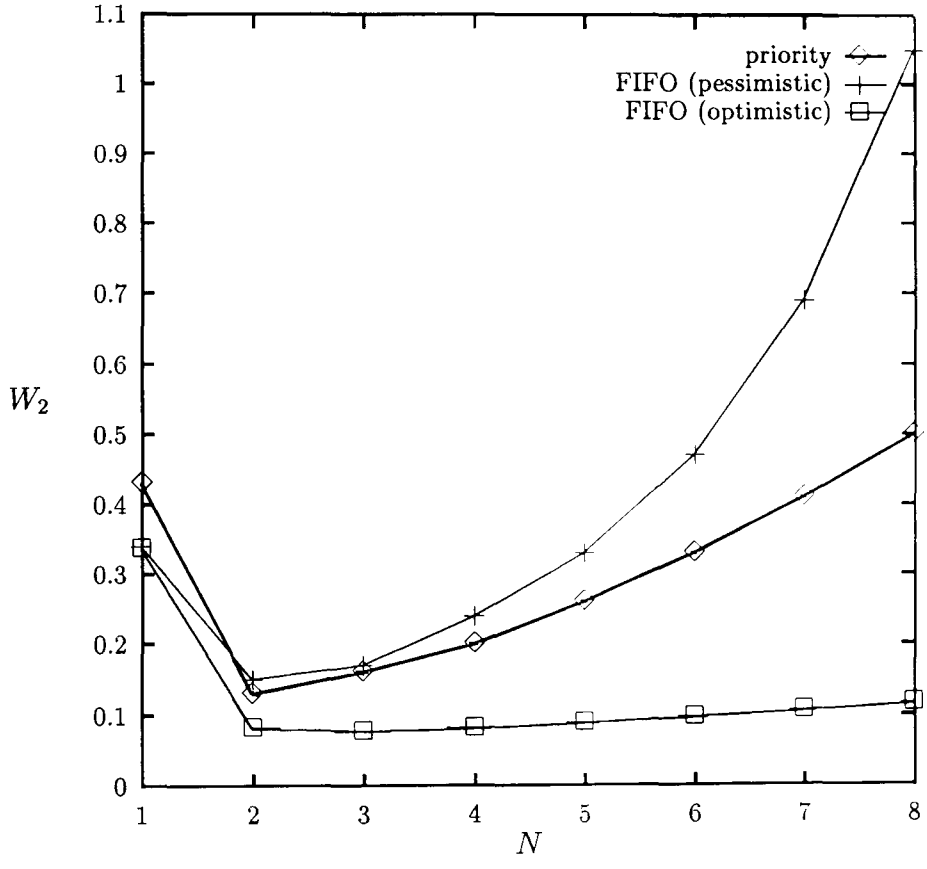



Figure 4.4: Mean Response Time as a function of Number of servers; $W = N, R = 1$

$\lambda = 30, r = .75, \mu_1 = \mu_2 = 33$

decreases with decrease in read quorum size. This is because as read quorum decreases more reads can execute in parallel. When read and write arrival rates are almost same, on decreasing $R$ (increasing $W$) response time of read jobs first increases but as soon as more than one read jobs can execute in parallel response time decreases. If write arrival rate is much larger in comparison to read, read response time increases with increase in write quorum size. With increase in write quorum, read quorum decreases and more reads can execute in parallel but at the same time write service time increases. When write arrival rate is larger the increase in write service time dominates and read response time increases.

Figure 4.6 shows the read response time for optimistic FIFO scheduling when read and write quorum changes. When read arrival rate is high, read response time first decreases due to the parallel execution of read jobs. But then the effect of increase in write service time dominates and read response time increases. When read and write arrival rates are same or when write arrival rate is high, read response time continuously increases with increase in write quorum due to the increase in write service time.

Figure 4.7 compares pessimistic FIFO scheduling and priority scheduling strategies with different quorum sizes. In the first set of curves read response time for FIFO scheduling is less than the read response time for priority scheduling. This is due to the fact that arrival rates of read and write jobs are very small in comparison to their service rates. In the second set of curves, where arrival rate of jobs are almost same as their service rates, read response time is some times less for priority scheduling and some times for FIFO scheduling.

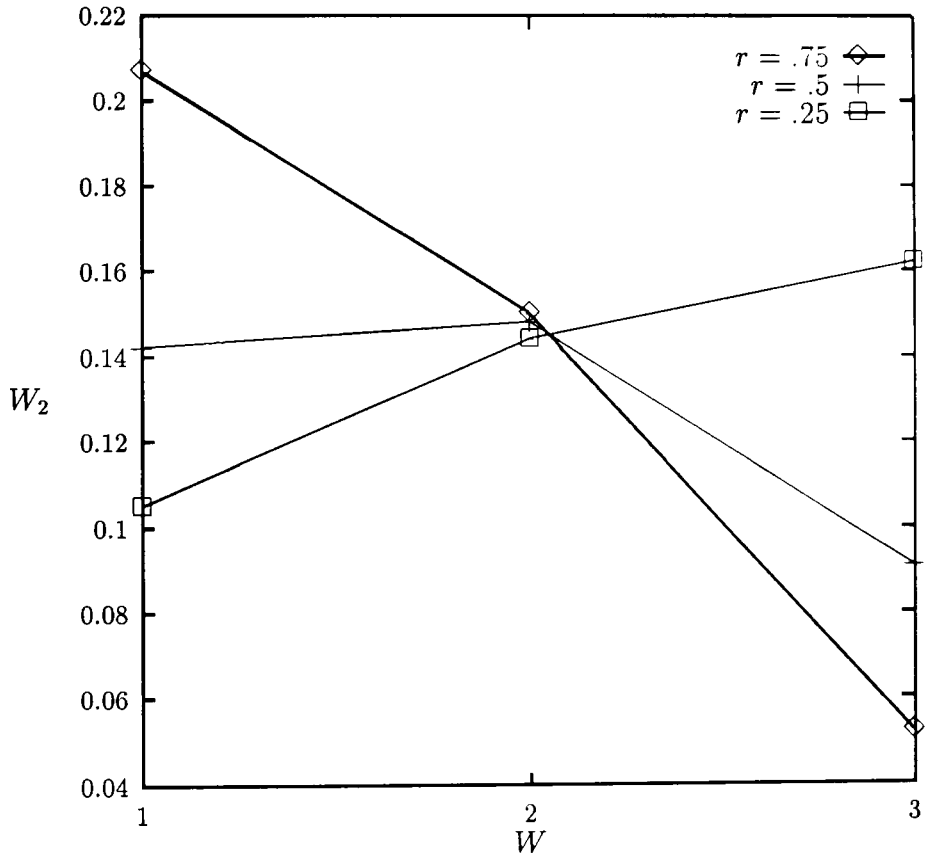



Figure 4.5: Mean Response Time with different quorum sizes

pessimistic FIFO scheduling

$N = 3, \lambda = 15, \mu_1 = \mu_2 = 33$

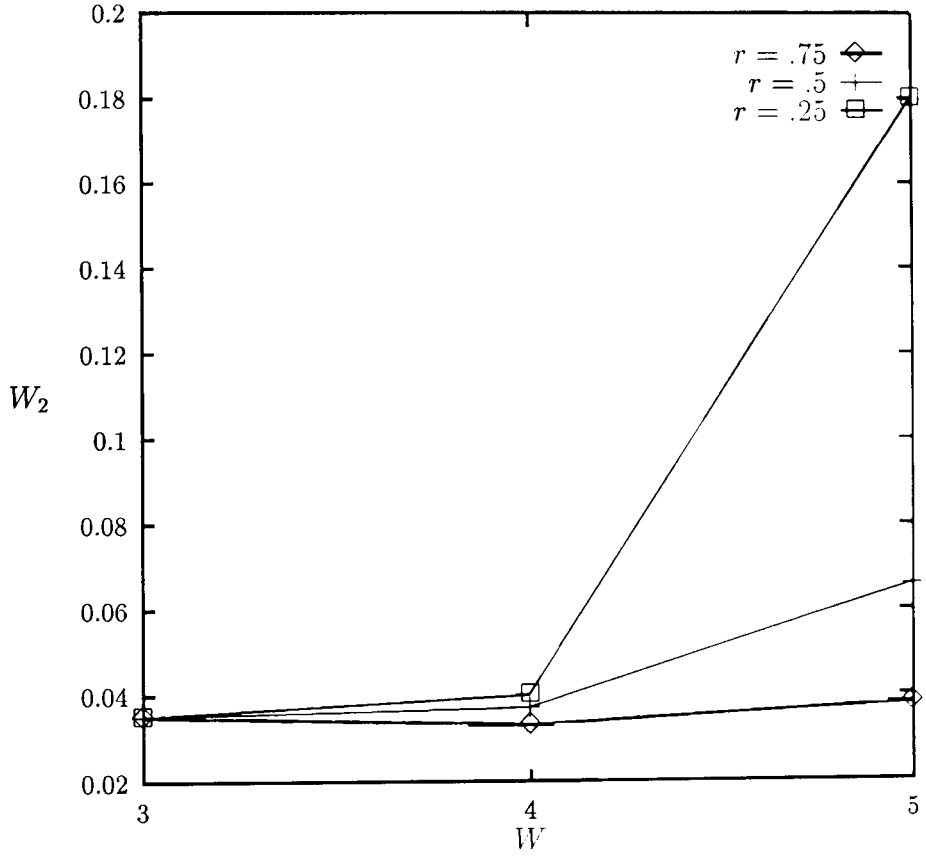



Figure 4.6: Mean Response Time with different quorum sizes

optimistic FIFO scheduling

$N = 5, \lambda = 15, \mu_1 = \mu_2 = 33$

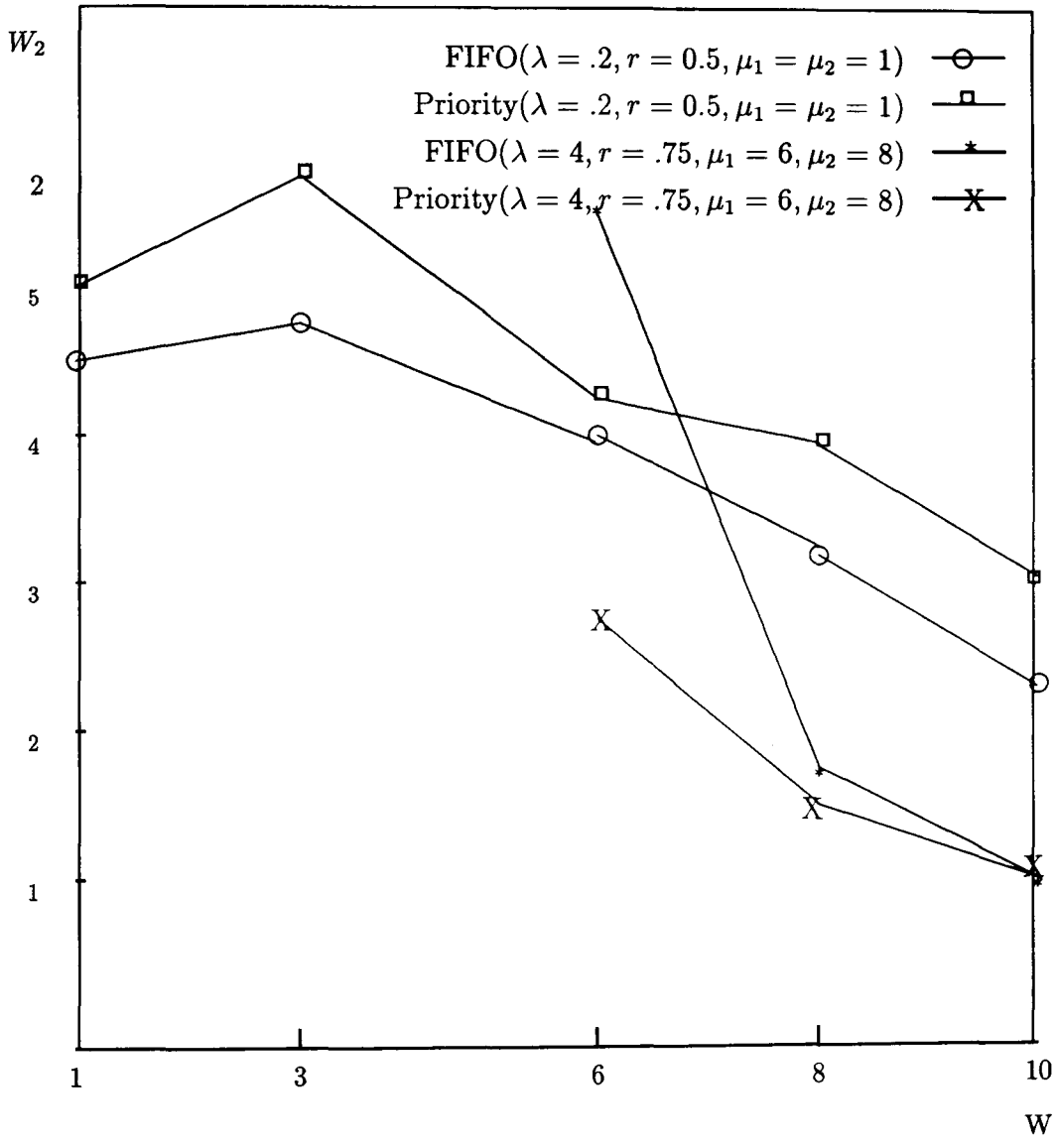



Figure 4.7: Priority vs pessimistic FIFO scheduling with different quorum sizes

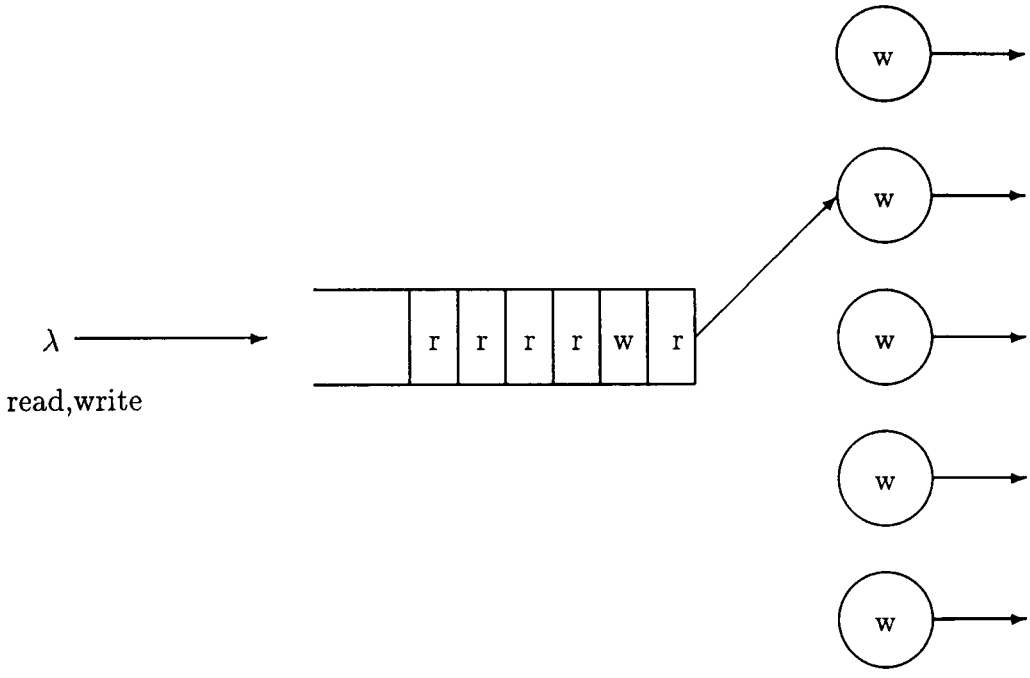
$N = 10$

## 4.4 Comparison with priority scheduling

The results of our numerical experiments show that in some cases performance can be improved by assigning higher priority to the jobs with low arrival rate in comparison to other jobs. We explain this with the help of following example. Let us consider a case with $N = 5$, $W = 5$ and $R = 1$ at time $t$. There are two write jobs (one executing and other waiting in the queue) and four read jobs (all waiting in queue) in the systems. The Figure 4.8 shows the relative position of read and write jobs in the queue when both type of jobs use the same queue and get the service in FIFO order. Let us further assume that a write service takes 2 seconds on an average and a read service takes an average of 1 second. Based on these averages the response time of read jobs for the situation shown in Figure 4.8 can be given as $(3 + 4 * 6)/5 = 5.4$ seconds. This is because after the completion of the service of current write only one read will get the service. This will be followed by the write service after which all the four reads will get the service in parallel. Now let us consider the case where read and write jobs join different queues and write jobs have priority over read jobs. In this case first both write will get the service and then all the five read jobs will get the service in parallel. This will reduce the overall read response time which in this case will be 5. It can be seen that the server utilization is more in second case.

## 4.5 Conclusion

This chapter is intended to demonstrate that assigning priorities to jobs may lead to better server utilization and so may improve the performance of the system. We do this by first presenting two models where jobs get service in first come first served basis. We then compare the results obtained with the results of chapter 3. The write




N=5, W=5, R=1

Figure 4.8: Example illustrating the benefit of priority scheduling

service time will always be less when write jobs have priority over read jobs. But the comparison shows that in many cases by assigning priorities even the response time of lower priority jobs can be reduced. We explained the reason behind this phenomenon with the help of an example in section 4.4.

# Chapter 5

# An exact solution for the system with a single server and breakdowns

## 5.1 Introduction

Chapter 3 gave a solution for the model with $N$ servers subject to random breakdowns and repairs, two types of jobs and a finite queue for type 1 jobs (it cannot exceed size $S$). We claimed in chapter 3 that if the ratio of arrival rate / service rate for type 1 jobs is small enough, the response time of type 2 jobs calculated for this model is very close to the response time for the model with both queues unbounded. There are many studies dealing with the performance of single server and multi server models with breakdowns. However either these studies consider only one job type or they provide an approximate solution. H.C.White and L.S. Christie were the first to consider server repair following breakdowns, or server vacation, in a queueing

system. In [71] they presented a M/M/1 queue model with random breakdowns and repairs. Later B. Avi-Itzhak and P. Naor considered 5 similar single server models in [10] where the service station is subject to breakdowns all with only one job type. I.L.Mitrany and B.Avi-Itzhak in [52] presented the analysis of a many-server queue with service interruptions and one job type. In [50] I. Mitrani and R. Chakka presented Spectral Expansion Solution for a Class of Markov Models whose state space is a lattice strip. B. Sengupta in [63] and K. Thiruvengadam in [66] also presented queueing systems with breakdowns and one job type. In [51] I.Mitrani and P.J.B. King analyzed Multiprocessor Systems with Preemptive Priorities and $N$ job types. They gave an exact solution for 2 job types and suggested how their method can be extended to get the approximate solution for $N$ job types. In this chapter we present an exact analysis for single server case with breakdowns and repairs and two job types, type 1 jobs having preemptive priority over type 2 jobs. Section 5.2 describes the model. In section 5.3 we give an analysis to find out the response time of type 2 jobs. As type 1 jobs have priority over type 2 jobs their response time can directly be calculated from the analysis presented in [52]. Section 5.4 presents results of our numerical experiments.

## 5.2 The Model

There is a single server whose operative periods are distributed exponentially with mean $1/\xi$. At the end of an operative period server breaks down and requires an exponentially distributed repair time with mean $1/\eta$. Two type of jobs arrive into the system. The arrival rate for type 1 and type 2 jobs are Poisson with rates $\lambda_1$ and $\lambda_2$, respectively. Their service times are also distributed exponentially with mean $\mu_1$ and $\mu_2$, respectively. Type 1 jobs have preemptive priority over type 2 jobs. This

model is illustrated in Fig. 5.1

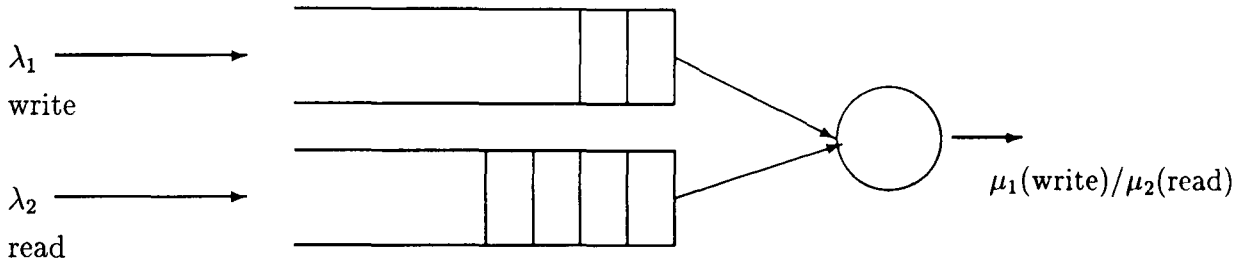



Figure 5.1: Single server with two type of jobs

Let $I(t)$ and $J(t)$ be the number of type 1 and type 2 jobs in the system at time $t$. Let $K(t)$ be the state of the server at time $t$ defined as

$$K(t) = \begin{cases} 0 & \text{if server is inoperative at time } t \\ 1 & \text{if server is operative at time } t \end{cases}$$

Triplet $[I(t),J(t),K(t)]$ is an irreducible Markov process on the state space $\{0,1,\ldots\} \times \{0,1,\ldots\} \times \{0,1\}$. As the server is operative only for $\eta/(\eta+\xi)$ fraction of total time, the ergodicity condition is

$$\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2} < \frac{\eta}{\eta+\xi} . \tag{5.1}$$

We assume that this condition holds.

## 5.3 Analysis

The object of the analysis is to determine the joint steady-state distribution of $I$ and $J$ both in case of operative and inoperative server, denoted by $p_1(i,j)$ and $p_0(i,j)$, respectively:

$$\begin{aligned} p_1(i,j) &= \lim_{t\to\infty} P[I(t)=i, J(t)=j, K(t)=1] \;;\; i,j=0,1,\ldots \\ p_0(i,j) &= \lim_{t\to\infty} P[I(t)=i, J(t)=j, K(t)=0] \;;\; i,j=0,1,\ldots \end{aligned}$$

These probability satisfy the following set of balance equations:

$$p_0(i,j)[\lambda_1+\lambda_2+\eta] = \lambda_1 p_0(i-1,j) + \lambda_2 p_0(i,j-1) + \xi p_1(i,j) \tag{5.2}$$

$$\begin{aligned} p_1(i,j)[\lambda_1+\lambda_2+\mu_1\delta(i>0)+\mu_2\delta(i=0,j>0)+\xi] &= \eta p_0(i,j) \\ +\mu_2\delta(i=0)p_1(0,j+1) &+ \mu_1 p_1(i+1,j) \\ +\lambda_1 p_1(i-1,j) &+ \lambda_2 p_1(i,j-1) \end{aligned} \tag{5.3}$$

where $p_0(-1,j)=0$ and $p_0(i,-1)=0$ by definition, and $\delta(A)$ is the indicator function: 1 if $A$ is true, 0 otherwise. Let us define generating functions for the two cases of operative and inoperative server as:

$$\begin{aligned} g_0(x,y) &= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} p_0(i,j)x^i y^j \;;, \\ g_1(x,y) &= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} p_1(i,j)x^i y^j \;, \end{aligned}$$

Multiplying both sides of equation (5.2) by $x^i y^i$ and summing over $i$ and $j$ we get the following equation:

$$g_0(x,y)[\lambda_1+\lambda_2+\eta] = \lambda_1 x g_0(x,y) + \lambda_2 y g_0(x,y) + \xi g_1(x,y)$$

which can be simplified to

$$g_0(x,y)[\lambda_1(1-x)+\lambda_2(1-y)+\eta] = \xi g_1(x,y) \tag{5.4}$$

In the same way multiplying both sides of equation (5.3) by $x^i y^i$ and summing over $i$ and $j$ yields following equation:

$$g_1(x,y)[\lambda_1+\lambda_2+\mu_1+\xi] - \mu_1 g_1(0,y) + \mu_2[g_1(0,y) - p_1(0,0)]$$

$$= \lambda_1 x g_1(x,y) + \lambda_2 y g_1(x,y) + \frac{\mu_1}{x}[g_1(x,y) - g_1(0,y)]$$

$$+\frac{\mu_2}{y}[g_1(0,y) - p_1(0,0)] + \eta g_0(x,y)$$

which can also be written as

$$g_1(x,y)[\lambda_1(1-x)+\lambda_2(1-y)+\mu_1(1-\frac{1}{x})+\xi] =$$

$$g_1(0,y)[\mu_1(1-\frac{1}{x}) - \mu_2(1-\frac{1}{y})]$$

$$+\mu_2(1-\frac{1}{y})p_1(0,0) + \eta g_0(x,y) \tag{5.5}$$

on substituting the value of $g_0(x,y)$ from equation (5.4) we get the following equation for $g_1(x,y)$ in terms of an unknown function $g_1(0,y)$ and an unknown probability $p_1(0,0)$

$$Q(x,y)g_1(x,y) = g_1(0,y)[\mu_1(1-\frac{1}{x}) - \mu_2(1-\frac{1}{y})] + \mu_2(1-\frac{1}{y})p_1(0,0) \tag{5.6}$$

where

$$Q(x,y) = \lambda_1(1-x) + \lambda_2(1-y) + \mu_1(1-\frac{1}{x})$$

$$+\xi - \frac{\xi\eta}{\lambda_1(1-x) + \lambda_2(1-y) + \eta} \tag{5.7}$$

The function $g_1(0,y)$ can now be determined by observing that for every $y \in (0,1)$, the quadratic equation

$$Q(x,y) = 0$$

has one real root, $x = f(y)$, satisfying $0 < f(y) < 1$, as $Q(1,y) > 0$ and $Q(0,y) < 0$. Since $g_1(f(y),y)$ is finite this gives us an expression for $g_1(0,y)$

$$g_1(0,y) = \frac{\mu_2 f(y)(1-y)p_1(0,0)}{\mu_1 y(f(y)-1) - \mu_2 f(y)(y-1)} \tag{5.8}$$

This leaves us with unknown probability $p_1(0,0)$ which can be determined by the following normalising condition.

$$g_0(1,1) + g_1(1,1) = 1 \tag{5.9}$$

The average response time for type 1 jobs, $W_1$, can be obtained directly from the results presented in [52] with parameters $\lambda_1$, $\mu_1$, $\xi$ and $\eta$. The average response time for type 2 jobs is given by

$$\lambda_2 W_2 = \frac{\partial}{\partial y}(g_0(1,1) + g_1(1,1)) \tag{5.10}$$

## 5.4 Results of Numerical experiments

We examine the effect of various parameters on the response time of type 2 jobs. In [52] Mitrani *et al* has already presented an exact analysis of the system with one or more servers subject to random breakdowns and repairs and one type of jobs. As in our case type 1 jobs have preemptive priority over type 2 jobs they will behave exactly in the same way as shown in [52]. This leaves us only with the performance of type 2 jobs to examine. To calculate the values of $g_0(0,1)$, $g_0'(0,1)$, $g_1'(1,1)$, one has to resolve indeterminancies of type 0/0. We use L'Hopital's rule once for generating functions and twice for their derivatives. The results of the experiments are displayed in Figure 5.2. The figure shows the effect of arrival rates, service rates, fault and repair rates on the average response time of type 2 jobs. As expected the average response time for type 2 jobs increases with decrease in service rate or increase in arrival rate for type 1 and type 2 jobs. Increase in the arrival rate of faults or decrease in the repair rate also increases the average response time for these jobs. Figure 5.3 compares exact average response time for type 2 jobs calculated from the analysis presented in this chapter and the approximate average response time obtained from the Spectral Expansion method with a bounded queue for type 2 (write) jobs. The curves show that when ratio arrival rate / service rate of type 1 jobs is very small the response time calculated from both the methods is almost same whereas if this ratio is large the response time calculated by the exact method is larger than the time calculated by the approximate method (of chapter 3). This confirms our claim made in chapter 3.

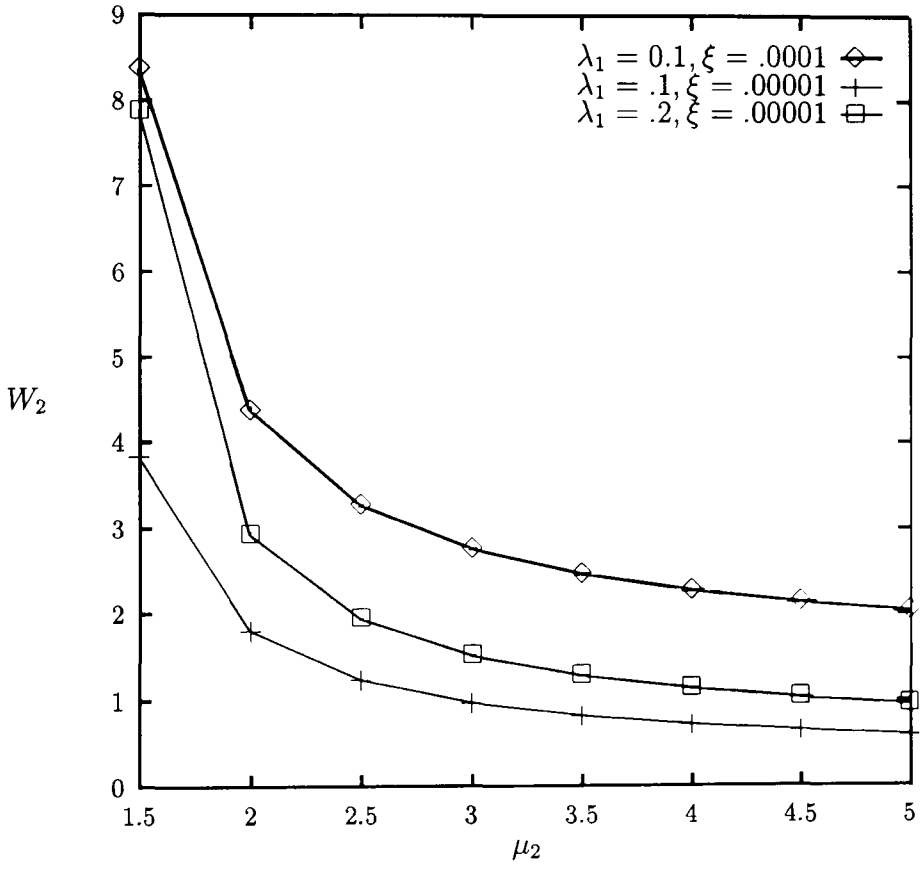



Figure 5.2: Response time as a function of service rate

$\lambda_2 = 1$, $\mu_1 = 1$, $\eta = .01$

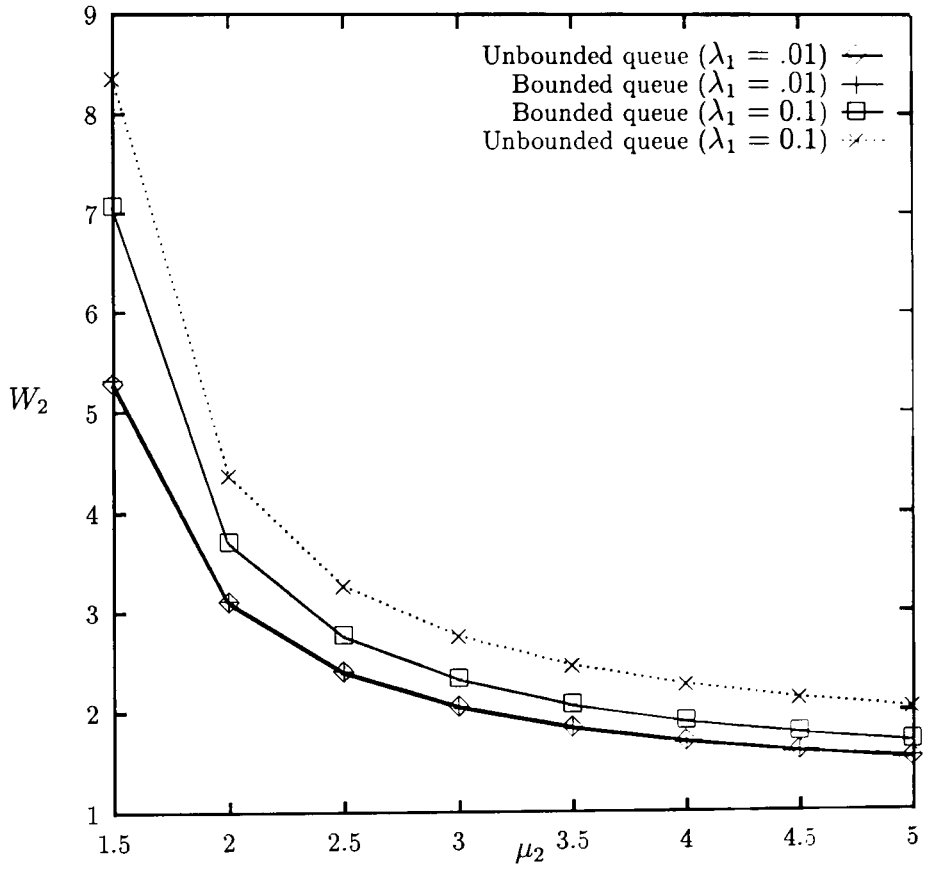



Figure 5.3: Bounded (S=1) vs Unbounded queue

$\lambda_2 = 1$, $\mu_1 = 1$, $\xi = .0001, \eta = .01$

## 5.5 Conclusion

This chapter considers the model of single server with breakdowns and two type of jobs, type 1 jobs having priority over type 2 jobs. It gives a method to calculate exact response time for type 2 jobs when arrival and service rates of jobs as well as arrival rate of faults and repair rates are Poisson. We also compare our results with the results obtained in chapter 3 for the same model with only difference that type 1 jobs have finite queue size. The comparison confirms our claim.

# Chapter 6

# Data Replication With Two Levels of Consistency

## 6.1 The Protocol

Chapter 3 and 7 present the analysis of strong and weak consistency protocols respectively. Strong consistency protocols guarantee that user will always get the most recent information. But with large number of replicas the performance of strong consistency protocols may not be acceptable as they require synchronization among at least a majority of these copies to ensure consistency. Weak consistency protocols, on other hand, try to improve performance by relaxing consistency constraints and so the data accessed by the user may not be the most recent one. Multilevel consistency protocols analyzed in this chapter try to integrate both approaches into the same framework by dividing replicas into groups or levels. Replicas at different levels provide different levels of consistency. In this chapter we present analysis of a two-level protocol where the replicas at level 0 provide strong consistency and replicas at level 1 may be out-of-date. We first consider the case of reliable replicas and then

present the analysis with breakdowns and repairs. The analysis in the case of latter is approximate as in chapter 3. Users reading from a level 1 replica may be interested in knowing the probability of getting most recent value of data. We give expressions to evaluate this probability. Finally we discuss some generalizations in the models.

## 6.2 Reliable Replicas

### 6.2.1 Model

In this section we describe our model for reliable replication. Three types of jobs, write, slow read and fast read, arrive into the system in independent Poisson streams with rates $\lambda_1$, $\lambda_2$ and $\lambda_3$ respectively. There are $N$ identical servers, each managing a copy of the data. Of these, $N_0$ are at level 0 and $N_1 = N - N_0$ are at level 1. There is a write and a slow read queue at level 0, with preemptive priority to writes, and a fast read and a write queue at each of the servers at level 1, with preemptive priority to fast reads. Replicas at level 0 maintain strong consistency with the help of quorums. An incoming write job joins the write queue at level 0, where its service requires the simultaneous possession of $Q_1(\leq N_0)$ servers (write quorum), for an exponentially distributed interval with mean $1/\nu_1$. After completing that service, an independent instance of the job is sent to each of the level 1 write queues, where service times are distributed exponentially with mean $1/\gamma_1$. A slow read service requires $Q_2 = N_0 + 1 - Q_1$ servers at level 0 (read quorum) and is exponentially distributed with mean $1/\nu_2$. Slow reads preempted by the arrival of a write job join the slow read queue again and get the service from the point of interruption. Fast read jobs join one of the fast read queues at level 1, with equal probability (this assumption could easily be relaxed). Their service times are distributed exponentially with mean

$1/\gamma_3$. The model is illustrated in Fig. 6.1.

One could also introduce 'pure delay' (infinite-server) nodes in order to model non-zero transfer times for jobs between remote user sites and the servers at levels 0 and/or 1. Such nodes would only add constants to the average response times. However, if write jobs in transit between level 0 and level 1 are similarly delayed, the probability that a level 1 replica is out-of-date is also affected (see section 6.2.2).

The strong consistency of replicas at level 0 is ensured by the quorum sizes and by the additional requirement that at most 1 write job can be in service there at any moment, regardless of the value of $Q_1$. The maximum number of slow read jobs that can be served in parallel is $r = \lfloor N_0/Q_2 \rfloor$, where $\lfloor x \rfloor$ is the integer part of $x$.

The service time parameters $\nu_1$ and $\nu_2$ depend, in general, on the quorum sizes. An access that engages a larger number of servers can be expected to take longer. Thus, the average write service times at level 0 usually increase with $Q_1$, and the slow read ones increase with $Q_2$. The nature of that increase depends on the way write and slow read operations are implemented.

If all replicas in a quorum are accessed in parallel, then service times $\nu_1$ and $\nu_2$ are given by equation (3.1) for some fixed $\mu_1$ and $\mu_2$, and are the averages of *the largest* of $Q_1$ (respectively $Q_2$) i.i.d. random variables, each distributed exponentially with mean $1/\mu_1$ (respectively $1/\mu_2$). These latter averages may include message-passing delays.

If, on the other hand, the operations are performed sequentially on all replicas, then these times are given by equation (3.2).

It is clear from the above description that the states of levels 0 and 1 are independent of each other. Indeed, since write jobs have preemptive priority at level 0 and are executed one at a time, they behave like the customers of an M/M/1 queue

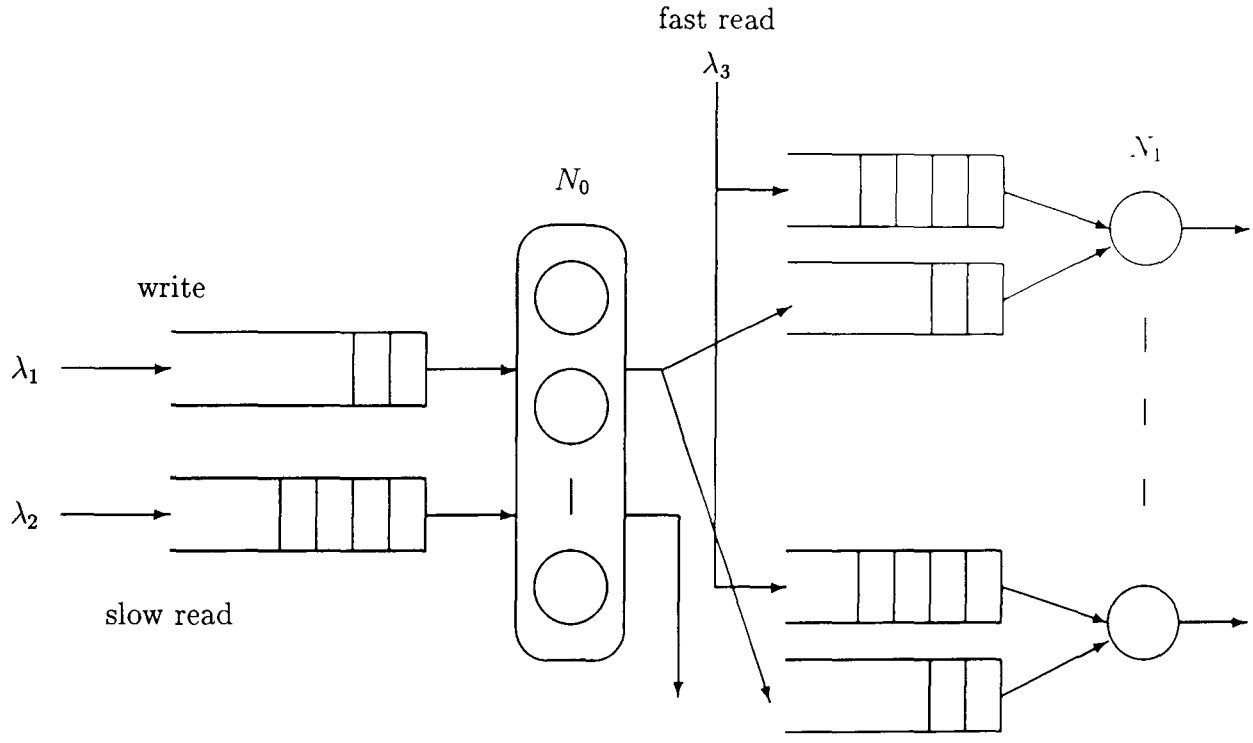



Figure 6.1: A two-level replication hierarchy

with parameters $\lambda_1$ and $\nu_1$. Therefore, the write departures from level 0 form a Poisson stream (with rate $\lambda_1$), whose past history is independent of the current state of the queue. Therefore, the current state of level 1 is independent of the number of write jobs at level 0. The different replicas at level 1 are of course dependent on each other, since their write arrival instants are correlated. However, for the purpose of our performance measures, the marginal state distribution of any given state 1 replica can be analyzed in isolation from the others.

Thus, we are justified in treating each level separately.

### 6.2.2 Analysis

*Level* 0

Let $I(t)$ and $J(t)$ be the numbers of write and slow read jobs present at level 0 at time $t$. Under the above assumptions, the pair $[I(t), J(t)]$ is an irreducible Markov process on the state space $\{0, 1, \ldots\} \times \{0, 1, \ldots\}$. Since the largest number of services in progress at any time can be either 1 write or $r$ slow read (but not both), the ergodicity condition for level 0 is

$$\frac{\lambda_1}{\nu_1} + \frac{\lambda_2}{r\nu_2} < 1 \, . \tag{6.1}$$

That condition will be assumed to hold. The first aim of the analysis is to determine the steady-state joint distribution of $I$ and $J$, denoted by $p_{i,j}$:

$$p_{i,j} = \lim_{t \to \infty} P[I(t) = i, J(t) = j] \;\; ; \;\; i, j = 0, 1, \ldots \, .$$

These probability satisfy the following set of balance equations:

$$p_{i,j}[\lambda_1 + \lambda_2 + \nu_1\delta(i > 0) + min(j, r)\nu_2\delta(i = 0)] = \lambda_1 p_{i-1,j} + \lambda_2 p_{i,j-1}$$

$$+\nu_1 p_{i+1,j} + min(j + 1, r)\nu_2\delta(i = 0)p_{i,j+1} \ ; \ i, j = 0, 1 \ldots , \qquad (6.2)$$

where $p_{-1,j} = 0$ and $p_{i,-1} = 0$ by definition, and $\delta(B)$ is the indicator function: 1 if $B$ is true, 0 otherwise.

The set of balance equations (6.2) are same as (3.4) and the analysis given in section 3.2.2 can be applied to get the response time of slow read jobs. It is given by equation (3.14).

*Level* 1

Since fast read jobs have preemptive priority at level 1, their average response time, $W_3$, can be obtained by treating a level 1 replica as an M/M/1 queue with arrival rate $\lambda_3/N_1$ and service rate $\gamma_3$:

$$W_3 = \frac{N_1}{N_1\gamma_3 - \lambda_3} \ . \qquad (6.3)$$

The other performance measure of interest is the steady-state probability, $U$, that an incoming fast read request gets a consistent version of the data. According to the PASTA property, $U$ is equal to the steady-state probability that a given replica at level 1 is consistent. In determining that quantity, we shall treat the general case where write jobs in transit between level 0 and level 1 are subjected to a random delay with mean $\tau$. For the purpose of this calculation, the slow read jobs at level 0 may be ignored, since they have low preemptive priority.

Consider a particular replica, $C$, at level 1, with its high priority fast read queue and low priority write queue, at some point in the steady-state. Note that if at that

moment there are write jobs at level 0, one of them is in service and therefore $C$'s data is being updated. Similarly, if there are write jobs in transit between level 0 and $C$, or present at $C$, then $C$ is inconsistent because there exist updates which have taken place at level 0 but not here. Conversely, if there are no write jobs at level 0, or in transit, or at $C$, then all preceding updates have been implemented and $C$ is consistent. Moreover, the numbers of write jobs at those three locations are independent of each other, as pointed out earlier (the departures from an M/G/$\infty$ delay node are also Poisson).

We can therefore write

$$U = q_1 q_2 q_3 \ , \tag{6.4}$$

where $q_1$ is the probability that there are no write jobs at level 0, $q_2$ is the probability that there are no write jobs in transit and $q_3$ is the probability that there are no write jobs at the level 1 replica.

Standard M/M/1 and M/G/$\infty$ results imply

$$q_1 = 1 - \frac{\lambda_1}{\nu_1} \quad , \quad q_2 = e^{-\lambda_1 \tau} \ . \tag{6.5}$$

Now, $q_3$ is the probability that there are no lower priority jobs in a single-server system with two priority types. Translating the notation of subsection 3.2.2 to this system, we have $q_3 = g(1,0)$. Using equation (3.6) and following the steps indicated after (3.9), with the level 1 parameters $g(1,0)$ can be written as

$$g(1,0) = \frac{\gamma_3 (1 - x_0)}{\lambda_1 x_0} p_{00} \tag{6.6}$$

where $x_0$ is the unique root in the interval (0,1) of the quadratic equation

$$\frac{\lambda_3}{N_1}x(1-x) + \lambda_1 x - \gamma_3(1-x) = 0\,.$$

and $p_{00}$ is given by

$$p_{00} = 1 - \frac{\lambda_3}{N_1\gamma_3} - \frac{\lambda_1}{\gamma_1}$$

The above expressions imply that the probability $U$ increases when $N_1$ increases (because the fast read arrival rate to each level 1 replica decreases); $U$ also increases when $N_0$ decreases (because the write service time at level 0 decreases). Thus, the 'most consistent' way of dividing $N$ replicas into two levels while ensuring that at least one is strongly consistent, is to chose $N_0 = 1$, $N_1 = N - 1$. However, that division is not necessarily optimal if one wishes to minimize the average response time for slow read jobs. Also, it may not be optimal if the average transfer delay, $\tau$, increases with $N_1$.

If the average response time of write jobs at level 1 is of interest, one can use either the analysis in section 3.2.2, or the known results for single-server queues with preemptive priorities.

### 6.2.3 Results of Numerical experiments

We first examine the effect of increasing the number of replicas at level 0, $N_0$, with total number of replicas $N$ fixed, and the quorum sizes are $Q_1 = N_0$, $Q_2 = 1$. All servers are fully reliable. The performance measures of interest are the average response times for slow read and fast read jobs, $W_2$ and $W_3$. The trade-off at level 0 is that when $N_0$ increases, the delays caused by write jobs increase, since $\mu_1$ is fixed and $\nu_1$ is given by (3.1); on the other hand, more slow read jobs can be executed in parallel. Fig. 6.2 shows that an initial lowering of $W_2$ can be achieved by increasing

$N_0$, especially when the slow read load is high. However, a point of no improvement is invariably reached (that point may be $N_0 = 1$), and eventually the write delays become dominant.

The increase in $W_3$ is explained by the fact that the total number of replicas , $N$, is constant; when $N_0$ increases, $N_1$ decreases and therefore the arrival rate of fast reads at each level 1 replica increases. The choice of $N_0$ and $N_1$ will depend on the performance required by the slow read and fast read jobs. Any values for $N_0$ and $N_1$ can be chosen between the lowest possible value of $N_0$ (for which the system at level 0 is stable) and the value after which no improvement in the slow read response time is possible. A choice of the former will give the lowest possible fast read response time. But the slow read response time will be a little higher than the lowest possible. Whereas a choice of latter will result in lowest possible slow read response time but a slightly higher fast read response time.

The effect of changing the quorum sizes, $Q_1$ and $Q_2$, with $Q_1 + Q_2 = N_0 + 1$ fixed (again all servers are fully reliable), is illustrated in Fig. 6.3. This time the behaviour is less predictable. Increasing $Q_1$ causes the slow read response time sometimes to increase, sometimes to decrease, and sometimes to increase and then decrease. The only reasonably general and intuitive observation that can be made is that when most of the offered load consists of read jobs, the allocation $Q_1 = N_0$, $Q_2 = 1$ is best, whereas $Q_1 = 1$, $Q_2 = N_0$ is best if most of the load consists of write jobs. It should be pointed out that if the performance measure is the overall average response time (including write jobs), rather than $W_2$, the situation is similar.

The fast read response time, $W_3$, is of course independent of the quorum sizes as long as $N_1$ does not change.

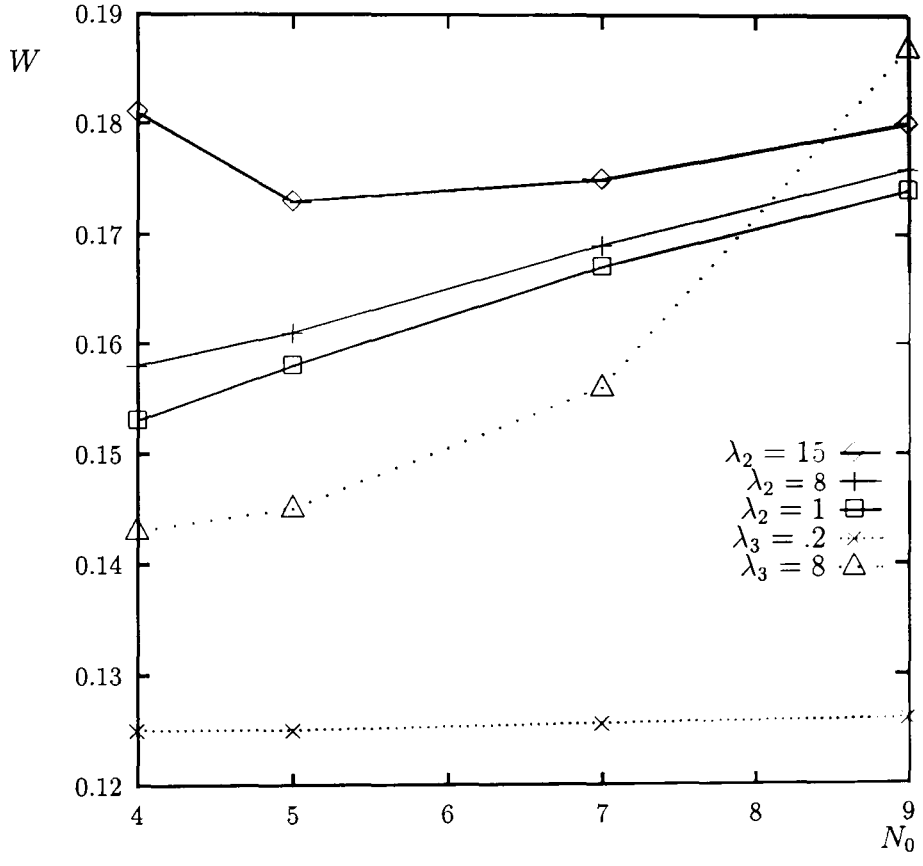



Figure 6.2: Dependence of $W_2$ (solid lines) and $W_3$ (dotted lines) on $N_0$.

$N = 12$, $\lambda_1 = 0.2$, $\mu_1 = \bar{\tau}$, $\mu_2 = \gamma_3 = 8$

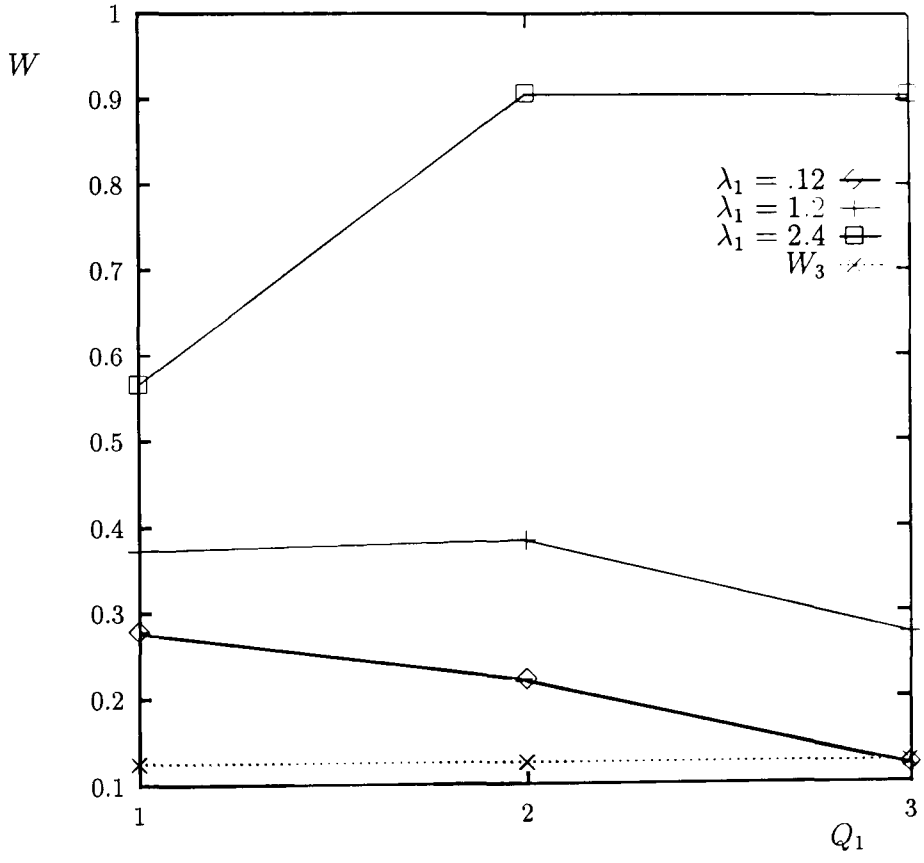



Figure 6.3: Effect of quorum sizes on $W_2$ (solid lines) and $W_3$ (dotted lines) .

$N_0 = 3$, $N_1 = 9$, $\lambda_2 = 1.2$, $\mu_1 = 8$, $\mu_2 = \gamma_3 = 9$

## 6.3 Unreliable Replicas

### 6.3.1 Model

This section describes the model when servers are not fully reliable. Suppose that each server goes through alternating periods of being operative and inoperative, independently of the others. Those periods are exponentially distributed with means $1/\xi$ and $1/\eta$, respectively. The other parameters and assumptions remain the same, except that a slow read service at level 0 can be in progress if either the write queue is empty, or a write quorum is not available. Thus, if there are both write and slow read jobs at level 0, and the number of operative servers, $m$, satisfies $Q_2 \leq m < Q_1$, then $\lfloor m/Q_2 \rfloor$ slow read jobs are served in parallel; if $m \geq Q_1$, one write job is served. The average write and slow read service times, $1/\nu_1$ and $1/\nu_2$, are given by (3.1) or (3.2) depending on whether the replicas in quorum are accessed in parallel or sequentially. Services interrupted by either breakdowns or preemptions are eventually resumed from the point of interruption.

As mentioned in section 3.3.2 the exact analysis for level 0 with both queues unbounded is, at present, intractable. We shall provide an approximate solution by assuming that queue 1 cannot exceed size $S$. Write jobs arriving when there are already $S$ of them in the system are lost. The accuracy of this approximation clearly increases with $S$, but so does its numerical complexity. However, it is possible to obtain accurate results with small values of $S$ when the offered load due to the write jobs, $\lambda_1/\nu_1$, is small compared to the processing capacity available to them. The latter is equal to the probability that there are at least $Q_1$ operative servers:

$$c_1 = \sum_{j=Q_1}^{N_0} \binom{N_0}{j} \frac{\eta^j \xi^{N_0-j}}{(\xi+\eta)^{N_0}} . \tag{6.7}$$

At level 1 if a server is down the read and write jobs in the queues associated with this server wait for the server to become operative again and do not get service from one of the operative servers.

### 6.3.2 Analysis

*level 0*

The analysis for level 0 is same as described in section 3.3.2 with slow reads treated as read jobs. At any time $t$ the state at level 0 is described by three integers, $K(t)$, $I(t)$ and $J(t)$, denoting the numbers of operative servers, write jobs present and slow read jobs present, respectively. The first two of these have finite ranges and we replace them by a single integer, $Y(t) = (N_0 + 1)I(t) + K(t)$, which takes values $0, 1, \ldots, M$, where $M = N_0 S + N_0 + S$. When $Y(t) < N_0 + 1$, there are $Y(t)$ operative servers and the write queue is empty; if $N_0 + 1 \leq Y(t) < 2(N_0 + 1)$, there are $Y(t) - N_0 - 1$ operative servers and 1 write job; etc. That random integer can be thought of as a Markovian environment which controls the behaviour of the slow read queue.

*level 1*

The average fast read response time at level 1, $W_3$, is given by the known result for an M/M/1 queue with breakdowns and repairs [10]:

$$W_3 = \frac{N_1}{N_1\gamma - \lambda_3}[1 + \gamma\frac{\xi}{\eta(\xi + \eta)}] , \tag{6.8}$$

where $\gamma = \gamma_3\eta/(\xi + \eta)$ is the *effective service rate* for fast reads at the unreliable server.

We do not know, in the presence of breakdowns, how to compute exactly the probability that a replica at level 1 is consistent. An approximate value for $U$ can be obtained by replacing $q_1$, in (6.4), with the probability that there are either no

write jobs, or less than $Q_1$ operative servers, at level 0. That probability is provided by the spectral expansion solution. Also, the value of $q_3$, given by (6.6), should be multiplied by the probability that the replica is operative, $\eta/(\xi + \eta)$. The reason is that the repair time is usually orders of magnitude larger than the interarrival times; so if the level 1 replica is inoperative, there is likely to be a write job in its queue.

### 6.3.3 Results of Numerical experiments

Fig 6.4 gives results for a system with breakdowns and repairs. The arrival and service rates are fixed, as well as the total number of replicas and the repair rate; $W_2$ and $W_3$ are plotted against $N_0$ for different breakdown rates. The quorum sizes are $Q_1 = N_0$, $Q_2 = 1$. The curves plotted for slow read response time are essentially the same as the curves for lower priority read jobs in Fig 3.4 and the same explanation for their behavior applies here also.

Again, the trade-off between longer service time for writes and more parallelism for slow reads implies that there is an optimal degree of replication at level 0 . Moreover, our intuition tells us that the presence of breakdowns should generally make that optimal degree larger; the less reliable the servers, the more of them would be needed. That intuition is clearly confirmed by the experiments. An important feature to note in Fig 6.4 is the fast read response time. A slight increase in fault rate increases the fast read response time too much. In fact the last two curves plotted for $W_3$ show that fast read response time is larger than the response time of slow reads for all possible values of $N_0$. This is because when a server fails the fast read and write operations in queues associated with this server wait till the server becomes operative again. Due to this the whole purpose of providing two different type of read operations (slow read and fast read) is lost.

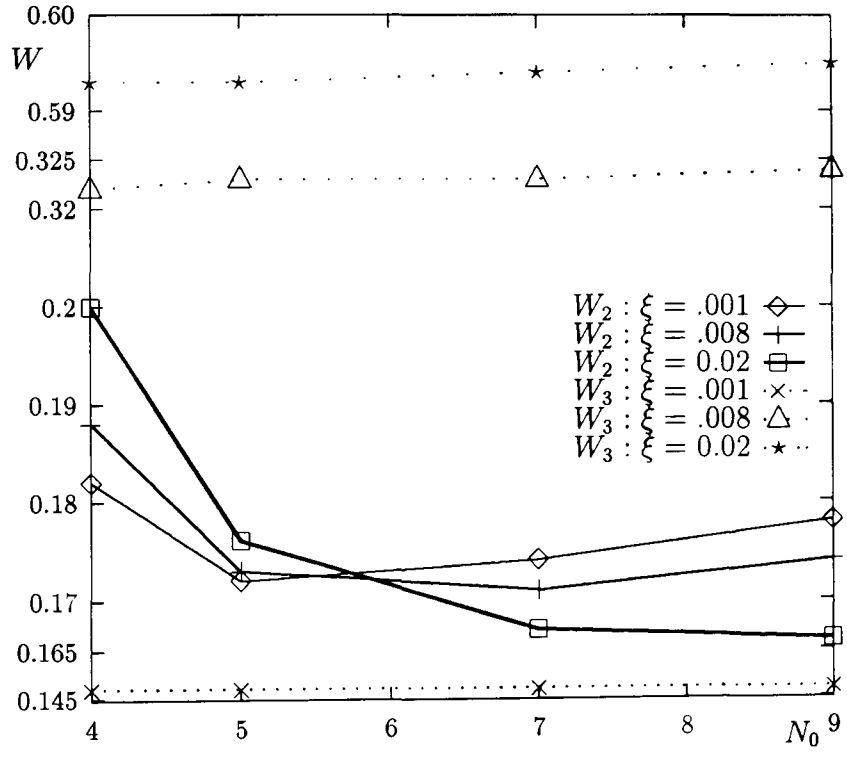



Figure 6.4: $W_2$ (solid) and $W_3$ (dotted) vs. $N_0$ in the presence of breakdowns.

$N = 12$, $\lambda_1 = 0.2$, $\lambda_2 = 15$, $\lambda_3 = 0.2$, $\mu_1 = 7$, $\mu_2 = \gamma_3 = 8$, $\eta = 0.2$

The last set of results deal with the role of quorum size in a model with breakdowns. The experiment illustrated in Fig. 6.5 mirrors the one in Fig. 6.3, as far as the response time of slow read jobs is concerned. It can be seen that even a slight unreliability of the servers (each of them is operative more than 99% of the time) can have a considerable effect on the curves. Now the quorum sizes $Q_1 = N_0$, $Q_2 = 1$ are optimal for all parameter values in the figure. However, if the performance measure is the overall average response time (including the write jobs), then it is again possible for the allocation $Q_1 = 1$, $Q_2 = N_0$ to be optimal.

Response time of fast read jobs is constant for all quorum sizes as $N_1$ remains fixed.

## 6.4 Generalizations

The modifications and generalizations suggested in section 3.5 may be applied in this case also for the level 0. In addition some other modifications that may be considered are:

- When a server at level 1 breaks down the fast reads waiting in its queue can be routed to other replicas. This is in practice the case. A client after waiting for the response for a predefined period (timeout period) may contact other servers for getting the value of data.

- Less restrictive assumptions concerning fast reads, such as generally distributed service times or unequal arrival rates at different level 1 replicas, could be accommodated easily.

- Replicas at level 1 may be updated on demand by the client.

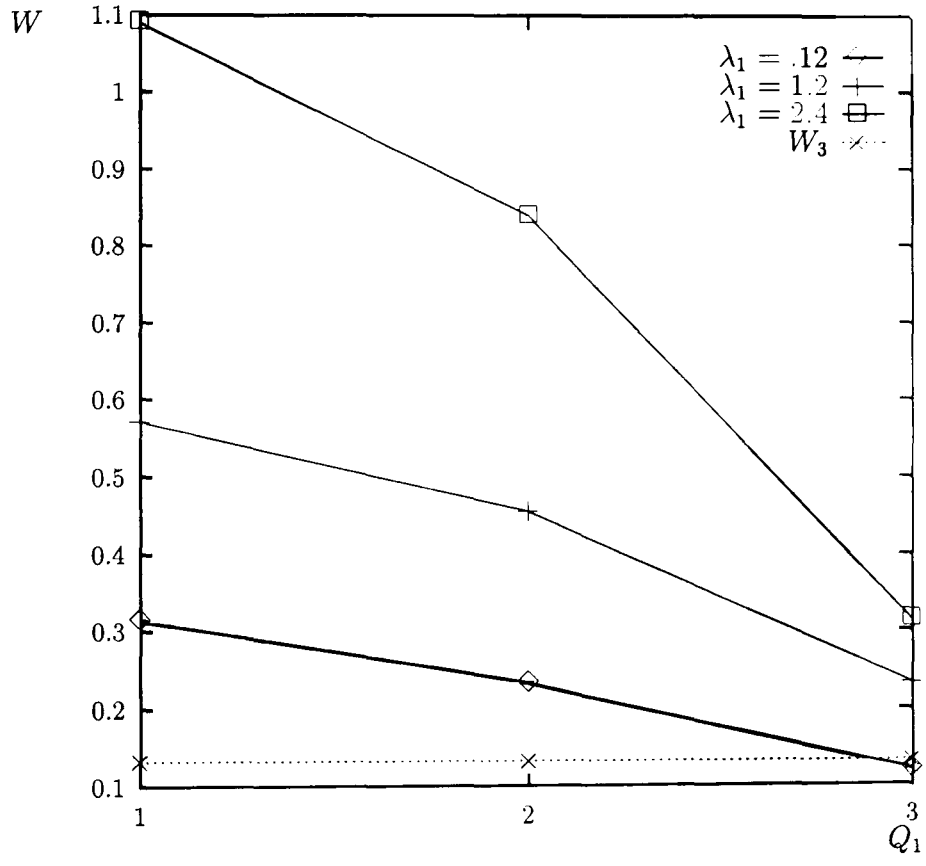



Figure 6.5: $W_2$ (solid) and $W_3$ (dotted) vs. $Q_1$ in the presence of breakdowns.

$N_0 = 3$, $N_1 = 9$, $\lambda_2 = 1.2$, $\mu_1 = 8$, $\mu_2 = \gamma_3 = 9$, $\xi = 10^{-7}$, $\eta = 0.004$

- Recent proposals have introduced bigger hierarchies of replicas – primary ,secondary, tertiary, etc. – with different scheduling policies at each level. While more complicated, such generalizations could be analyzed by similar methods.

## 6.5 Conclusion

The models presented here help us in understanding the behaviour of replicated data systems when replicas are organized in levels with each level providing a different type of consistency. The effects of different parameters can be evaluated and optimal decisions concerning the degree of replication, quorum sizes and the division of replicas between level 0 and level 1 can be taken. The solution of the model without breakdowns is exact; its numerical complexity is on the order of $O(N_0^3)$ (solving a set of $2N_0$ simultaneous linear equations). The model with breakdowns at level 0 is solved approximately but as accurately as desired, subject to constraints on computing resources. That solution involves finding the eigenvalues and eigenvectors of a matrix polynomial, and solving a set of simultaneous linear equations; its complexity is on the order of $O[(N_0S + N_0 + S)^3]$, where $S$ is the imposed bound on the number of write jobs in the system. The response time of fast reads at level 1 can be found exactly, without or with breakdowns, while the probability of reading a consistent replica is exact in the former case and approximate in the latter.

The following design guidelines are suggested by our results: For a given set of environmental parameters ($\lambda$'s, $\mu$'s, $\gamma$'s, $\xi$, $\eta$), and performance measures, one should first find the optimal configuration of level 0 , i.e., $N_0$ and $Q_1$. This is likely to consist of a small number of replicas. Then, as many replicas at level 1 should be provided as is economically feasible , in order to improve both the performance and the dependability of the fast reads . As shown the response time of fast read jobs

varies too much with the change in arrival rate of faults. One should therefore also examine that given a fault and repair rate how much improvement in response time can be achieved by providing fast reads.

# Chapter 7

# Weak Consistency Protocols

## 7.1 Introduction

Different applications have different consistency requirements, ranging from 'weak' (where the order and timing of updates may or may not be important), to 'strong' (insisting that all copies must be identical at all times). We assume a reasonably demanding requirement which is nevertheless realistic: all updates must be performed on all replicas in the order in which they arrive, although not necessarily at the same time. This is known as *sequence consistency*. Having received an update request, a site must wait until it knows about *all* preceding update requests received at other sites, before it can execute the new one.

Information about updates is propagated among the replica sites by means of messages. Each site maintains a log containing the outstanding update requests it knows about, together with the time and place of their origin. At random intervals, one of the sites sends its current (timestamped) log to another; the second site then incorporates the received log into its own. The frequency of these messages, and their destinations, depend on the system parameters. In order that a site may execute

an update request in its log, the latter must incorporate the logs of all other sites; moreover, those logs must have been sent after the request so as not to miss any previous updates.

This scheme of propagating update requests is of a type known as 'lazy replication'. The latter was proposed by R. Ladin *et al.* [39] for the purpose of handling a large class of weak consistency requirements, including sequence consistency. A similar scheme, whereby the recipient of a message replies immediately to the sender with its own current log, was suggested by R.A. Golding [27]. He used the name 'timestamped anti-entropy protocol'. We shall refer to the sending of a message in one direction as 'gossip', and to the sending of a message and receiving a reply as 'exchange gossip'. Both types of protocols will be examined.

The performance measure of principal interest is the average response time of an update request, i.e. the interval between the initial arrival of the request at one of the sites, and the first instant thereafter when the request can be executed on that site. To see why the problem of determining that quantity is non-trivial, consider a simple system with 3 replicated sites employing a gossip protocol and suppose that an update request arrives originally at site 1. Before that request can be executed, site 1 must receive gossip from sites 2 and 3, either directly or indirectly. Denoting the event 'site $i$ sends a gossip message to site $j$' by $\{i \to j\}$, we see that the response time of the request is the shortest interval until one of the following sequences of events occurs: $(\{2 \to 1\}, \{3 \to 1\})$, or $(\{3 \to 1\}, \{2 \to 1\})$, or $(\{2 \to 3\}, \{3 \to 1\})$, or $(\{3 \to 2\}, \{2 \to 1\})$. With exchange gossip, there are 16 similar sequences, since any exchange can be initiated by either of two protagonists, and site 1 may be one of them. Of course, in between the 'useful' messages, there may be many others that do not contribute new information to the log at site 1. One can readily appreciate

that, as the degree of replication grows, the combinatorial complexity of the problem increases very rapidly. An approach that reduces that complexity is required.

The detailed assumptions of the model are described in section 7.2. The response time problem is reduced to one of finding an average first passage time for a Markov process. However, the size of the state space makes the standard 'brute force' method impractical. We first give a solution in section 7.3 for the special case when all destinations other than the source are chosen with equal probability. An efficient solution is obtained in section 7.4, by showing that the problem is equivalent to another first passage time, for a possibly different Markov process, which turns out to be easier to solve. Indeed, in some special cases, the solution can be written in closed form. The special case of section 7.3 is then considered again and it is shown that the solution obtained from both approaches is same. Section 7.5 gives the analysis for the exchange-gossip scheme. In [20] F. Cristian *et al.* suggested the Train protocol for broadcasting updates. Section 7.6 gives a solution for the average response time of an update while using the scheme suggested in [20]. Some numerical results, including results for different network topologies are presented in section 7.7. Section 7.8 considers the problem of determining the average interval between the arrival of an update request at some site, and executing it on *all* sites. Upper and lower bounds on that interval, which is referred to as the 'sojourn time' of the request, are derived. An evaluation of the accuracy of the bounds is also presented. We conclude the chapter by mentioning the main results of the chapter.

## 7.2 The model

Data objects are replicated on $N$ different sites numbered $1, 2, \ldots, N$, communicating with each other via some network. Each site receives a stream of update requests

arriving locally and stores them in its log (accesses that do not modify the objects are not important in the present context and are ignored). The logs are propagated among the sites by means of gossip or exchange gossip messages, so as to implement the global sequence consistency requirement. The system is illustrated in figure 7.1.

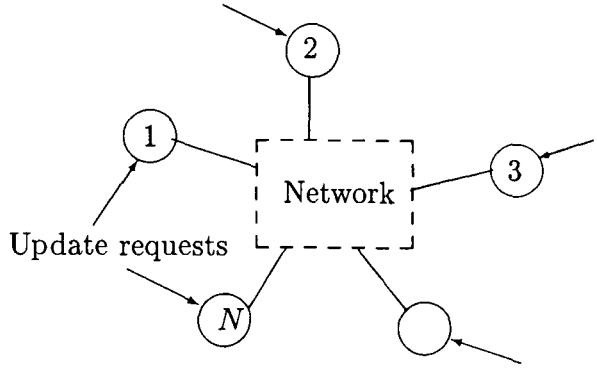



Figure 7.1: A replicated distributed system with $N$ sites

Consider first the gossip protocol. Site $i$ sends gossip messages at intervals which are i.i.d. random variables distributed exponentially with parameter $\mu$. The destination of each message is selected at random, regardless of past history: site $i$ chooses site $j$ with probability $q_{i,j}$ $(i, j = 1, 2, \ldots, N)$, such that $q_{i,i} = 0$ and

$$\sum_{j=1}^{N} q_{i,j} = 1 \ ; \ \ i = 1, 2, \ldots, N \ .$$

The product $q_{i,j}\mu$ represents the rate at which site $i$ sends gossip messages to site $j$. Since the intervals between messages are normally much larger than the message transmission times, the transmissions are assumed to be instantaneous.

The information about update requests received before a given moment in time is propagated among the sites by gossip sent after that moment. Suppose, without loss

of generality, that we start observing the system at an instant when a new update request arrives at site 1; denote the content of the site $i$ log at that instant by $L_i$. At a distance in time $t$ after the new arrival, the state of knowledge of the different sites is described by the Boolean matrix, $A(t) = [a_{i,j}(t)]$ $(i, j = 1, 2, \ldots, N)$, whose elements are equal to

$$a_{i,j}(t) = \begin{cases} 1 & \text{if site } i \text{ has incorporated } L_j \\ 0 & \text{otherwise} \end{cases} . \tag{7.1}$$

Note that site $i$ can obtain $L_j$ either directly from site $j$, or indirectly through another site. Thus, the value of $a_{i,j}(t)$ will become 1 at the first occurrence of one of the following events: either $\{j \rightarrow i\}$ (site $j$ sends a gossip message to site $i$), or $\{k \rightarrow i\}$ for some site $k$ such that $a_{k,j}(t) = 1$.

The above assumptions imply that $X = \{A(t)\,;\, t \geq 0\}$ is a Markov process. Its initial state is $A(0) = I$, where $I$ is the identity matrix of order $N$ (i.e. each site knows about its own log only). The sending of a gossip message may or may not cause a state transition, depending on whether the destination site already has the information provided by the sender. More precisely, if $\{j \rightarrow i\}$ at time $t$, then

$$A_{i,\cdot}(t) = A_{i,\cdot}(t-) \text{ or } A_{j,\cdot}(t-) \,, \tag{7.2}$$

where $A_{i,\cdot}$ is the $i$th row of matrix $A$, or is the element-by-element boolean OR operation and $t-$ is the time 'just before' $t$.

The response time, $R_1$, of the update request is the interval until site 1 incorporates all other logs, i.e. the smallest value of $t$ such that $A_{1,\cdot}(t) = \mathbf{1}$, where $\mathbf{1}$ is a vector of size $N$ whose elements are all equal to 1. More generally, the interval until the update can be executed on site $i$, $R_i$, is the smallest value of $t$ such that $A_{i,\cdot}(t) = \mathbf{1}$. From this discussion it is clear that $R_i$ may depend on $i$, but it does not depend on the site

where the request arrives. The important index in the definition of the response time is the site where the request is to be executed.

Thus, the problem of finding the average response time of an update request can be formulated as one of determining the average first passage time (FPT) of the Markov process $X$ from state $I$ to the set of states in which $A_{i,.}(t) = \mathbf{1}$, for some given $i$. These states are called *absorbing states.* In principle, the answer can be obtained by writing and solving a set of simultaneous linear equations. In practice, however, that approach is prohibitively expensive even for moderate values of $N$. This is due to the size of the state space. Since each non-diagonal element of the matrix $A$ can have two values, the number of possible states is $2^{N(N-1)}$. The number of equations that have to be solved is only slightly smaller—it is on the order of $O(2^{N(N-2)})$.

A more efficient solution is obviously desirable. We first solve the system for the special case when all replicas are identical. We then present an equivalent formulation of the system and give a solution for more general case.

## 7.3 Analysis

In this section we consider the case when system is homogeneous i.e. all replicas are identical and all destinations other than the source are chosen with equal probability. For this case $R_i$ is independent of $i$ and without loss of generality $i$ can be assumed as 1. We denote the average response time of an update by $E(R)$ as in this case the index can be omitted.

Let $K$ be the integer valued random variable representing the number of events of type $\{i \rightarrow j\}$ that occur until absorption is reached. Denote

$$m_p = E(R|K = p)$$

$$q_p = P(K = p)$$

Then $E(R)$ can be calculated by summing up the following series:

$$E(R) = \sum_{p=N-1}^{\infty} m_p q_p \tag{7.3}$$

$p$ takes values starting from $N-1$ because absorption is not possible in less than $N-1$ events. This is due to the fact that every site except 1 should send a gossip at least once before the system goes into some absorbing state.

Let $f_p(x)$ be the joint density function that absorption occurs after $p$ gossip messages and takes time $x$:

$$f_p(x)dx = P(R = x, K = p) \tag{7.4}$$

Consider the case when out of these $p$ messages, the site that sends the last gossip message sends $j$ messages and out of the remaining sites, $k_i$ sites send $i$ messages each, $i = 1, 2, \ldots, l$, where $l$ is the maximum number of messages send by a single site. Since, in order that absorption occurs, each site should send at least one message, the maximum value that $l$ can take is $p - N - 2$. Theses numbers satisfy

$$1 + k_1 + \cdots + k_l = N - 1, \tag{7.5}$$

$$j + k_1 + 2 * k_2 + \cdots + l * k_l = p \tag{7.6}$$

Note that a given site sends $i$ messages during an interval of length $x$ with probability $\frac{(\mu x)^i}{i!}e^{-\mu x}$. Therefore, the density $f_p(x)$ has the form

$$f_p(x) = \sum_S C_{k_1 k_2 \cdots k_l}((\mu x)e^{-\mu x})^{k_1} \cdots (\frac{(\mu x)^l}{l!}e^{-\mu x})^{k_l}\frac{(\mu x)^{j-1}}{(j-1)!}e^{-\mu x}\mu$$

where $C_{k_1 k_2 \cdots k_l}$ is the constant that gives the number of ways $k_i$ sites can be selected from total $N-1$ sites and $S$ contains all combinations of $j, k_1, \ldots, k_l$ s.t. equations (7.5) and (7.6) are satisfied. Because of (7.5) and (7.6), the above expression can be written as

$$f_p(x) = C\mu^p x^{p-1} e^{-(N-1)\mu x} \tag{7.7}$$

where $C$ is some constant.

Given this distribution $q_p$ and $m_p$ can be calculated as

$$\begin{aligned} q_p &= \int_0^\infty f_p(x)dx \\ &= C\int_0^\infty \mu^p x^{p-1} e^{-(N-1)\mu x} dx \\ &= \frac{(p-1)!}{(N-1)^p} C \end{aligned} \tag{7.8}$$

and

$$\begin{aligned} m_p &= \int_0^\infty x f_p(x)dx/q_p \\ &= C\int_0^\infty \mu^p x^p e^{-(N-1)\mu x} dx/q_p \\ &= C\frac{p!}{(N-1)^{p+1}\mu}/q_p \\ &= \frac{p}{(N-1)\mu} \end{aligned} \tag{7.9}$$

This last expression can be justified intutively by claiming that $\frac{1}{(N-1)\mu}$ is the average interval between two events and there are $p$ such events. This gives the following expression for E(R):

$$E(R) = \frac{1}{(N-1)\mu} \sum_{p=N-1}^{\infty} pq_p \tag{7.10}$$

Thus the problem of finding the average response time is reduced to that of determining the average number of message sequences leading to absorption in $p$ events.

To calculate $q_p$ let us consider first all sequences of messages containing p messages. As the total number of messages that can be sent in any state of the Markov process is $(N-1)^2$, there is a total of $(N-1)^{2p}$ such sequences (we will alternatively refer to these sequences of messages as paths). Not all of these sequences of messages lead to an absorbing state.

Let $S_p$ = number of message sequences leading to absorption in $p$ events

Then $q_p$ can be written as

$$q_p = \frac{S_p}{(N-1)^{2p}} \tag{7.11}$$

We introduce a shorthand notation for a sequence of gossip messages, $s(i,j,k,x)$, where $i$ is the site sending the first gossip message, $j$ is the site receiving the last gossip message, $k$ is the total number of gossip messages in the sequence and $x$ is an integer defined below. There may be more than one sequence denoted by the same shorthand notation. Replica 1 knows about updates of replica $j$ if there is a sequence of message(s) of the form

$$s(j,1,k,.) = \begin{cases} \{j \rightarrow 1\} & \text{if } k = 1 \\ \{j \rightarrow i\} s(i,1,k-1,.) & \text{if } i \neq 1, k > 1 \end{cases}$$

Given a sequence of $p$ messages, the system will be in an absorbing state if this sequence includes at least one suffix of the form $s(j,1,.,.)$ for every $j \neq 1$. To count all sequences leading to absorption, we define the variable $x$ for some message sequence $s$ as follows:

Define set $A_s$ to contain all replicas $j$ ($j \neq 1$) for which at least one suffix of the form $s(j,1,.,.)$ exists in $s$. Then $x$ is the cardinality of the set $A_s$. Since $A_s$ can not

contain more than $N-1$ sites, $x \leq N-1$. We also define set $A'_s = A_s \cup 1$. These definitions imply that any sequence of gossip messages of the form $s(j, 1, p, N-1)$ leads to absorption in $p$ steps.

Examine all possible $(N-1)^{2p}$ sequences of length $p$, starting from the last message. Based on the last message we can divide all $(N-1)^{2p}$ sequences into two groups (1) $(N-1)^{2p-1}$ message sequences, whose last message is $\{j \rightarrow 1\}$ for some $j \neq 1$. For all these sequences the subsequence consisting of the last message only has $x = 1$ and is of the form $s(j, 1, 1, 1)$. (2) remaining $(N-1)^{2p-1}(N-2)$ message sequences, whose last message is $\{j \rightarrow i\}$, for some $j$ and $i \neq 1$. Now consider the message sequences in group (1). Any one of these $(N-1)^{2p-1}$ sequences can be placed in one of the two groups based on the value of $x$ for the suffix subsequence consisting of messages $p-1$ and $p$. For one of these groups the message subsequence of length 2 (messages $p-1$ and $p$) will have $x = 2$ based on message $p-1$ and for the other group the subsequence of messages $p-1$ and $p$ will have $x = 1$.

In general, based on the last $p-k$ messages, all sequences can be divided into $\min\{N, p-k+1\}$ groups where for every sequence in group 0 the suffix of length $p-k$ is of the form $s(j, i, p-k, 0)$ and $i \neq 1$; in group 1 it is $s(j, 1, p-k, 1)$, in group 2 it is $s(j, 1, p-k, 2)$ and so on. Denote the group of message sequences where $x$ sites deliver their logs to site 1 in the last $p-k$ messages, by $G_{p-k,x}$, $x = 0, 1, \ldots, min(N-1, p-k)$. Any sequence which leads to absorption must belong to a group whose $x$ value satisfies $x \geq max(0, N-k-1)$. This can be explained as follows:

Since $x$ sites deliver their logs in the last $p-k$ messages, at least $N-1-x$ sites must do so in the first $k$ messages.

For a given suffix, $s$, of length $p-k$, let $Q_{k,s}(x)$ be the number of sequences from group $G_{p-k,x}$, which lead to absorption. This number will be the same for all different

suffixes of length $p-k$ having the same value of $x$, because the number of possible prefixes depends only on the value of $x$, and not on the actual sites that contribute to that value. This enables us to omit $s$ from the subscript of $Q_{k,s}(x)$ and define $Q_k(x)$ as the number of sequences leading to absorption, such that they all have the same (fixed) suffix of length $p-k$ during which $x$ sites deliver their logs to site 1. The quantity $Q_p(0)$ is the number of all sequences of length $p$ leading to absorption.

To write a recurrence for $Q_k(x)$ let us examine the relationship between groups. Consider a sequence, $s$, from the group $G_{p-k,x}$. It may belong to one of the two groups, $G_{p-k+1,x+1}$ or $G_{p-k+1,x}$, depending on the source and destination of message $k$. Let that message be $\{j \to i\}$.

(1) If $j$ is not among the sites counted in $x$ and $i$ is among those sites, or is 1, then $s$ is in $G_{p-k+1,x+1}$. There are a total of $(N-1-x)(x+1)$ such possibilities for message $k$.

(2) In all other cases the sequence belongs to the group $G_{p-k+1,x}$. The number of remaining possibilities is $(N-1)^2-(N-x-1)(x+1)$.

Thus $Q_k(x)$ satisfies the following recurrence relation:

$$\begin{aligned} Q_k(x) \;=\; & (N-x-1)(x+1)Q_{k-1}(x+1)+ \\ & ((N-1)^2-(N-x-1)(x+1))Q_{k-1}(x) \end{aligned} \tag{7.12}$$

The boundary conditions are:

I. $Q_0(x)=1$ for $x=N-1$

II. $Q_k(x)=0$ for $x>N-1$

III. $Q_k(x)=0$ for $x+k<N-1$

Now among the sequences leading to absorption in $p$ steps, there are some that in fact lead to absorption in less than $p$ steps. If the process has reached an absorbing state by step $p-1$, then it will remain in it regardless of message $p$. Since there are

$(N-1)^2$ possibilities for message $p$, the number of sequences such that absorption is reached for the first time at step $p$ is equal to:

$$S_p = Q_p(0) - (N-1)^2 Q_{p-1}(0) \tag{7.13}$$

It can be seen that the numerical complexity of the solution of (7.12) is $O(2^p)$. A more efficient solution for solving this system is given in next section that presents an equivalent formulation of the problem.

## 7.4 An equivalent formation

Consider the interval, $T_i$, between the arrival of an update request at site $i$ and the first instant thereafter when all other sites know about it (even if they may not be able to execute it). In terms of the Markov process $X$, defined in section 7.2, that is the first passage time from state $I$ to the set of states in which $A_{.,i}(t) = \mathbf{1}$, where $A_{.,i}$ is the $i$th column of matrix $A$.

The random variable $T_i$, which we shall call the 'spreading time' for site $i$, is of some interest in its own right. However, its main importance lies in the fact that it can be related to the response time, $R_i$, and is easier to model. First we need to introduce another system.

The model defined in the section 7.2, with parameters $N$, $\mu$ and $q_{i,j}$, will be referred to as the 'primary system'. Now consider a similar model with the same number of sites and the same value of $\mu$, where the probability that site $i$ chooses site $j$ as the destination of a gossip message, $\tilde{q}_{i,j}$, is equal to the probability that site $j$ chooses site $i$ in the primary system:

$$\tilde{q}_{i,j} = q_{j,i} \ ; \ \ i,j = 1,2,\ldots,N \ . \tag{7.14}$$

This will be called the 'dual system'. It exist when

$$\sum_{j=1}^{N} q_{j,i} = 1 \ .$$

This is assumed to be the case.

Clearly, duality is a symmetric relation; if the dual system is taken as primary, then the primary would be the dual. Moreover, if the primary system is symmetric, i.e. if $q_{i,j} = q_{j,i}$ for all $i$ and $j$, then it coincides with its dual.

The following result will provide the desired simplification:

**Lemma 1** The random variable $R_i$ in the primary system is equal, in distribution, to the random variable $T_i$ in the dual system.

**Proof:** Let $s = (\{k_1 \to k_2\}, \{k_3 \to k_4\}, \ldots, \{k_m \to i\})$ be a sequence of gossip-sending events in the primary system, such that the corresponding transitions of the process $X$ constitute a first passage from state $I$ to a state with $A_{i,\cdot}(t) = \mathbf{1}$. The messages in that sequence help to deliver the log contents of all other sites to site $i$. Denoting by $P(s)$ the probability that $s$ occurs and by $f_s(x)$ the p.d.f. of its duration, we can express the p.d.f. of $R_i$ in the form

$$f(x) = \sum_{s} P(s) f_s(x) \ , \tag{7.15}$$

where the summation is over all appropriate sequences.

Now consider the reverse sequence in the dual system: $\tilde{s} = (\{i \to k_m\}, \ldots, \{k_4 \to k_3\}, \{k_2 \to k_1\})$. The messages in $\tilde{s}$ carry information in the opposite direction: they propagate the log of site $i$ to all other sites. The corresponding transitions of the Markov process form a first passage from state $I$ to a state with $A_{\cdot,i}(t) = \mathbf{1}$. The p.d.f. of $T_i$ in the dual system can be written in a form similar to (7.15):

$$\tilde{f}(x) = \sum_{\tilde{s}} \tilde{P}(\tilde{s}) \tilde{f}_{\tilde{s}}(x) \,, \tag{7.16}$$

where $\tilde{P}$ and $\tilde{f}$ refer to probability and p.d.f. in the dual system. There is a one-to-one mapping between the sequences $s$ and $\tilde{s}$. Moreover, the definition of duality implies that $P(s) = \tilde{P}(\tilde{s})$ and $f_s(x) = \tilde{f}_{\tilde{s}}(x)$. Hence, $f(x) = \tilde{f}(x)$, QED.

Thus, the computation of an average response time in the primary system is reduced to the computation of an average spreading time in the dual system. This is a significant improvement, because the propagation of information from site $i$ to the other sites can be described by a much simpler Markov process, $Y$, whose state at time $t$, $\sigma(t)$, is the subset of sites that have already obtained the log of site $i$. The initial state of $Y$ is the subset consisting of site $i$ only: $\sigma(0) = \{i\}$. If site $i$ sends a gossip message to site $j$, there will be a transition to state $\{i, j\}$. Then, if either site $i$ or site $j$ sends a message to site $k$, $Y$ will jump to state $\{i, j, k\}$, etc. Since $\sigma(t)$ can only increase in size, the process $Y$ will reach state $\{1, 2, \ldots, N\}$ after exactly $N - 1$ transitions. The first passage time of $Y$ from state $\{i\}$ to state $\{1, 2, \ldots, N\}$ is the spreading time $T_i$.

If the process $Y$ is in state $\sigma$, and if $j$ is a site which is not in $\sigma$, then the transition rate from $\sigma$ to $\sigma \cup j$ is given by

$$\xi_{\sigma,\sigma\cup j} = \mu \sum_{k\in\sigma} \tilde{q}_{k,j} = \mu \sum_{k\in\sigma} q_{j,k} \,, \tag{7.17}$$

according to the definition of duality. The total rate of leaving state $\sigma$ is equal to

$$\xi_{\sigma} = \sum_{j\in\bar{\sigma}} \xi_{\sigma,\sigma\cup j} \,, \tag{7.18}$$

where $\bar{\sigma}$ is the complement of $\sigma$ with respect to $\{1, 2, \ldots, N\}$.

Let $E(T_\sigma)$ be the average first passage time of $Y$ from state $\sigma$ to state $\{1, 2, \ldots, N\}$. These quantities satisfy the following set of linear equations:

$$E(T_\sigma) = \frac{1}{\xi_\sigma} + \sum_{j \in \bar{\sigma}} \frac{\xi_{\sigma,\sigma\cup j}}{\xi_\sigma} E(T_{\sigma\cup j}) \,, \tag{7.19}$$

where $E(T_{\{1,2,\ldots,N\}}) = 0$ by definition.

Note that the equations in (7.19) are in fact recurrences in terms of the cardinality of $\sigma$. They can therefore be solved by successive substitution. When $\sigma$ contains $N-1$ sites, (7.19) gives $E(T_\sigma) = 1/\xi_\sigma$. The next application of (7.19) provides $E(T_\sigma)$ for $\sigma$ containing $N-2$ sites, and so on down to $E(T_i)$.

An important special case of this model is the 'homogeneous' system, solved in section 7.3, where all destinations other than the source are chosen with equal probability, i.e. $q_{i,j} = 1/(N-1)$ $(i \neq j)$. Then the primary and secondary systems coincide and the response and spreading times do not depend on the site: $E(R_i) = E(T_i) = E(R)$. The first passage times $T_\sigma$ depend only on the cardinality of $\sigma$ and not on its membership. Denote by $t_m$ the value of $E(T_\sigma)$ when $\sigma$ contains $m$ sites. Then the recurrences (7.19) yield

$$t_m = \frac{N-1}{m(N-m)\mu} + t_{m+1} \,, \tag{7.20}$$

with $t_N = 0$ by definition. Hence, the average response time

$$E(R) = t_1 = \frac{N-1}{\mu} \sum_{m=1}^{N-1} \frac{1}{m(N-m)} \,. \tag{7.21}$$

The equation (7.21) gives the same results as the equation (7.3). This last expression can be simplified by rewriting the terms under the summation sign in the form

$$\frac{1}{m(N-m)} = \frac{1}{N}\left[\frac{1}{m} + \frac{1}{N-m}\right] \,.$$

The two resulting sums are in fact identical. Therefore,

$$E(R) = \frac{2(N-1)}{N\mu} \sum_{m=1}^{N-1} \frac{1}{m} = \frac{2(N-1)H_{N-1}}{N\mu} , \tag{7.22}$$

where $H_n$ is the $n$th harmonic number. Thus, when $N$ is large, the average response time is approximately equal to

$$E(R) \approx \frac{2 \ln N}{\mu} . \tag{7.23}$$

There are other special cases where the solution may be obtained in closed form. For example, suppose that the sites are connected by a one-directional ring network, with site $i$ sending messages to site $i+1$ ($i = 1, 2, \ldots, N-1$), and site $N$ to site 1. The dual system here is the opposite-directional ring, where site $i$ sends messages to site $i-1$ ($i = 2, 3, \ldots, N$), and site 1 to site $N$. It is not difficult to see that in this case,

$$E(R) = \frac{N-1}{\mu} . \tag{7.24}$$

## 7.5 Exchange Gossip

The analysis in this section applies almost without change to the case where the sending of a gossip message from site $i$ to site $j$ (with probability $q_{i,j}$) prompts an immediate message from site $j$ to site $i$. Similarly, if site $j$ sends a gossip message to site $i$ (with probability $q_{j,i}$), the latter will reply immediately with a message of its own. If such an exchange occurs at time $t$, the resulting transition of the Markov process $X$ is

$$A_{i,\cdot}(t) = A_{j,\cdot}(t) = A_{i,\cdot}(t-) \text{ or } A_{j,\cdot}(t-) .$$

Again there is an equality between a response time in the primary system and a spreading time in the dual one. The transition rates for the process $Y$ are now different, since an exchange that increases the membership of $\sigma$ can be initiated either from within $\sigma$ or from outside. The new equation corresponding to (7.17) has the form

$$\xi_{\sigma,\sigma\cup j} = \mu \sum_{k\in\sigma} (q_{j,k} + q_{k,j}) \,. \tag{7.25}$$

Equations (7.18) and the recurrences (7.19) remain valid.

In the homogeneous special case, where all exchange gossip destinations are equally likely, the formula (7.21) becomes

$$E(R) = \frac{N-1}{2\mu} \sum_{m=1}^{N-1} \frac{1}{m(N-m)} \,. \tag{7.26}$$

When $N$ is large, this is approximately equal to

$$E(R) \approx \frac{\ln N}{\mu} \,. \tag{7.27}$$

In the ring network, there is no difference between gossip and exchange gossip.

## 7.6 Train Protocol

In the Train protocol described by the F. Cristian *et al.*, [20] there is a cyclic order among sites. A train containing a sequence of updates circulates from one site to another in this order. A site that wants to broadcast an update waits for the train to arrive. When the train arrives the sender first delivers all updates carried by the train, and then appends all updates that it wants to broadcast at the end of the train. The sender removes these updates when he sees the train again. It can be readily

seen that this scheme behaves exactly in the same way as gossip when the sites are connected by a one-directional ring network, with site $i$ sending messages to site $i+1$ ($i = 1, 2, \ldots, N-1$), and site $N$ to site 1. The average response time of an update, therefore, can be given by equation (7.24).

## 7.7 Numerical experiments

We first examine the effect of increasing the number of replicas (fully reliable). The performance measure of interest is the average execution time of an update. The trade-off here is between the advantage of increasing the number of replicas (as it increases the read parallelism) and the disadvantage of longer service time for write jobs. Figure 7.2 shows increase in average response time of write jobs against number of replicas. This increase is logarithmic. Figure 7.3 compares the performance of the schemes used by gossip, exchange-gossip and train protocol for spreading updates. It is clear from the graph that performance of gossip and exchange-gossip schemes are better than train protocol in most cases. The trade-off between the performance of gossip and exchange-gossip actually depends on the time taken to send a gossip and time taken to exchange the information between two replicas. If both are same exchange-gossip scheme performs much better than gossip but if later is large enough in comparison to former then the performance of gossip protocol is better.

It is intuitively obvious that the higher the connectivity of the network by means of which the sites communicate with each other, the better the performance of the replication protocol will be. This observation has already been quantified in two extreme cases. In a fully connected, homogeneous network, the average response times grow logarithmically with $N$. In a minimally connected, ring network, those averages grow linearly with $N$.

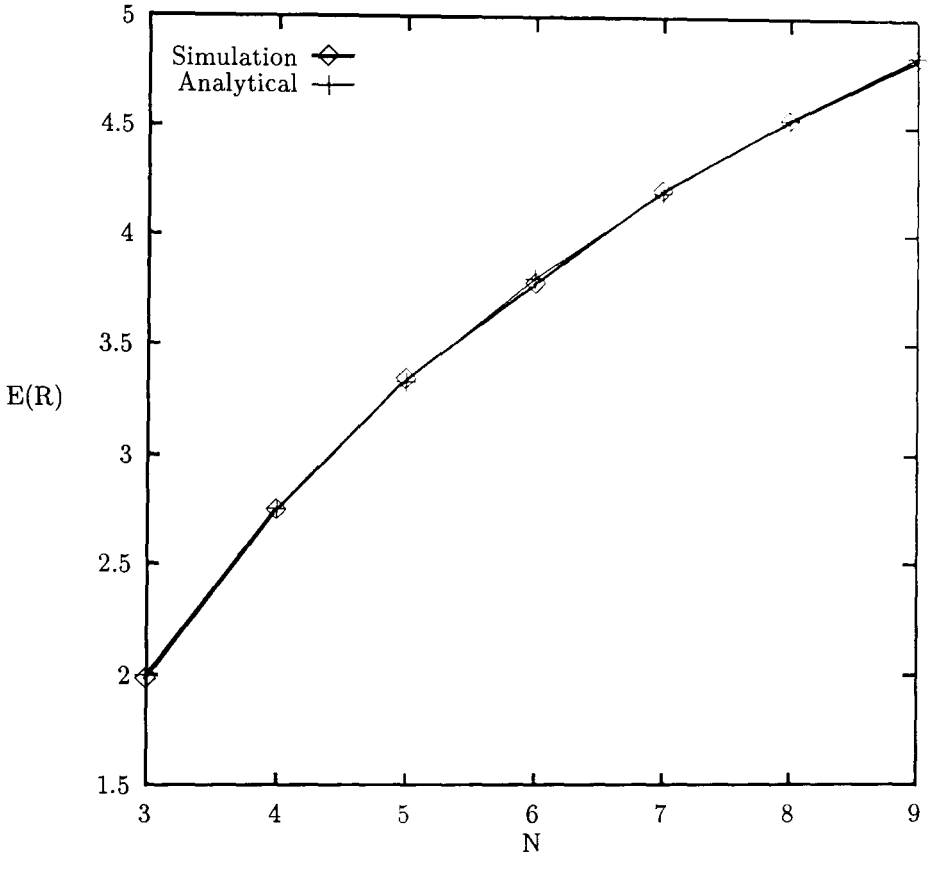



Figure 7.2: Analytical vs. simulation results for write response time

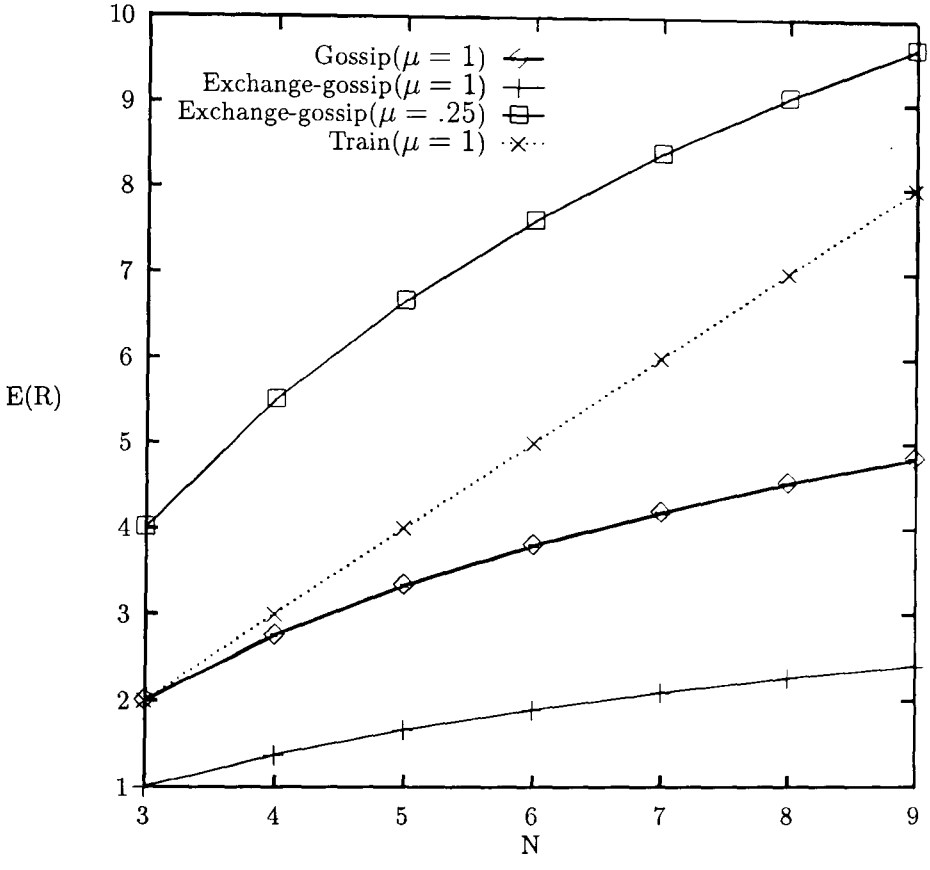



Figure 7.3: Response time for gossip, exchange-gossip and Train schemes

For the purpose of comparison, we have also examined an intermediate case—a mesh network where every site sends messages to each of its four immediate neighbours with probability 1/4. In order to simplify the calculations, it is assumed that the mesh is similar to a torus, i.e. the left-hand neighbour of a site on the left-hand edge is the corresponding site on the right-hand edge; the former is the right-hand neighbour of the latter; the top neighbour of a site on the top edge is the corresponding site on the bottom edge; the former is the bottom neighbour of the latter. An 8-site mesh of this type is illustrated in Figure 7.4.

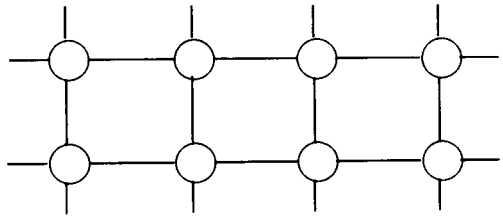


Figure 7.4: A mesh network with 8 sites

Since the routing matrix of the mesh network is symmetric, the primary and dual systems coincide. The average response time and the average spreading time are equal and do not depend on the site.

The performance of the ring, mesh and fully connected networks, measured by the average response time as a function of the number of sites, is shown in Figure 7.5. The mesh results were obtained by solving the recurrence equations (7.19).

As expected, the mesh performs better than the ring, but not as well as the fully connected network. The graph appears to suggest that the mesh average response time grows approximately linearly with $N$, but at a lower rate than the ring.

Figure 7.6 compares the results obtained by summing the series of (7.21) with the

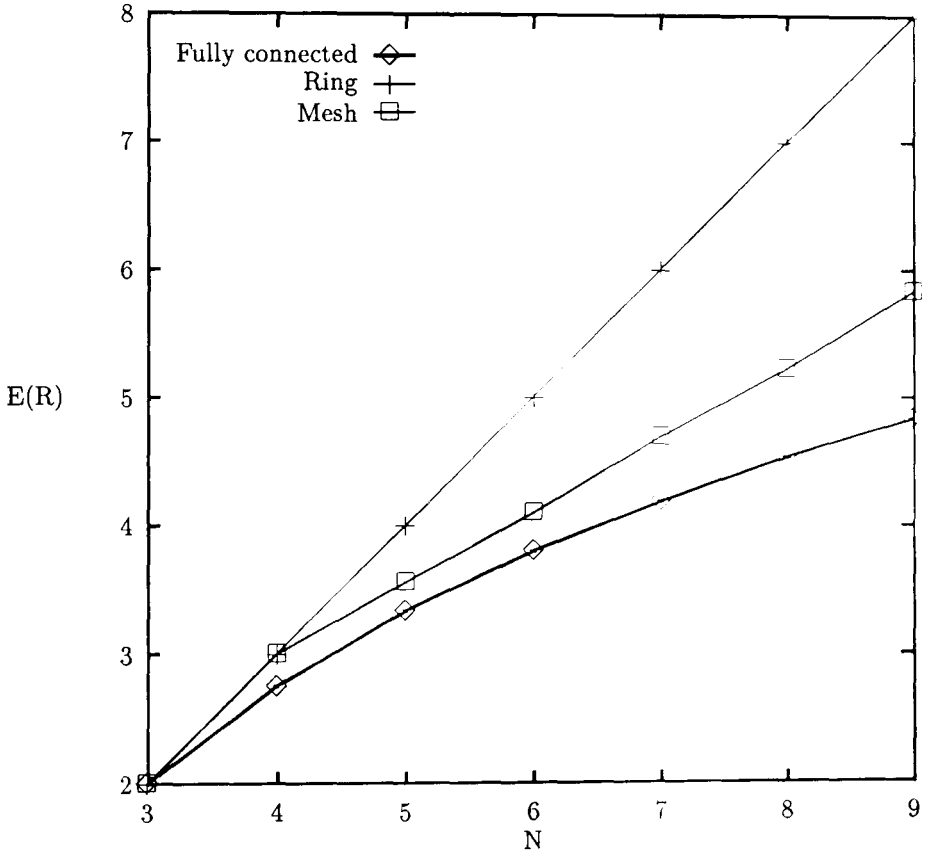



Figure 7.5: Response time for ring, mesh and fully connected networks

results obtained by direct formula given in (7.23).

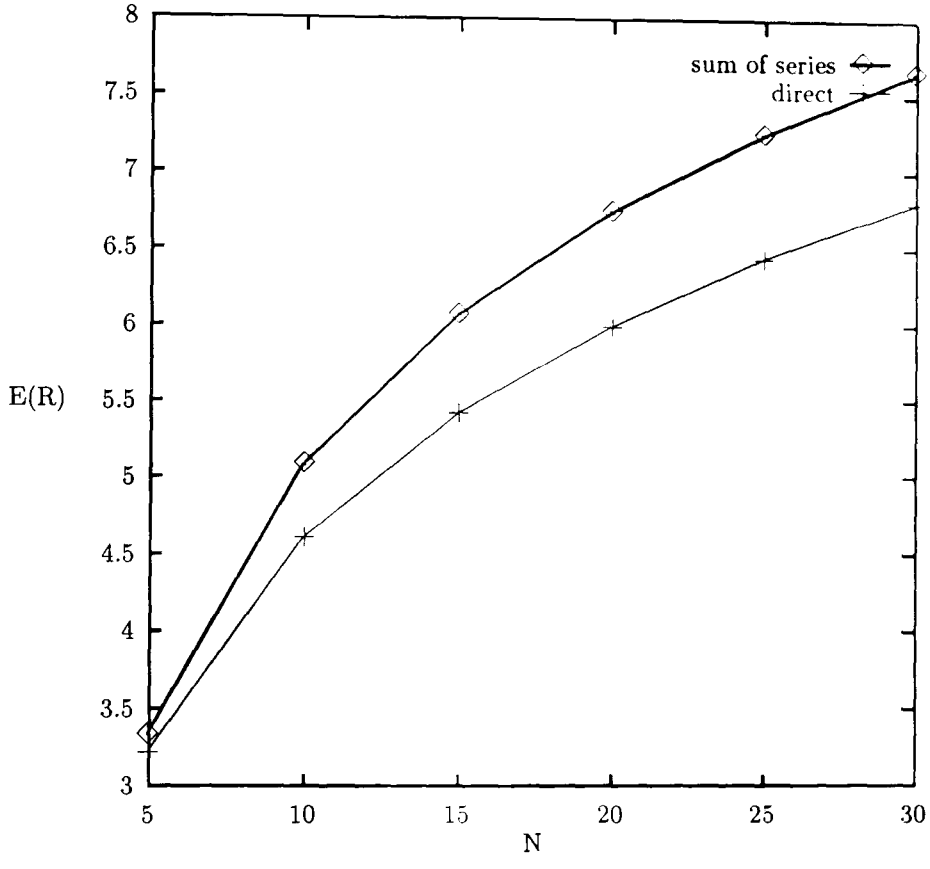



Figure 7.6: Results obtained by direct and indirect (series sum) methods

## 7.8 Sojourn time of an update request

The interval, $S$, between the arrival of an update request and its execution at all sites is called the 'sojourn time' of the request. Within that interval, every site obtains the log of every other site. In terms of the Markov process $X$, whose state is the Boolean matrix $A(t)$, this is the first passage time from state $I$ to the state $A(t) = 1$ (all elements of $A(t)$ are equal to 1). Unfortunately, we do not have an efficient algorithm for calculating the average sojourn time exactly. Instead, it is possible to provide reasonably tight upper and lower bounds.

To derive a lower bound for the gossip protocol, let $D$ be the interval until all but one of the sites have sent at least one message each. At the end of $D$, there is still one site which has not yet sent a message. Clearly it must do so, and its log must be propagated to all other sites, before the sojourn time can complete. Hence, $S$ is bounded below by the sum of $D$ and the spreading time of that last site.

Since we do not know its index, we write

$$E(S) \geq E(D) + \min_i E(T_i) \,. \tag{7.28}$$

To find $E(D)$, note that the average interval until the first site to send a message is $1/(N\mu)$; after that, the average interval until the second site to send a message is $1/((N-1)\mu)$, etc. Hence,

$$E(D) = \frac{1}{\mu} \sum_{m=2}^{N} \frac{1}{m} = \frac{H_N - 1}{\mu} \,, \tag{7.29}$$

where $H_N$ is the $N$ th harmonic number.

In the homogeneous special case, where $q_{i,j} = 1/(N-1)$ $(i \neq j)$, all average spreading times are equal and (7.28) becomes

$$E(S) \geq \frac{1}{\mu}\left[3H_{N-1} - \frac{2H_{N-1}+N-1}{N}\right] . \tag{7.30}$$

For large values of $N$ this is approximately

$$E(S) \geq \frac{3\ln N}{\mu} . \tag{7.31}$$

An upper bound on $E(S)$ can be obtained by remarking that if one waits for site $i$ to absorb all other logs (the response time of site $i$), and after that for site $i$ to propagate its log to all other sites (the spreading time of site $i$), then the sojourn time will certainly complete. Since that is true for all $i$, we can write

$$E(S) \leq \min_i [E(R_i) + E(T_i)] . \tag{7.32}$$

Note that both $E(R_i)$ and $E(T_i)$ in the above equation refer to the primary system. If the system is symmetric, then those two quantities are equal. For such systems, the upper and lower bounds differ by a factor less than 2.

When $q_{i,j} = 1/(N-1)$ $(i \neq j)$, the upper bound is

$$E(S) \leq \frac{4(N-1)H_{N-1}}{N\mu} , \tag{7.33}$$

which for large $N$ is approximately

$$E(S) \geq \frac{4\ln N}{\mu} . \tag{7.34}$$

In this case, since both bounds are logarithmic, the true value of $E(S)$ must also be on the order of $O(\ln N)$, with some coefficient whose value is between $3/\mu$ and $4/\mu$.

If the sites are connected by a ring network, the logarithmic contribution of $E(D)$ in the lower bound is dominated by the linear growth of $E(T_i)$. Then the two bounds are

$$\frac{N-1}{\mu} \leq E(S) \leq \frac{2(N-1)}{\mu}$$

Similar results can be obtained for the exchange gossip protocol. The upper bound (7.32) applies without modification. In the lower bound, the interval $D$ no longer helps, since a site may participate in an exchange without initiating it. A simpler, if worse, lower bound is provided by

$$E(S) \geq \min_i E(T_i) \,. \tag{7.35}$$

In the homogeneous system under exchange gossip, the average sojourn time is bounded by

$$\frac{(N-1)H_{N-1}}{N\mu} \leq E(S) \leq \frac{2(N-1)H_{N-1}}{N\mu} \,, \tag{7.36}$$

which for large $N$ is approximately equal to

$$\frac{\ln N}{\mu} \leq E(S) \leq \frac{2\ln N}{\mu} \,. \tag{7.37}$$

For the ring network, the same linear bounds as before apply.

An estimate for the average sojourn time in the fully connected homogeneous network, as a function of $N$, was obtained by simulation. The width of the confidence interval was less than 2% of the performance value. The results are presented in Figure 7.7, together with the analytical upper and lower bounds. It is evident that, at least in this system, the lower bound is much closer to the true value of the performance measure than the upper bound. However, it is possible, by exploiting the structure of the model, to derive an improved upper bound which comes within 20% of the true values.

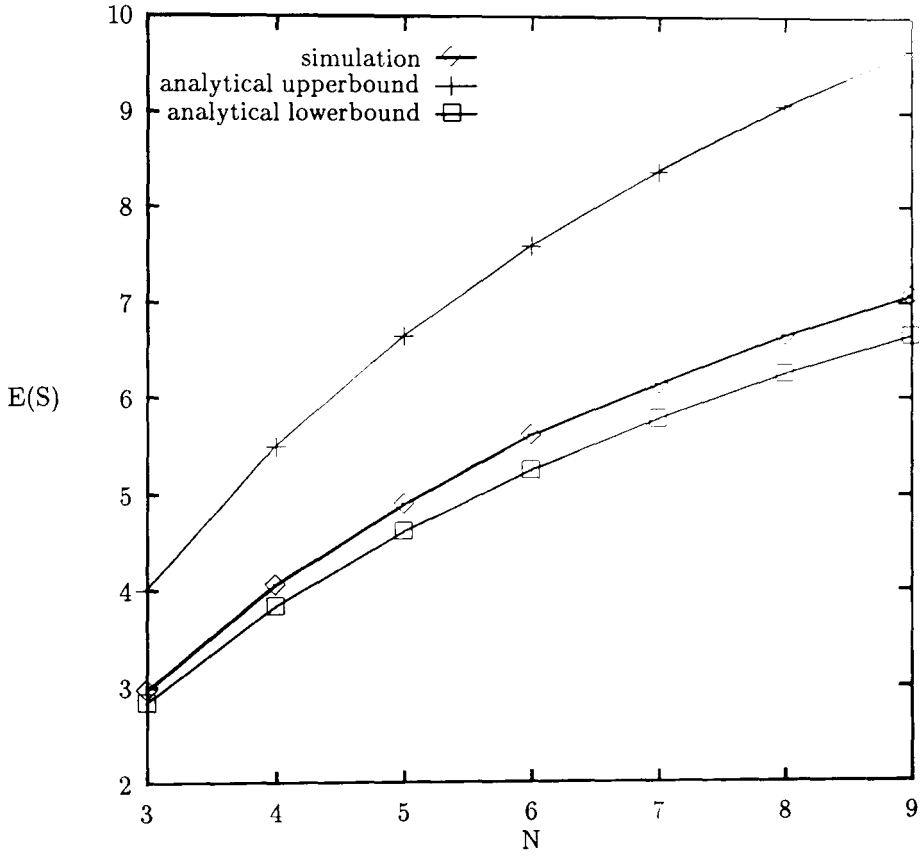



Figure 7.7: Bounds and simulation estimates for $E(S)$

## 7.9 Conclusion

The analysis given here provide useful insight into the behaviour of replicated data systems. The effects of different parameters can be evaluated and decisions concerning the degree of replication and choice of protocol to be used can be taken. The solution of the model that gives delay time for an update is exact; its numerical complexity is of the order of O($N$). The numerical complexity of upper bound on the time when the update can be executed on all replicas is also of the order of O($N$).

The study can further be extended to consider the case when replicas are subjected to breakdown and repair. The main results in this chapter can be summerizes as:

[1.] The equality in distribution between the response time in the primary system and the spreading time in the dual system.

[2.] The recurrence equations for calculating $E(T_i)$.

[3.] The explicit formula for the average response time in the fully connected homogeneous network and the corresponding logarithmic approximation.

[4.] The upper and lower bounds for the average sojourn time and their explicit versions in the fully connected homogeneous network.

We still do not know how to analyze a replicated system where the dual does not exist, i.e. where the rows of the routing matrix add up to 1 but the columns do not. This is an interesting open topic for future research.

# Chapter 8

# Conclusions

This chapter summerizes the work which has been presented in this thesis and suggests some possible areas for further research.

## 8.1 Summary of Thesis

This thesis presents the performance evaluation of different data replication protocols. We have analysed these protocols both when replicas are reliable and when breakdowns and repairs may occur. We selected the queueing theory approach to analyse these protocols as it captures the effect of delays caused by queueing and congestion at various nodes of the network. We first classified these protocols into three categories based on their approach to implement updates. These categories are: Strong consistency protocols, Weak consistency protocols and multi-level consistency protocols.

The first part of the thesis concentrates on protocols that follow the quorum based approach to maintain strong consistency. An analysis of these protocols both in case of reliable replication and unreliable replication has been presented. The

effect of scheduling strategies on the performance of the protocol has been studied. We considered two scheduling strategies: Priority and FIFO. The model in the case of unreliable replication with priority scheduling has $N$ servers subject to breakdowns and repairs and two classes of jobs. We found that the exact analysis of this model is yet intractable. The analysis presented is approximate yet very close to exact. To show this we gave an exact solution for one server subject to breakdowns and repairs and two classes of jobs. A comparison of results for the two cases (approximate results and exact results in case of one server) justifies our claim.

Quorum based schemes for maintaining strong consistency need synchronization among a large number of replicas. Due to this they do not perform well for wide area networks where the number of replicas may be hundreds or thousands. The performance can be improved by relaxing the consistency constraints. The second part of the thesis analyzes protocols that take advantage of this fact to allow some replicas to be out-of-date while maintaining a group of replicas strongly consistent. A user reading from a replica may be interested in knowing the probability of that replica being up to date. A method to calculate this probability has also been presented. These results have been published in IPDS'96.

Finally we presented the analysis of the protocols that allow updates to occur on any replica asynchronously. Replicas store these updates in their log and later propagate them to other replicas by sending messages containing information about these updates. The two schemes of propagating updates have been analysed. One where replicas send gossip messages to each other randomly and the other where two replicas exchange messages with each other. The updates are being implemented in the order of arrival. Response time of an update has been formulated as the first passage time from the arrival of the update on some replica until the time when that

replica is ready to execute it. The effect of the network topology on the performance of the protocol has also been studied. Upper bounds and lower bounds on the time until all replicas are ready to execute a given update have also been established.

## 8.2 Contributions

Each chapter states the contributions made in the area covered in that chapter. A brief summary of these contributions is, as follows:

- The analysis of quorum based protocols for reliable replication shows that there is an optimal degree of replication that gives the best performance. The performance measure is the response time of the jobs. As the number of replicas increases the response time decreases but once this optimal degree of replication is reached any increase in number of replicas increases the response time instead of decreasing it.

- The analysis of quorum based protocols for reliable replication and with fixed number of replicas shows that the optimal choice for quorum depends on the arrival rate of jobs. When most of the offered load consists of read jobs, the allocation $R = 1, W = N$ is best, whereas $R = N, W = 1$ is preferable when most of the load consists of the write jobs.

- The analysis of quorum based protocols with breakdowns and repairs shows the same behaviour. There is an optimal degree of replication in this case also which is generally larger than the optimal degree of replication in the case of reliable replication.

- It is shown that even a slight unreliability of servers have a considerable effect on the shape of curves plotting the response time of jobs. It not only affects

the optimal degree of replication but also the choice of quorum sizes. For the cases where $R = N, W = 1$ give the best performance when replicas are reliable, the choice of $R = 1, W = N$ may give better performance in case of unreliable replication.

- A comparison of priority scheduling and FIFO scheduling for quorum based protocols shows that in many cases better performance can be achieved by assigning higher priorities to the jobs with larger arrival rate.

- A model with one server subject to breakdowns and repairs and two job types is considered. An exact solution of this model with priority scheduling for write jobs is presented.

- The problem of data replication with two levels (level 0 and 1) of consistency is considered. The performance measure of interest is the response time of slow reads (that need most recent value of data) and fast reads (that may read older versions of data). The replicas at level 0 are always strongly consistent and a slow read reads from level 0 replicas. Replicas at level 1 may be out-of-date and a fast read reads from some level 1 replica. It is shown that the states of level 0 and level 1 are independent of each other. The analysis for both levels is presented. As the level 0 uses quorum based protocols to maintain consistency, the analysis of level 0 is same as the one for quorum based protocols. All results that hold for quorum based protocols apply for level 0 also. The response time for higher priority jobs at level 1 can be obtained by treating a level 1 replica as an M/M/1 queue. The curves showing the response time of slow and fast read jobs are presented. The analysis gives an insight into the behaviour of the system and helps in making optimal decisions concerning the degree of

replication, quorum sizes and the division of replicas between level 0 and level 1.

- An exact method to calculate the probability that a level 1 replica is out-of-date is presented for reliable replication. A method to calculate the approximate value of this probability is suggested in case of breakdowns and repairs.

- Two schemes, gossip and exchange gossip, for spreading updates among geographically distributed sites are studied. The analysis when replicas execute these updates in the order of their arrival is presented. The problem of finding the average response time of an update can be formulated as one of determining the first passage time from the state when a replica receives the update to the state when it knows about all preceding updates arrived at other replicas. It is shown that the complexity of this problem is of the order of $O(2^{N(N-2)})$. To solve this problem more efficiently we use a counting argument leading to recurrence relations, and also an approach based on a dual system and a different first passage time. Closed form solutions and single logarithmic approximations are obtained in some special cases.

- It is shown that the connectivity of the replicas affects the performance of the scheme. The greater the connectivity, the better is the performance.

- We give closed form solutions for the upper and lower bounds on the time when all replicas are ready to implement the update for a fully connected homogeneous network.

## 8.3 Further Work

This research gives an insight into the behaviour of replication protocols. It also leaves many questions unanswered for future research. These problems that need more research have already been highlighted in the chapter covering the material related to that area. Here we summarize these open problems for research.

- The analysis of quorum based protocols in this thesis assumes that all replicas are identical. The system can still be solved when the replicas are not identical (write service time is different for each write request) provided (1) the read quorum is one and write quorum is all (2) the read service time has the same distribution for all read services. The problem to analyze the protocols for a more general case when both read and write service times are different and quorum sizes are also other than 1 and $N$ is open for research.

- The thesis compares the performance of quorum based protocols for two scheduling strategies: priority and FIFO. The performance of the scheme with other scheduling strategies may be examined.

- We presented the analysis of a two-level replication. Recent proposals have introduced bigger hierarchies of replicas. It may be worth analysing systems with more than two levels. While more complicated, such genralizations could be analysed by similar methods used for analysing two-level replication.

- The analysis presented for two-level replication assumes that all updates arrive at level 0 only. They are then propagated to replicas at level 1. Schemes where updates may be submitted at any level or where replicas at level 1 are updated on demand may be studied.

- The analysis presented in this thesis for two-level replication considers each replica at level 1 in isolation. If a replica breaks down, the jobs in its queue wait for its repair. In a more general case these jobs can be routed to other replicas at level 1 which are not faulty. The analysis of this case may be useful.

- The analysis of gossip and exchange gossip schemes considers that updates are executed in the order they arrive. The protocols suggested for providing weak consistency support many other orderings: causal, ordering imposed by client etc. The problem to study the performance of these protocols analytically for other cases is still open for research.

- The thesis gives upper and lower bounds on the time when an update can be executed on all replicas. An attempt to get an exact solution for this time may be worth trying.

- The analysis of gossip and exchange gossip schemes can further be extended to consider the case when replicas are subjected to breakdown and repair.

## 8.4 Concluding Remarks

With development of large scale wide area systems and use of replication in these systems to provide (a) fault tolerance and (b) improve performance it has become vital to study the performance of these protocols. We hope that the analysis presented in this thesis will be helpful in making decisions about different aspects of replication and will motivate people to continue research in this area.

# Appendix A

The 'brute force' approach which relies on first evaluating the scalar polynomial $det[Q(z)]$, then finding its roots, is very inefficient for large $N$ and is therefore not recommended. An alternative which is preferable in most cases is to reduce the quadratic eigenvalue-eigenvector problem

$$\psi[Q_0 + Q_1 z + Q_2 z^2] = \mathbf{0} \,, \tag{A.1}$$

to a linear one of the form $\psi Q = z\psi$, where $Q$ is a matrix whose dimensions are twice as large as those of $Q_0$, $Q_1$ and $Q_2$. The latter problem is normally solved by applying various transformation techniques. Efficient routines for that purpose are available in most numerical packages.

This linearization can be achieved quite easily if the matrix $Q_2 = C$ is non-singular. Indeed, after multiplying (A.1) on the right by $Q_2^{-1}$, it becomes

$$\psi[H_0 + H_1 z + I z^2] = \mathbf{0} \,, \tag{A.2}$$

where $H_0 = Q_0 C^{-1}$, $H_1 = Q_1 C^{-1}$, and $I$ is the identity matrix. By introducing the vector $\mathbf{\Psi} = z\psi$, equation (A.2) can be rewritten in the equivalent linear form

$$[\psi, \mathbf{\Psi}] \begin{bmatrix} 0 & -H_0 \\ I & -H_1 \end{bmatrix} = z[\psi, \mathbf{\Psi}] \,. \tag{A.3}$$

If $C$ is singular but $B = Q_0$ is not, a similar linearization is achieved by multiplying (A.1) on the right by $B^{-1}$ and making a change of variable $z \to 1/z$. Then the relevant eigenvalues are those outside the unit disk.

If both $B$ and $C$ are singular, then the desired result is achieved by first making a change of variable, $z \to (\gamma + z)/(\gamma - z)$, where the value of $\gamma$ is chosen so that the matrix $S = \gamma^2 Q_2 + \gamma Q_1 + Q_0$ is non-singular. In other words, $\gamma$ can have any value which is not an eigenvalue of $Q(z)$. Having made that change of variable, multiplying the resulting equation by $S^{-1}$ on the right reduces it to the form (A.2).

# References


[1] N. Adly, M. Nagi, and J. Bacon. "A Hierarchical Asynchronous Replication Protocol for Large Scale Systems", *Proc. of the IEEE Workshop on Parallel and Distributed Systems*, pp. 152-157, Princeton, 1993.

[2] N. Adly, M. Nagi, and J. Bacon. "Performance Evaluation of a Hierarchical Replication Protocol: Synchronous versus Asynchronous", *Proc. IEEE SDNE Services in Distributed and Networked Environments*, pp. 102-109, Whistler, 1995.

[3] N. Adly. "Management of replicated data in large scale systems", *Technical Report, No. 383*, University of Cambridge, Computer laboratory, November 1995.

[4] D. Agrawal, A. El Abbadi, "Efficient Techniques for Replicated Data Management", *Procs. Workshop on Management of Replicated Data*, 48-52, 1990.

[5] D. Agrawal, A. El Abbadi, "Quorum Consensus Algorithm for Secure and Reliable data", *TRCS-88-17*, Department of Computer Science, UCS B, 1988.

[6] M. Ahamad and M.H.Ammar, "Performance Characterization of Quorum - Consensus Algorithms for Replicated Data", *IEEE Trans. on Software Engineering*, **15**, 492-496, 1989.

[7] M. Ahamad, M.H.Ammar and S.Y.Cheung, "Optimizing the Performance of Quorum Consensus Replica Control Protocols", *Procs. Workshop on Management of Replicated Data*, 102-107, 1990.

[8] Akhil Kumar, "Hierarchical Quorum Consensus: A new algorithm for managing replicated data", *IEEE Transactions on Computers, Vol. 40, No. 9* 996-1004, September 1991.

[9] R. Alnoso, D. Barbara and H. Gracia-Molina, "Data Caching issues in an information retrieval system", *ACM Transactions on database systems*, Vol. 15, 359-384, 1990.

[10] B. Avi-Itzhak and P. Noar, "Some Queueing Problems with the Service Station Subject to Breakdowns", *Operations Research*, **11** ,303-320, 1963.

[11] F. Baccelli and E.G. Coffman, Jr., "A Data Base Replication Analysis Using an M/M/m Queue with Service Interruptions", *Performance Evaluation Review*, **11**, 102-107, 1982.

[12] D.Barbara and H. Gracia-Molina. "The case for controlled inconsistency in replicated data", *Procs. Workshop on Management of Replicated Data*, 35-38, 1990.

[13] P. A. Bernstein, V. hadzilacos and N. Goodman, " Concurrency Control and Recovery in Database Systems", Addison-Wesley, 1987.

[14] A. D. Birrell, R. Levin, R. M. Needham and M. D. Schroeder, "Grapevine: An exercise in distributed computing", *Communications of the ACM*, Vol. 25, No. 4, 260-274, April 1982.

[15] S. Y. Cheung, M.H.Ammar and M. Ahamad, "The Grid protocol: A high performance scheme for maintaining replicated data", *Proc. of 6th International Conference on Data Engineering*, 438-445, IEEE, 1990.

[16] E.G. Coffman, Jr., E. Gelenbe, and B. Plateau, "Optimization of the Number of Copies in a Distributed Data Base", *IEEE Trans. Soft. Engin.*, **7**, 78-84, 1981.

[17] E.G. Coffman, Jr., H.O. Pollak, E.Gelenbe, and R.C.Wood, "An analysis of parallel-read sequential-write systems", *Performance Evaluation* 62-69, 1 (1981).

[18] C.A. Courcoubetis and M.I. Reiman, "Optimal Control of a Queueing System with Simultaneous Service Requirements", *IEEE Trans. Automatic Control*, **32**, 1987.

[19] G. Coulouris, J. Dollimore and T. Kindberg. "Distributed Systems: Concepts and Design", Addison-Wesley, 1994.

[20] F. Cristian, R. de Beijer and S. Mishra, "A performance comparison of asynchronous atomic broadcast protocols", *Distributed System Engineering*, 177-201, 1(1994).

[21] S. B. Davidson, H. Garcia-Molina and D. Skeen, "Consistency in Partitioned Networks", *Computing Surveys*, Vol. 17, No. 3, 341-370, September 1985.

[22] A. Demers, M. Gealy, D. Greene, C. Hauser, W. Irish, J. Larson, S. Manning, S. Shenker, H. Sturgis, D. Swinehart, D. Terry and D. Woods, "Epidemic Algorithms for Replicated Database Maintenance", *Technical Report*, Xerox, Palo Alto Research Center, CSL-89-1, January 1989.

[23] A.R.Downing, I.B. Greenberg, and J.M. Peha. "OSCAR: A System for Weak-Consistency Replication", *Procs. Workshop on Management of Replicated Data*, 26-30, 1990.

[24] P. B. Danzig, Dante DeLucia, and K. Obraczka. "Massively Replicating Services in Wide-Area Internetworks" *Technical Report, USC-CS-94-595*, University of Southern California, Los Angeles.

[25] D.K. Gifford, "Weighted Voting for Replicated Data", *Procs. 7th Symp. on Operating System Principles*, Pacific Grove, 150-161, 1979.

[26] Richard A. Golding. "Weak consistency group communication for wide-area systems", *Proceedings of the Second Workshop on the Management of Replicated Data*, 13-16, November 1992.

[27] Richard A. Golding. "The timestamped anti-entropy weak-consistency group communication protocol", *Technical Report UCSC-CRL -92-29, Computer and Information Science Board*, University of California at Santa Cruz, 1992.

[28] Richard A. Golding and D.D.E. Long. "Simulation Modelling of Weak-Consistency Protocols", *Proc. of International Workshop on Modelling analysis and simulation of computer and Telecommunication systems*, San Diego, 223-38, 1993.

[29] Richard A. Golding and D.D.E. Long. "Modelling replica divergence in a weak-consistency protocol for global-scale distributed data bases", *Technical report UCSC-CRL-93-09*, Computer and Information Sciences Board, University of California, Santa Cruz, February 1993.

[30] Richard A. Golding, "Weak-consistency group communication and membership", PhD Thesis, University of California, Santa Cruz, 1992.

[31] J. Gray, P. Helland, P. O'Neil and D. Shasha. "The dangers of replication and a solution", *SIGMOD'96*, Montreal, Canada, 173-82, 1996.

[32] J. N. Gray, R. A. Lorie, G. R. Putzolu and I. L. Traiger, "Granularity of Locks and Degrees of consistency in a shared database", *Technical Report*, IBM Research, September 1975.

[33] S.Jajodia and D.Mutchler. "Dynamic Voting", *Procs. of the ACM SIGMOD International Conference on Management of Data*, 227-38, June 1987.

[34] P. Jalote, "Fault Tolerance in Distributed Systems", *PTR Prentice Hall*, 1994.

[35] V.G.Kulkarni and L.C.Puryear, "A Reader-Writer Queue with Reader Preference", *Queueing Systems*, **15**, 81-97, 1994.

[36] V.G.Kulkarni and L.C.Puryear, "Stability and Queueing Time Analysis of a Reader-Writer Queue with Alternating Exhaustive Priorities", *Queueing Systems*, **19**, 81-103, 1995.

[37] L. Kleinrock, "Queueing Systems, Volume I: Theory", *John Wiley and Sons, Inc.*, 1975.

[38] P. J. B. King, "Computer and Communication Systems Performance Modelling", *Prentice Hall International (UK) Ltd.*, 1990.

[39] R. Ladin, B. Liskov, and L. Shira. "Lazy Replication: Exploiting the semantics of distributed services", *Proc. of the 9th ACM Symposium on Principles of Distributed Computing*, 43-57, Quebec City, CA, Aug. 1990.

[40] D.D.E. Long, "Analysis of Replication Control Protocols", *Procs. Workshop on Management of Replicated Data*, 117-121, 1990.

[41] D.D.E. Long and J.F. Paris, "On Improving the Availability of Replicated Files", *Procs. 6th Symp. on Reliability in Distributed Software and Database Systems*, Williamsburg, 77-83, 1987.

[42] B. W. Lampson, "Designing a Global Name Service", *Proc. of the fifth annual ACM symposium on principles of distributed computing*, Calgary, Alberta, Canada, 1-10, August, 1986.

[43] M. Little and D. McCue, "The Replica management system: a scheme for flexible and dynamic replication", *Proc. of the 2nd Workshop on Configurable Distributed Systems*, Pittsburg, March 1994.

[44] D. McCue and M. Little, "Computing Replicas Placement in Distributed Systems", *Proceedings of the IEEE Second Workshop on Replicated Data*, Monterey, 58-61, November 1992.

[45] E. D. Lazowska, J. Zahorjan, G. S. Graham and K. C. Sevic, "Quantitative System Performance" *Printice-Hall, Inc.*, Englewood Cliffs, New Jersey, 1984.

[46] C. Ma, "Designing a universal name service", PhD Thesis, Computer Laboratory, Cambridge University, 1992.

[47] D. Mitra and P.J. Weinberger, "Probabilistic Models of Database Locking: Solutions, Computational Algorithms and Asymptotics", *JACM*, **31**, 855-878, 1984.

[48] I. Mitrani, "Modelling of computer and Communication Systems", *Cambridge University Press*, 1987.

[49] I. Mitrani, "Simulation techniques for discrete event systems", *Cambridge University Press*, 1987.

[50] I. Mitrani and R. Chakka, "Spectral Expansion Solution for a Class of Markov Models: Application and Comparison with the Matrix-Geometric Method", *Performance Evaluation*, **23**, 241-260, 1995 .

[51] I.Mitrani and P.J.B. King, "Multiprocessor Systems with Preemptive Priorities", *Performance Evaluation*, **1**, 118-125, 1981.

[52] I.L.Mitrany and B.Avi-Itzhak, "A many-server queue with service interruptions", *Technion*, Israel Institute of Technology, Haifa, Israel.

[53] M. Misra and I. Mitrani, "Analysis of data replication with two levels of consistency", *2nd annual IEEE International Computer Performance and Dependability Symposium*, Urbana-Champaign, Illinois, 230-39, 1996.

[54] M. Misra and I. Mitrani, "Evaluation of update algorithms for replicated data", 3rd CaberNet Plenary workshop, Rennes, 1997.

[55] H. Garcia-Molina, "The future of data replication", *Fifth Symp. on Reliability in Distributed Software and Database Systems*, Los Angeles, California, 13-19, Jan. 1986.

[56] H. Garcia-Molina, "Elections in a distributed computing system", *IEEE Transactions on Comput.*, 48-59, Jan. 1982.

[57] D.Mutchler. "Some (naive?) Questions About Replica Control", *Procs. Workshop on Management of Replicated Data*, 113-116, 1990.

[58] R.D. Nelson and B.R. Iyer, "Analysis of a Replicated Data Base", *Performance Evaluation*, **5**, 133-148, 1985.

[59] D. C. Oppen and Y. K. Dalal, "The Clearinghouse: A decentralized agent for locating named objects in a distributed environment", *Technical Report*, Xerox, Office Products Division, OPD-T8103, Oct. 1981.

[60] J.F. Paris, "Voting with Witnesses: A consistency Scheme for Replicated Files", *Proc. Sixth International Conference on Distributed Computing Systems*, 606-12, May 1986.

[61] D. Saha, S. Rangarajan, and S. K. Tripathi, "An Analysis of the Average Message Overhead in Replica Control Protocols", *IEEE Transactions on parallel and distributed systems*, Vol. 7, No. 10, 1026-34, Oct. 1996.

[62] A. Sheth and M. Rusinkiewicz. "Management of interdependent data: Specifying Dependency and Consistency Requirements", *Procs. Workshop on Management of Replicated Data*, 133-136, 1990.

[63] B. Sengupta, "A Queue with service interruptions in an alternating Makovian Environment", *Operations Research*, 38, 308-318, 1990.

[64] W.Smith and P. Decitre, "An Evaluation Method for Analysis of the Weighted Voting Algorithm for Maintaining Replicated Data", *Procs. 4th Int. Conf. on Distributed Computing Systems*, San Francisco, 494-502, 1984.

[65] M. Stonebraker, "Concurrency Control and Consistency of Multiple Copies of data in Distributed INGRES", *IEEE Transactions on Software Engineering*, 188-194, May 1979.

[66] K. Thiruvengadam, "Queueing with Breakdowns", *Operations Research*, 11, 62-71, 1963.

[67] R. Thomas, "A majority consensus approach to concurrency control for multiple copy databases", *ACM TODS*, 3(3), 180-209, June 1979.

[68] P. Triantafillou, "High Availability is Not Enough", *Procs. 2nd Workshop on the Management of Replicated Data*, Monterey, 40-43, 1992.

[69] K. S. Trivedi, "Probability and Statistics with Reliability , Queuing, and Computer Science Applications", *Prentice-Hall, Inc., Englewood Cliffs, NJ*, 1982.

[70] Ada Waichee Fu, "Delay-Optimal Quorum Consensus for Distributed Systems", *IEEE Transactions on parallel and distributed systems*, Vol. 8., No. 1, 59-69, Jan. 1997.

[71] H.C.White and L.S. Christie, "Queueing with Preemptive Priorities or with Breakdown", *Operations Research*, 6, 79-95, 1958.

[72] L. Lamport. "Time, clocks and the ordering of events in a distributed system", *Comms. ACM*, Vol. 21, no. 7, pp. 558-65.