Performance Evaluation of Concurrency Control and Locking in Distributed Database Systems

By
Lu Jiazhen

A thesis submitted for the degree of Doctor of Philosophy
in Computer Science
in the Faculty of Science,
University of London

Department of Computer Science
University College London
1989

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

To My Parents

and

To My Husband

Acknowledgements

First of all, I would like to thank my supervisors, Professor C.H.C. Leung and Professor P.T. Kirstein for their guidance through the course of this work. In particular, I am in great debt to Professor C.H.C. Leung for many valuable discussions and important advices. I would like to thank Professor P.K. Kirstein for his valuable suggestion that the mathematic theory should be put into practice, which leads to the work of implementation in the thesis and for him to give me the opportunity to work in the department as a research assistant, which provides financial support for me to continue my Ph.D study.

I would also like to thank Professor S.R. Wilbur and Ms. K. Paliwoda for their many discussions on the implementation work. I am also indebted to British Council for providing me with the initial financial support.

Most of all, I would like to thank my parents and my husband, Guoxiong, for their continuous support and encouragement, without which the completion of the work would not be possible.

CONTENTS

LIST OF FIGURES

Abstract

This thesis studies the methods of evaluating the performance of centralized and distributed database systems by using analytic modeling, simulation and system measurement. Numerous concurrency control and locking mechanisms for distributed database systems have been proposed and implemented in recent years, but relatively little work has been done to evaluate and compare these mechanisms. It is the purpose of this thesis to address these problems.

The analytic modeling intends to provide a consistent and novel modeling method to evaluate the performance of locking algorithms and concurrency control protocols in both centralized and distributed databases. In particular, it aims to solve the problems of waiting in database locking, and blocking in concurrency control protocol which have not been solved analytically before. These models, which are based on queueing network and stochastic analysis, are able to achieve a high degree of accuracy in comparison with published simulation results.

In addition, detailed simulation models are built to validate the analytic models and to study various concurrency control protocols and distributed locking algorithms; these simulation models are able to incorporate system details at very low levels such as the communication protocols, elementary file server operations, and the lock management mechanisms.

In order to further validate the findings through measurements, an actual distributed database management system is specifically implemented which adopts the two phase commit protocol, majority consensus update algorithm, multicast communication primitives, dynamic server configuration, and failure recovery. Various performance measurements are obtained from the system such as the service time characteristics of communication and file servers, system utilization and throughput, response time, queue length and lock conflict rates.

The performance results reveal some interesting phenomena such as systems with coarse granularity outperform those with fine granularity when lock overhead is not negligible, and that the effect of the database granularity is small in comparison with the effect of the number of replicated copies. Results also suggest that centralized two phase commit protocol outperforms other types of two phase commit protocol, such as basic, majority consensus and primary copy two phase commit protocol under some circumstances.

Chapter1. Introduction

1.1 The Area of Performance Evaluation of Centralized and Distributed Database Systems

In recent years, distributed database (DDB) research has become increasingly important in the area of information processing because of the widespread use of local databases and the increasing availability of computer networks. The main goal of the distributed database management systems (DDBMS) is to manage logically correlated data across computer networks. The complexity of DDBMS is beyond that of traditional database and network in isolation. The problems of concurrently accessing and updating the logically correlated data as a whole, preventing inconsistent data manipulation, and providing crash recovery facilities are raised. A vast amount of research work has been done to solve the above problems, which leads to the extensive studies and implementations of the concurrency control and distributed locking algorithms. [45,46,47,48,44] However relatively little work has been done to provide an effective evaluation method to estimate and compare these algorithms.

Because of the size and complexity of the DDBMSs, it is not feasible to build many experimental systems to estimate and compare the designs and proposed algorithms. Thus the theoretical performance evaluation of DDBMSs becomes increasingly important in both the system design stage and algorithms comparison stage. As pointed out by Ferrari[22], "The importance of performance and its evaluation in all technical fields is obvious. Performance is one of the fundamental categories of attributes that are indispensable for the viability of any technical system. Computer systems are no exception to these rules. Studying their performance aspects is an essential and fundamental component of computer engineering." Despite the great demand in the performance evaluation area, relatively little work has been done to provide quantitative evaluations of DDBMSs. The performance evaluation of distributed database system involves a detailed estimation of the characteristic and functionality of the system architecture, namely, the concurrency control algorithms, the locking mechanisms, the network architectures, the database i/o structures and the statistical descriptions of all the components in the distributed database system. The goal of the evaluation is to provide a quantitative description of a distributed database system; thus it gives a useful guidance and active involvement into the system design and implementation, installation management, capacity plan formulation, and performance tuning and upgrading. The techniques involved in distributed database performance evaluation can be classified into three categories: analytic modeling, simulation and measurement. In analytic modeling, queueing network methods can be used to represent a DDB system as a network of queues. In simulation, the characteristic and functionality of various DDB systems can be simulated at a detailed level; thus providing a more accurate evaluation of the system. The simulation results can be used to verify the analytic results. In measurement, samples from on-line hardware and software monitors can be obtained to provide the statistical description of the service time

distributions of the system components and various mean measurements of the system. Measurements from an actual DDBMS can also validate analytic models. All the techniques have their place in the performance evaluation area.

Concurrency control is one of the most important aspects of DDBMSs. Its main function is to coordinate concurrent accesses to the databases distributed over different sites, while still guaranting data integrity. If the concurrent execution of a distributed transaction is equivalent to a serial execution of the transaction, the concurrency execution is said to be *serializable*. And if the concurrent execution can guarantee that a distributed transaction is either done completely or not done at all, the concurrent execution is then considered to be *atomic*. The concurrency control mechanisms should guarantee both serializability and atomicity of transaction executions. Numerous concurrency control algorithms have been proposed and some have been implemented for DDBMSs. These algorithms can be mainly classified into two categories: two-phase-locking and timestamps[8]. In two-phase-locking (2PL), they can be further derived into basic 2PL[8], primary copy 2PL[76], majority consensus 2PL[79], and centralized 2PL[3,24]. In timestamps, there are basic timestamps[7], Thomas timestamp generation rule[80], multiversion timestamps[65], and conservative timestamps[7].

The performance evaluation of concurrency control method aims to provide a theoretical treatment to various concurrency control algorithms proposed above and compare their superiorities. Because of the increasing complexity of the existing DDBMSs, the experimental comparison of concurrency control algorithms becomes infeasible. Therefore the development of effective analytic models and the construction of representative simulation models have a key role in evaluating the DDB performance. In analytic modeling, different concurrency control algorithms should be specified in the form of different mathematical models. How efficiently and accurately the model can represent the real situation depends on the effectiveness of the analytic methods and the accuracy of the theoretical definitions of the algorithms. An effective analytic model should provide a detailed specification of the algorithm in terms of its characteristics and functionalities. The detailed statistical descriptions of the system components obtained from real system measurements are also vitally important to the evaluation. Simulation approach is another method to evaluate the performance evaluation of DDBMSs. The major advantage of simulation over analytic modeling is that simulation can be used to model the real system at a more detailed level, thus giving a more accurate representation of the real system performance and providing a validation of the analytic models. However it requires a much higher cost in terms of cpu time and storage and sensitivity analysis is also difficult.

Locking is another important characteristic of DDBMSs. Distributed locking integrates the locking methods used in the centralized database and, furthermore deals with the problems of maintaining the serializability and atomicity of the DDB systems. The problem of evaluating the locking performance is complicated, because the nature of locking consists of both hardware contention and software contention; this means that transactions have to compete not only for hardware resources, like cpu, i/o, etc, but also for

data resources. The particular difficulty caused by data contention is that there are various different ways to compete for the data resources due to the complexity of the software itself. In data contention, two major collision resolution algorithms are used: the fixed waiting algorithm and the scheduled waiting algorithm. The fixed waiting collision resolution algorithm is to restart a blocked transaction after a fixed period regardless of the availability of the locks. The scheduled waiting algorithm is to free blocked transactions as soon as the required locks become available. Performance evaluation of locking aims to model these two algorithms.

In centralized databases, concurrency control is achieved by using two major locking mechanisms: the dynamic locking and static locking together with the introduction of two types of locks, namely exclusive and nonexclusive locks. Dynamic locking, as implied by its name, has the power to dynamically grant locks, and then to execute a transaction bit by bit whenever a lock becomes available. Upon conflicts, the transaction then has to wait until the conflicting locks are released while still holding the granted locks. Deadlock detection algorithms have to be used to detect and solve possible deadlocks caused by dynamic locking. In static locking, a transaction cannot start execution unless all the required locks are granted. Therefore there is no deadlock problem in this case. Various different scheduling mechanisms have been used in implementing the dynamic and static locking algorithms. The conflicting transactions can be scheduled to retry in several ways, such as waiting for a fixed interval, waiting for a random interval, or restarting as soon as conflict locks are released. All these different mechanisms will affect the system performance considerably; thus they have to be taken into account in the performance evaluation. Dealing with exclusive and nonexclusive locking is another major task in evaluating locking performance. The introduction of these two different types of locks enables databases to gain maximum concurrency, while still guaranteeing the serializability of the transaction execution.

In distributed databases, locking is achieved mainly by two major approaches, the distributed two-phase-locking and timestamps. Distributed two-phase-locking is performed by granting the locks at each site in the first phase, and releasing them in the second phase after transaction execution. While the timestamp mechanism is achieved by first identifying a specific serial order of the transactions, and then ensuring that serializable execution at each site is equivalent to the chosen serial order. The performance evaluation of distributed locking aims to define the problem and to develop a feasible method to solve it. The attention should be paid particularly to the distributed blocking phenomenon which is a new problem in the area. Distributed blocking is caused by lock conflict. Transactions are forced to wait in the blocked queue upon lock conflict. How to release the blocked transactions depends on the collision resolution algorithm used. Evaluating distributed blocking is a rather complicated task; thus has not been tackled so far.

The performance of DDB system depends on not only the characteristics of DDBMSs but also the architectures of the underlying computer network. Computer networks can have various different physical topologies, such as star, hierarchical, ring, completely interconnected, and irregular topology. They can

also be classified into local area and wide area networks. The communication delay and cost associated with them vary significantly from one another, and so does their performance. The performance also depends on the actual implementation of the communication protocols. As proposed by the International Standard Organization (ISO), protocols are decomposed into seven communication layers each of which performs its unique function. How and on which layer a protocol is implemented makes a significant difference to the performance and functionality of the system. The basic network performance parameters which affect the bearing DDBMSs are the network topology, the communication delay, the throughput, the resource capacity and utilization.

A distributed database system is composed of many local databases whose performances affect that of the whole DDB system. The performance of a local database management system depends on the architecture of the database schema and the data manipulation tools implemented for the particular database. A database schema is referred to as a logical database description. It defines the logical structure of the data and their relations; it thus provides clearly structured cross-references. There are three main database schemas: the relational model, network model and hierarchical model. The architecture of a schema determines the way of cross-referencing at the conceptual level. At the physical level, data are actually stored on the secondary storage device, such as drum, disk and tape. The main objective of physical organization is to provide fast access to the database with the minimum storage cost. The basic file system organization has the following file structures; pile, sequential file, indexed-sequential file, indexed file, direct file, and multi-ring file. The performance evaluation of these file organizations should reflect the characteristics of file manipulations and provide quantitative measures and comparison of the performance characteristics for different types of files.

The performance of database systems also depends on the computer systems. The time sharing operating system which schedules the execution of a transaction may decompose the computation into a number of distinct processes. Processes are the basic computational units managed by an operating system. The various processes of a single computation may require different units of the computer's hardware and may be executed in parallel. The scheduling of processes and their computations determine the performance of database system built upon the operating system. The evaluation of the commonly available operating systems and their power in data processing is necessary.

In conclusion, the main aims of this thesis are

1. to study the effect of locking algorithms in both centralized and distributed databases;
2. to study and develop systematic analytic methods to evaluate the concurrency control algorithms in distributed databases and compare the performance of different concurrency control algorithms;
3. to validate the analytic model by simulation; and

4. to validate the analytic model by implementing and measuring an actual distributed database system.

1.2 Current Research in the Area

1.2.1 Analytic Modeling of Centralized Databases

In general, the evaluation of centralized databases has been better studied than that of distributed systems. In particular, the concurrency control and locking mechanisms of centralized databases have been treated extensively. The locking mechanism can be mainly classified into static two phase locking, dynamic locking and conflict oriented locking; while the collision resolution algorithms can be classified as fixed and scheduled waiting. Most of the performance models only deal with the problem of exclusive locking. The non-exclusive locking phenomenon is not well studied. Some studies cover the evaluation of access pattern and non-uniform file attribute. Some better known results achieved so far are given below.

Markov Chain Approach

Markov chain model has been used by Menasce, [61] Mitra [63] and Thomasian[81] to analyze the concurrency control algorithms in centralized databases.

Menasce and Nakanishi [61] proposed an analytic model to evaluate two different concurrency control mechanisms; the locking oriented (LOCC) and conflict oriented (COCC) mechanisms. They decomposed the analytic model into two levels. Level 1 is a computer system consisting of a cpu service centre and a number of parallel i/o service centres. At this level, the probability of a transaction succeeding in its conflict test is considered to be fixed and an approximation technique is used to obtain the average time a transaction spent in a computer system. The level 2 model substitutes the whole computer system by a number of parallel exponential servers with a known service rate. The resulting model was analyzed using a truncated Markov chain model to analyze the detailed concurrency control mechanisms. This establishes a system of non-linear equations which could be solved interactively. In their model, the lock tables are assumed to be kept in main memory, therefore the time needed to check or release locks is considered to be negligible and all locks are exclusive. They also assumed that transactions arrive according to a Poisson distribution, the service time of a transaction at a cpu or i/o device is exponentially distributed, and for database manipulation, read set is equal to write set, and access to a database has a uniform distribution. Their analytic results were compared with simulation results. The range of error is around 10%. Their analytic model is used to estimate the average response time affected by different database size, different read/write sized, different number of parallel i/os, etc. They also compared the performance of COCC with LOCC under two different conditions and concluded that LOCC has a better performance than COCC in both cases.

Mitra and Weiberger [63] introduced a Markov chain probabilistic model of database locking. Their model intends to give the exact formulae for equilibrium database locking performance. It models the

interference phenomenon of locking with the assumption of blocked transactions being cleared. The locking graph of transactions is aggregated into a coarser definition of state in order to form the corresponding Markov process. The Markov process has a product form in the equilibrium distribution which is constituted from $O(N^p)$ terms, where N is the number of items in the database, and p is the level of multiple programming of transactions. Mitra and Weiberger used an algorithm to reduce the computation of the resultant Markov process from $O(N^p)$ to $O(Np)$ multiplications. Mitra and Weinberger made the following assumptions that locking is static; blocked transactions are cleared without retry; interarrival process is Poisson and for exact result, the service time distribution is assumed to be exponentially distributed. Mitra and Weinberger's model is used to estimate the mean concurrency and throughput of the system, and the probability of nonblocking is a function of total offered traffic. Mitra and Weiberger's model is the first one to find the exact analytic results of locking performance. With the introduction of the numerical algorithm to reduce the $O(N^p)$ calculation into $O(Np)$, the model is considered to be feasible to deal with the locking problem, although it is still rather expensive. The assumption of blocked transactions being cleared restricts the application of the model. Although an approximate solution to the retry case is given, it seems that the approach is much simplified. The model also excludes the lock request and release service time. The only service time included is the transaction processing time which is assumed to be exponentially distributed.

Thomasian [81] has studied the performance of static two phase locking using both Markov chain analytic model and simulation model. The main difference of Thomasian's work from others in modeling static locking is that the execution states of the system are derived from the lock request table, i.e. the lock conflict model is deterministic rather than probabilistic. Therefore the states of the correspondent Markov process are derived from the predefined lock request table. In order to simplify the complexity of the model, he used a hierarchical decomposition method to analyze the system and replace it with a composite queue with exponential service times. Thomasian used an approximation method to obtain the throughput of the system up to the point of the maximum level of concurrency for transactions in order to reduce the infeasible calculation of the Markov chain. The resultant throughputs are then used in conjunction with a one dimensional birth-death process to analyze the mean performance characteristics of the system. Thomasian assumed that lock request table and transaction classes are predefined, i.e. they have deterministic service time distributions; computer system is only characterized by its throughput in processing various combinations of transactions; interarrival process is Poisson; service time of the cpu and i/o compound system is exponentially distributed; and all jobs share a service with a FCFS discipline; The performance results estimated by Thomasians model are the throughputs of different classes of transactions, the mean number of transactions in the system and mean response time of the overall transaction classes. These assumptions, especially the first two, can simplify the locking model to some extent and avoid some difficult problems involved in seeking a product form solution of queueing network or solving the computationally unfeasible Markov process. The assumption of deterministic lock request seems to restrict the applicability of the model, since lock requests are usually probabilistic rather than

deterministic. Furthermore, a computer system can not be accurately characterized only by its throughput. Its performance depends on the states and other parameters of the system such as the queueing discipline, the arrival and service rate and their distributions. Another drawback of the model is that the lock request time and its blocking time are not presented in the model; thus the locking phenomenon is not modeled accurately.

Probability Approach

A analytic method based on basic probability is used by Shum and Spirakis. [74] They developed an analytic model for both dynamic 2PL and general 2PL. For dynamic 2PL in a centralized database, they gave some worst case bounds of the probability of lock conflicts. The worst case bounds are compared with the computation solution of a Markov process describing the actual system. All the worst case bounds are below the actual value and some of them are far from actuate. The upper bound on the average number of restarts per transaction and average time of a transaction are also given for the dynamic 2PL case. Shum and Spirakis also built a model for the general 2PL. The model is formed on the basis of the wait-for graph under some assumptions, which is used to estimate the steady state rate of conflicts and deadlocks in the system. They concluded that the rate of deadlock is proportional to the average number of transactions in the system. In the model, Shum and Spirakis assumed that all locks are exclusive locks, each transaction locks the same number of data items and all data items are accessed uniformly. They also assumed that the input process is Poisson and service time distribution is exponential. The behavour of cpu and i/o is also simplified and substituted by a general service centre with service time exponentially distributed. Another major drawback of the model is that the results of the model depend on some operational parameters such as the mean number of free transactions in the system. Therefore the application of the evaluation is restricted to only the existing systems.

Queueing Network Approach

A queueing network method is used by Irani and Lin [31] to analyze the performance of different concurrency control algorithms in a database system. Two models were developed to investigate the effect of locking granularity on the performance of a database system. They used the BCMP queueing network model introduced by Baskett et al[6] to solve their problem. The problem of analytically modeling the waiting time for a blocked request is avoided in their model. They suggested users to use simulation or empirical measurement to estimate the waiting time. To simplify the problem, they also made several assumptions that the waiting time for a blocked transaction is constant, independent of the number of concurrent transactions and the locking granularity; the probability of lock conflict is inversely proportional only to the number of granules; and database storage access time is exponentially distributed. They gave some results of the resource utilization and throughput for both i/o unit and cpu. They also gave the percentage useful i/o(/cpu) and locking overhead i/o(/cpu). They concluded that coarse granularity performs better. The main disadvantage of Irani and Lin's model is that the evaluation has to involve simulation or empirical measurements, which is either time consuming or practically difficult. Moreover

the assumption made in the estimation can reduce the accuracy of the whole evaluation considerablely.

Flow Diagram Approach

A flow diagram method is introduced by Tay, Suri and Goodman[78,77]. The model uses only the steady state average values of the variables. It is designated to evaluate the performance of both static and dynamic locking, transaction blocking, multiple transaction classes, nonuniform access and transactions of the indeterministic length. The method is based on the steady state mean value analysis and has a straightforward and simple nature. The main drawback of the method is that the hardware contention of computer system is factored out. Thus the effect of competing hardware resources is not accurately evaluated.

We can summarize that All the analytic models of centralized databases[61,74,63,81,78,77,31] are based on the assumptions of Poisson interarrival process and exponential or constant service time distributions. The problem of evaluating multiple transaction classes is dealt with by Thomasian [81] and Tay[78,77]. However, in Thomasian's model the lock requests by multiclass transactions are deterministic rather than probabilistic. And Tay's multiclass model is based on the assumption of no hardware contention. The problem of blocking has been tackled by Irani[31] and Tay[78,77]. Irani uses simulation and empirical measurement to estimate the waiting time of blocking, which is either time consuming or practically difficult. Tay's method of evaluating the waiting time in the blocked queue is based on the assumption of no hardware contention, which reduces the accuracy of the evaluation. The blocking with scheduled waiting is not studied.

In the above models for centralized databases, the following problems need to be further studied. Firstly, in transaction blocking, the scheduled waiting collision resolution algorithm is widely used in database systems but relatively little work has been done to evaluate such algorithm. Secondly all the above models assume Poisson interarrival time and exponential server time distribution, but in real system it is usually not justified. Thirdly most of the above models evaluate only single class transaction and only two models have tried to evaluate multiclass transactions under stringent assumptions. Forthly none of the above models have estimated the effect of locking operation having priority over transaction execution operation, which is usually the mechanism implemented in most database systems. In this thesis, these problems will be tackled.

1.2.2 Analytic Modeling of Distributed Databases

The analytic methods for evaluating distributed database systems have not been well developed in general, due to the complexity of distributed concurrency control algorithms. The research in this area is still at its very early stage. Among the few developed models, most of them are based on basic probability evaluation or even deductive reasoning, such as Menasce, Garcia-Molina, Sevcik and Badal's models.

Probability Approach

Menasce and Nakanishi [62] have studied the performance of time stamps concurrency control algorithm based on the two phase commit protocol for distributed databases. They used a group of nonlinear equations to represent the relations between various parameters, such as probability of conflict, arrival rate, system utilization and response time. The equations were solved by using iteration methods. The analytic model was verified by simulation. They provided the performance results of the arrival rate, the probability of conflict and the size of the query change. The major assumptions made in Menasce and Nakanishi's model are: Poisson arrival rate; exponential service time distribution for cpu and i/o; fixed size read and write set; full data replication and constant transmission time. The assumptions of Poisson arrival rate and exponential service times do simplify the analytic model significantly, but nevertheless reduces the accuracy of the results. Moreover the model only deals with immediate restart upon conflicts, waiting is not studied. The assumption of full duplication also reduces the applicability of the model.

Garcia-Molina [25] used analytic model based on basic probability to estimate the performance of distributed database systems. The concurrency control algorithm he analyzed was a majority consensus method introduced by Thomas[80]. The underlying communication was built on a ring network. The analytic model used by Garcia-Molina is straightforward and iterative in nature. The results obtained by Garcia-Molina indicate that centralized concurrency control mechanism is better than the decentralized ones; both centralized and decentralized mechanisms perform well under light traffic but poorly under heavy traffic. The assumptions made in Garcia-Molina's model are no parallel communication, full data replication, predeclared data objects and update only transactions. The first assumption has significantly restricted the applicability of the model since most distributed database systems employ parallel communication as the means to obtain high concurrency.

Sevcik [72] used some simplified probabilistic formulae and a cumulative distribution function to evaluate the probabilities of conflicts and the maximum of parallel delay in isolation. The evaluation method is iterative in nature. It starts with assuming the probabilities of the exceptional events to be zero and goes on to evaluate each delay in isolation. The process is repeated until the successive estimates cease changing. Sevcik estimated five concurrency control algorithms, i.e. centralized 2PL, conservative T/O, aggressive T/O, distributed 2PL and basic T/O. Sevcik made the following assumptions: the distribution function of parallel delay is formed by negative exponential distribution; each transaction has only a single processing site and that is its original site and transactions access data items in a uniform fashion.

Deductive Approach

Badal [4] introduced a deductive reasoning approach to analyze the impact of concurrency control (CC) on distributed database systems. The evaluations show the possible dependency and relationship between two parameters, such as degree of interference vs. classes of CC mechanisms, degree of locality vs. degree of CC centralization, acquisition of data objects by transactions vs. degree of CC centralization and degree

of locality vs. classes of CC mechanisms. The deductive reasoning is obtained by observations. No formal analytic calculation is used.

Queueing Network Approach

A queueing network model is used by Sheth, Singhal and Liu [73] to analyze the effect of network parameters on the performance of distributed database systems. The delays in transmission channels of the long haul networks with star and completely connected topologies were estimated by using Jackson's queueing network method. The results show that the completely connected network topology outperforms the star network. The assumptions made in their analysis are Poisson interarrival rate; exponential service time; full data replication and neglected queueing delay in cpu. These assumptions very much simplify the analytic model, which enables the analytic model to use straightforward queueing network method introduced by Jackson. However the assumptions may not be justifiable for the environments with heavy load and multilevel architectures. In particular the last assumption which neglects queueing delay in cpu is usually not true in a real system.

In the above models for distributed databases, [4,25,62,72,73] evaluation methods need to be further extended to provide a systematic way to define a distributed database by a wide range of parameters rather than a few; to release those restrictive assumptions such as full data replication, exponential service time; Poisson interarrival time, etc. and to build a unified model to evaluate a wide variety of distributed concurrency control algorithms.

1.3 The Goal of the Thesis

The prime goal of the thesis is to introduce a consistent performance modeling method to evaluate various concurrency control mechanisms for both centralized and distributed database systems. Since most of the concurrency control mechanisms fall into the two phase locking (2PL) category (the other being time stamps), and the evaluation method of 2PL mechanisms in distributed databases is not well studied, we concentrate on providing a consistent method to evaluate the 2PL based concurrency control algorithms, such as basic 2PL, primary copy 2PL, majority consensus 2PL and centralized 2PL.

From the literature survey in the previous section we can conclude that in both centralized and distributed database systems the problem of solving waiting time of a blocked transaction is not studied by most of the researchers except Tay. But Tay's model of waiting in centralized databases is considerablely restricted by the assumption of no hardware contention in the system. We intend to solve this problem by introducing a waiting model which not only factors in the hardware contention but also considers the effect of priority of locking over transaction execution.

In almost all the proposed analytic models in the literature the interarrival process is assumed to be Poisson and service time distribution is exponential. This assumption very much simplifies the modeling technique. However it is not usually justifiable in a real computer system[54]. It is very desirable to lift this

restriction. An extended diffusion approximation method is used to model distributed databases with non-Poisson interarrival rate and non-exponential service time distribution.

In actual systems, locking operations usually have priority over transaction execution operations. However there is no available analytic modeling method so far in dealing with the problem efficiently due to the complexity of priority queueing. It is one of the purposes of the thesis to address and solve this problem.

The complexity of 2PL algorithm in distributed databases results in the lack of a systematic method to define the concurrency control algorithm. In all the researches given so far there is no formal way of defining the concurrency control algorithms. In the thesis we introduce a systematic method to formally define a concurrency control algorithm by using a communication flow matrix and an access pattern matrix. Important system parameters such as degree of data replication, degree of data locality, read-write ratio, etc. are represented as the components of the matrices. All the 2PL concurrency control algorithms can be modeled by this method in a consistent fashion.

The major weakness of most analytic models is the lack of key descriptive parameters, which is caused by imposing stringent assumptions such as full data replication, read-only or write-only, full communication connection, single class transaction etc to simplify modeling technique. In the thesis we aim to overcome this weakness by introducing as many descriptive parameters in the model as possible. Firstly the network topology is included as one of the descriptive parameters of the distributed database system. All types of topologies such as star, ring, mesh, bus and fully connected network can be modeled. Secondly the degree of data replication is included as an important parameter of the system. It varies from single copy to fully replicated distributed databases. Thirdly the read-write ratio is introduced in the model to monitor the effect of read and write. Fourthly locking granularity is studied by introducing two parameters: total number of lock granules and the mean number of locks required by each transaction in the system. Fifthly the restriction of transactions being single class is released by allowing multiclass transactions to be modeled. Finally we introduce the technique to define the service time distributions for various types of database operation, such as insert, delete, append, reorganize, etc.

Validation is a very important part of performance evaluation. In this work a simulation model for distributed database system has been built to verify the analytic model. The simulation model intends to prove the vital part of methods introduced in the analytic model. The results are very conclusive in nature. Moreover our analytic model for centralized databases is also well validated by the simulation results given by Ries and Stonebraker[69].

Validation by system measurement has not been well studied, which is a reason why performance evaluation methods have not been enthusiastically accepted. In this thesis we not only intend to prove that our analytic model is correct and applicable but also intend to show the reader how to use it and explain the results. An actual distributed replicated database system is implemented on several Sun workstations over

Ethernet (a but network) with basic 2PL and majority consensus 2PL concurrency control algorithms. An analytic model is built on the basis of the actual system and various system performances are estimated by the model. These analytic results are further verified by the results obtained by system measurements.

1.4 Thesis Organization

Chapter 2 gives the preliminaries of distributed database modeling. It is not simply a review of the previous researches but rather a coherent reorganization of a wide variety of system modeling techniques which can be applied to distributed database modeling. In addition the author has made some new contributions in extending the existing methods which are not available before. Major components of distributed database system, i.e. computer time sharing systems, computer networks and databases, are studied and their evaluation methods are presented. Furthermore a method to evaluate database-bound computer system with multiprogramming is introduced.

Chapter 3 studies the concurrency control model of centralized databases. The prime goal is to introduce a method to model the waiting time of the blocked transactions in both open and closed systems. Furthermore novel methods are introduced to deal with priority execution of lock operations, nonexponential service time distributions and multiclass transactions. The analytic results are validated using the simulation results given by Ries and Stonebraker[69].

Chapter 4 introduces the methods to model the concurrency control protocol for distributed databases and extends the modeling techniques introduced for centralized databases to distributed databases. A particular locking protocol, basic 2PL, is studied throughout the chapter. A systematic method is presented to define a concurrency control algorithm. Restrictions on interarrival process and service time distributions are released by applying and extending diffusion approximation to the queueing network theory. The analytic results are validated by a simulation model.

Chapter 5 shows the consistency and integrity of the modeling method introduced in chapter 4 by applying it to three popular two phase locking algorithms, i.e. primary 2PL, majority consensus 2PL and centralized 2PL. These results are compared and some useful conclusions are drawn at the end.

Chapter 6 gives an actual application of the analytic modeling techniques to a real system together with validating by system measurements. It introduces the implementation of a real-time distributed replicated database and measurements of the system. A performance model is built for this actual system, with particular emphasis on the novel analytic model to evaluate bus network. Various analytic results are verified by direct measurements.

Chapter 7 gives the conclusions of the thesis and future directions of this research. In appendixes, the list of notions, theorem derivation and evaluation of service time distributions are included. In the thesis, some formula derivations include detailed steps in order to provide easy reading.

Chapter 2. Preliminaries of DDB Modeling

2.1 Introduction

A distributed database system is composed of hardware components, software components and data components. The structure of the hardware, the mechanisms of the software and the pattern of data structure determine the overall system performance.

Data Components

are defined by the granularity and the structure of data. Users of a DDB system compete not only for hardware and software resources but also, more importantly, for data resources. For example, a write oriented transaction will have to exclusively lock the records; while a read operation requires non-exclusive locks to prevent others from writing the records at the same time. There are a wide variety of data locking algorithms used for both centralized and distributed databases. Typical examples are basic two phase locking (2PL), primary copy 2PL, majority consensus 2PL, and centralized 2PL.

Software Components

A distributed database management system is designed to support data transparency, transaction atomicity, serializability, concurrency and reliability for distributed transactions. Data transparency supports a global picture of the logical data stored at various local databases. Thus a user do not have to know the actual physical locations of the data; Transaction atomicity guarantees that a distributed transaction is either done completely or not done at all. The system will never be in an incomplete state; Serializability is defined as the correct sequence of executions when transactions are executed parallelly; Concurrency control is used to achieve transaction serializability with maximum parallelism and reliability is referred to as the ability to deal with failures, such as site failure, network failure etc. A typical implementation of distributed database management system consists of various software developments such as concurrence control, locking and recovery. The software components of the distributed database system are the programs which forms the distributed database management system. They are

- Transaction Manager (TM) which supports data transparency, concurrency control and recover for distributed transactions;
- Communication Manager (CM) which supports communications between distributed database managers;
- local Database Manager (DM) which manipulates the local database.

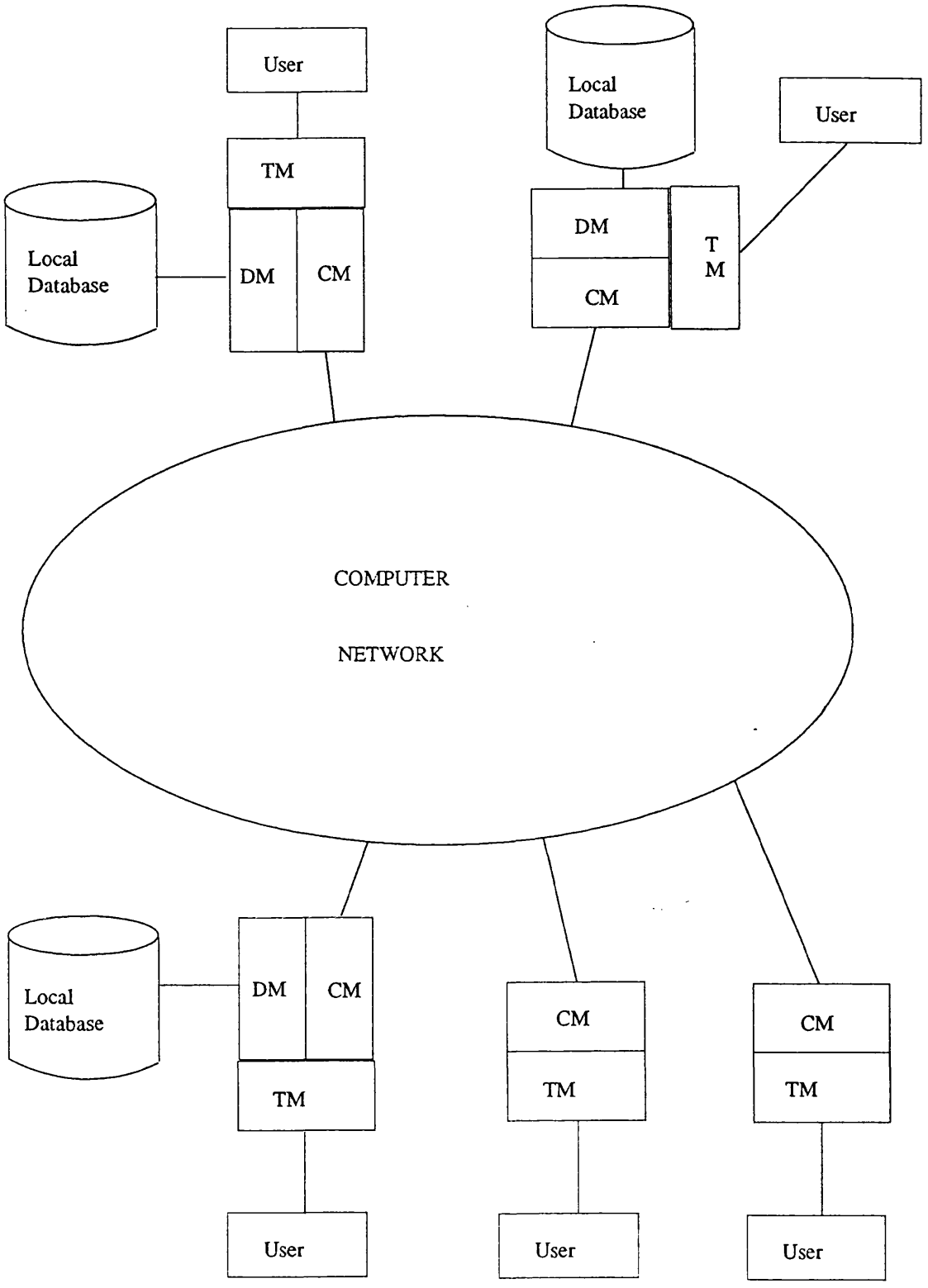


Figure 2.1. Architecture of a Distributed Database Management System

The architecture of a distributed database management system is shown in figure 2.1. In a typical implementation, a local database and its manager (DM) situate at the same site with the transaction manager (TM), although a DM and TM can be built separately where the communication between TMs and DMs is via the communication manager (CM).

Inside a TM there is always a data dictionary which keeps the global data schema and data allocation mapping information. In distributed database terms, the mapping is classified into the three layered schemas namely global schema, fragmentation schema and allocation schema. A global schema provides users with a logical overview of the data stored in the whole distributed database system. The users can read and update data without knowing their actual locations. A fragmentation schema is used to define the logical partitions of the global schema. An allocation schema is the actual mapping between the logical data partitions and their physical locations. With these three schemas stored in the data dictionary, a TM can therefore achieve data transparency. A TM can be seen as a transaction server which provides users with the global naming of data for reference and the tool to achieve concurrent and correct propagation of a distributed transaction.

At the lower level of the system a local database manager (DM) performs the corresponding data manipulations according to the instructions of the TM. There is a clear interface between a DM and a TM. The agreement about a distributed transaction is reached at the TM level and then data manipulations are performed at the DM level.

The communication manager (CM) is a piece of software for passing messages between TMs to control the concurrency of the distributed transactions. The efficiency of concurrency control protocol depends very much on the underlying communication networks.

Hardware Components

All the software functions described above will not work at all without the underlying hardware support. The hardware components of a distributed database system consist of the cpu, the file system and the communication network. The cpu and file system are essential for local database manipulations. The communication network is the foundation of the concurrency control protocols in distributed databases. Thus the structure and capacity of these hardware systems will have a strong impact on the performance of distributed database systems.

This chapter introduces the essential methods for hardware components evaluation. In section 2, the evaluation method for computer time sharing system is discussed. The three most widely used scheduling algorithms, i.e. the round robin, last-come-first-serve and batch processing are evaluated. Section 3 introduces the evaluation method for point-to-point network. Section 4 describes the database and file evaluation methods. In section 5, the queueing network models of database oriented computer system with multiprogramming are developed.

2.2 Computer Time Sharing System Evaluation

2.2.1 Introduction

With the rapid growth of information technology, computer systems become increasingly essential for information processing. This has led to that computer power becomes less expensive and more efficient nowadays. The access to computers has become easier and faster with the introduction of time sharing computer systems.

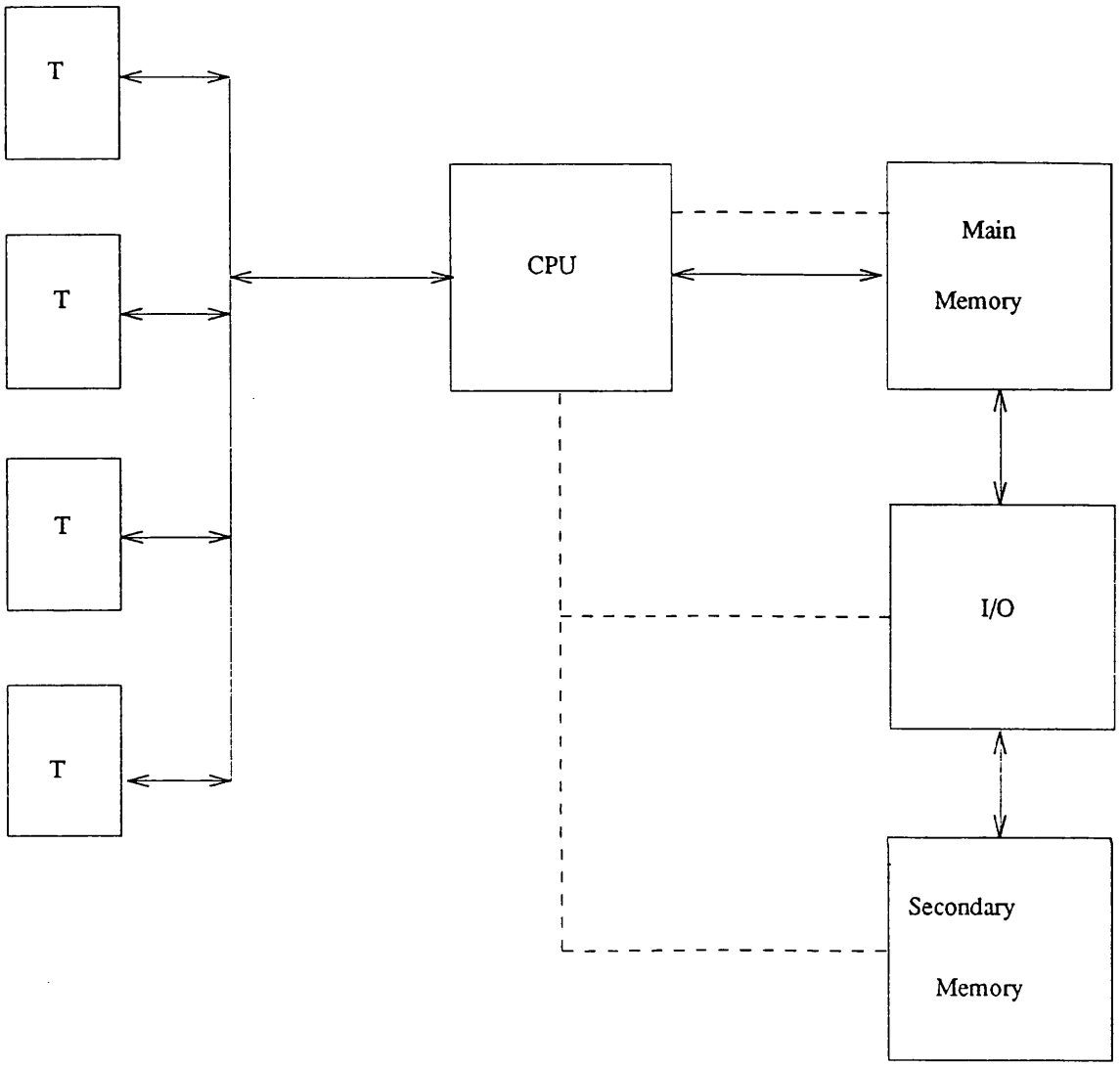
For a time sharing computer system, many users can access the same computer simultaneously through their own remote terminals. Each user goes through a thinking phase and a computing phase. Since users' thinking time is usually much longer than the computing time in each phase, a computer can serve many user's computing phases simultaneously without affecting the response times.

As shown in figure 2.2 a time sharing computer system consists of the following resources: remote terminals, communication links, main memory, central processing unit (cpu) and input/output devices which transfer data from secondary memory to main memory. The data generated by users at the remote terminals flow to the cpu via communication links. When receiving each user's request, the cpu will start computing. If data are required from secondary storage, the cpu will control the i/o device to transfer data from secondary memory to main memory. There is a control function to assign the cpu processing resource, the memory resource and the i/o resource to each user's request. Since the number of users can be very large, there will be inevitablely resource conflicts. Therefore the control function is actually the scheduling control over the resources. The resources are assigned to jobs according to some scheduling algorithms. The objectives of the scheduling control are to increase the efficiency of resource usage and be reasonable to users. For example, a scheduling algorithm may give smaller jobs priorities over larger jobs. The most well known scheduling algorithm is the so called round robin algorithm. In addition, the batch algorithm and the last-come-first-serve algorithms are also widely used. In the following sections the performance models of these scheduling algorithms are discussed.

2.2.2 Round Robin Scheduling Algorithm

The round robin scheduling algorithm is a very simple and robust algorithm. It always takes the first job in the queue; serves it for a fixed quantum of time; then returns it to the end of the queue; and starts to serve the next job at the beginning of the queue. Each job in the queue is served an equal quantum in one turn of the service. The cpu processing time for each job is divided by the number of jobs currently in the queue. So each job uses one nth of the cpu capacity, where n is the number of jobs in the queue. The structure of the round robin scheduling algorithm is show in figure 2.3.

Suppose that q_m is a quantum of cpu service time which a job receives at its mth return to service. The job arrives according to Poisson process with mean $1/\lambda$. The service time of a job is independent and identically distributed with distribution function given by $B(t)$ and mean service time given by $1/\mu$. The



denotes data flow, denotes control

Figure 2.2. Computer System Structure

utilization of the cpu service is given by $\rho=\lambda/\mu$. $\bar{N}(t)$ is defined as the number of jobs in the queue with attained service equal to t secs and $\bar{R}(t)$ is the average response time with attained service equal to t secs. It is due to Kleinrock[37] that

$$\bar{R}(t) = \frac{t}{1-\rho} \tag{2.1}$$

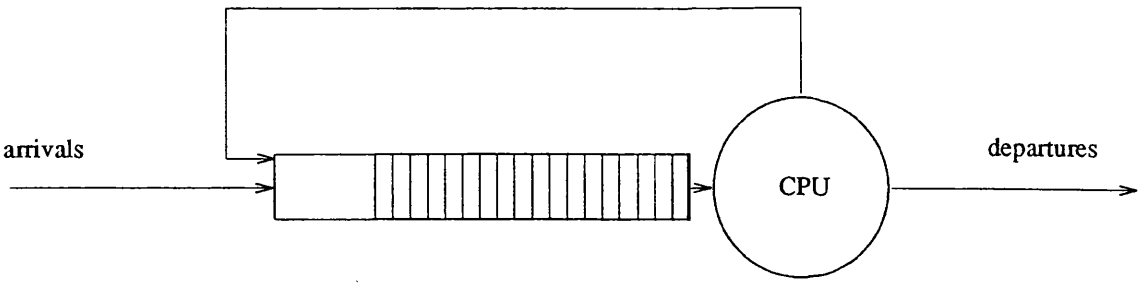


Figure 2.3. Structure of the Round Robin Scheduling Algorithm

$$\begin{aligned}\bar{N}(t) &= \lambda[1-B(t)]\bar{R}(t) \\ &= \frac{\lambda[1-B(t)]t}{1-\rho}\end{aligned} \tag{2.2}$$

This shows that the mean response time with the attained service of t secs does not depend on the service time distribution. From the above equations, the average response can be given by

$$\begin{aligned}\bar{R} &= \int_0^\infty \bar{R}(t)dB(t) \\ &= \frac{1}{\mu-\lambda}\end{aligned} \tag{2.3a}$$

and queueing length by

$$\bar{N} = \frac{\rho}{1-\rho} \tag{2.3b}$$

which are exactly the same as those of the $M/M/1$ queue.

For the case of exponential service time the distribution of response time with attained service of t secs has been derived by Coffman[13].

2.2.3 Last-Come-First-Serve Scheduling Algorithm

The last-come-first-serve (LCFS) scheduling algorithm, as the name implies, assigns the highest priority to the last arrived job. A newly arrived job always enters cpu immediately. After being served by cpu, it either goes back to the head of the queue upon requiring more service or leaves the system upon completing service. The structure of the model is shown in figure 2.4.

Here we consider a preemptive resume on LCFS scheduling algorithm in which a new job always preempts the job currently being served. The service will be resumed after the completion of the new job.

The mean response time and its Laplace-Stieltjes transform are respectively given by Kleinrock as[37]

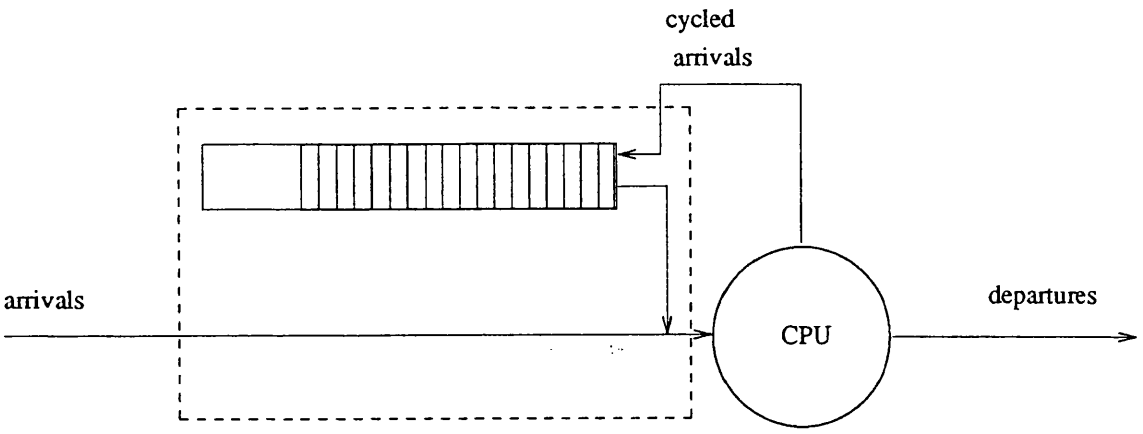


Figure 2.4. Last-Come-First-Serve Scheduling Algorithm

$$\bar{R}(t) = \frac{t}{1-\rho} \tag{2.4}$$

and

$$\bar{R}^*(s \mid t) = e^{-[s+\lambda-\lambda\mu^*(s)]t} \tag{2.5}$$

where $\mu^*(s)$ is the root of the following equation

$$\mu^*(s)=B^*(s+\lambda-\lambda\mu^*(s)) \tag{2.6}$$

and $B^*(s)$ denotes the Laplace-Stieltjes transform of $B(t)$.

It is surprising that the average response time of the LCFS algorithm is the same as that of the RR algorithm. $\bar{R}(t)$ is also independent of the service time distribution function $B(t)$. The unconditional average response time and queuing length are respectively

$$\begin{aligned} \bar{R} &= \int_0^\infty \bar{R}(t)dB(t) \\ &= \frac{1}{\mu-\lambda} \end{aligned} \tag{2.7}$$

and

$$\begin{aligned} \bar{N} &= \lambda\bar{R} \\ &= \frac{\rho}{1-\rho} \end{aligned} \tag{2.8}$$

It is easily seen that the mean measures of the RR and LCFS scheduling algorithm are the same as those of the $M/M/1$ queueing. It is pointed out by Baskett and Muntz[6] that the RR and LCFS scheduling model with general service time distributions having a rational Laplace transform can be indeed modeled by $M/M/1$ queue.

The RR, LCFS and $M/M/1$ service centres all have exactly the same marginal distribution while RR and LCFS service centres even allow multiclass jobs being modeled.

2.2.4 Batch Processing Algorithm

A batch processing algorithm is usually used in a computer system with no time sharing. A cpu always takes batch jobs according to the first-come-first-serve (FCFS) discipline. There is only one job being served at any instance of time. This batch processing can be easily modeled by the $M/G/1$ queue. The average response time and queueing length of the FCFS queue are respectively given by

$$\bar{R} = \frac{1}{2}\sigma\mu\frac{\rho}{1-\rho} + \frac{1}{\mu} \tag{2.9}$$

and

$$\bar{N} = \rho + \frac{\lambda^2\sigma}{2(1-\rho)} \tag{2.10}$$

where

$$\sigma = \int_0^\infty t^2 dB(t) \tag{2.11}$$

is the second moment of the service time.

2.3 Computer Networks Evaluation

2.3.1 Introduction

Computer communication networks play a very important role in distributed database systems. A network is a system which connects a number of geographically distributed computer systems or terminal systems together for data exchange. From the geographic point of view, a computer network can be divided into wide area network and local area network. A wide area network can be world-wide while a local area network can be only a few meters long.

A computer network connects a number of host computers and terminal concentrators into the communication subsystem. A host computer is used to provide computing services; while a terminal concentrator is used to join a group of user's terminals. Messages pass through the network between computers, as shown in figure 2.5.

The communication subsystem provides a common communication interface to all the network subscribers, i.e., host computers, terminal concentrators, etc. It contains all the communication functions which are essential for subscribers to communicate with each other. The major functions of the

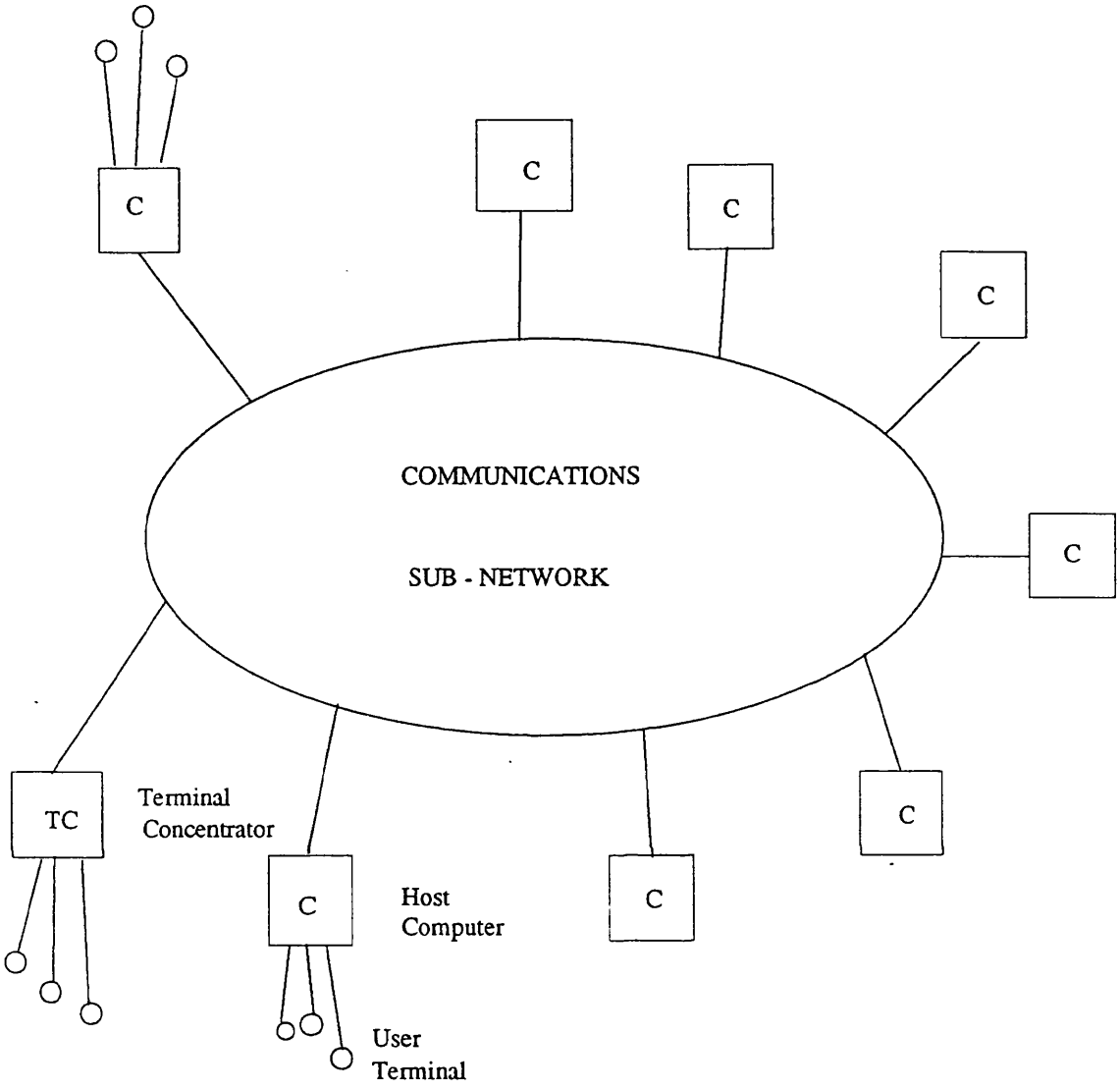
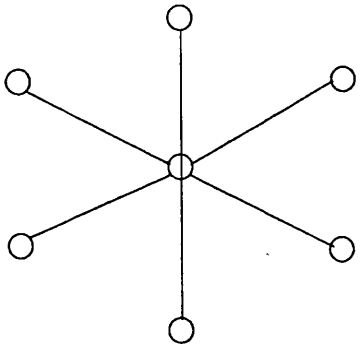


Figure 2.5. Structure of Communication Network

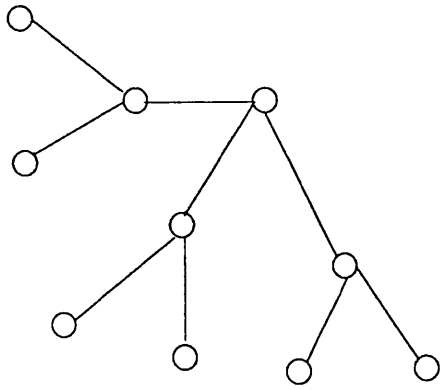
communication subnet are routing, flow control and reliable delivery.

The geographical structure of a communication subnet, which is usually called the topology of the network has the following forms: star, ring, tree, mesh and bus as shown in figure 2.6. The star, ring, tree and mesh network topologies assume that the subscribers of the network must communicate by a point-to-point link. This type of network is also called store and forward network. The bus and satellite structures allow broadcast type communications upon which any subscriber can communicate with all the others simultaneously. The cost of transmitting a message to all subscribers is exactly the same as to a single subscriber.

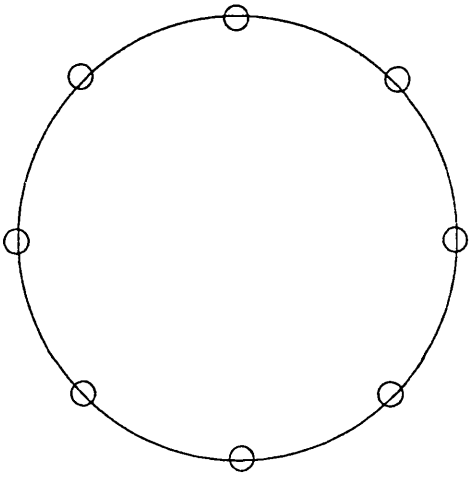
At the conceptual level, all communications are governed by communication protocols, i.e. the rules for communication procedure. The communication protocol defines precisely the way, the format and the



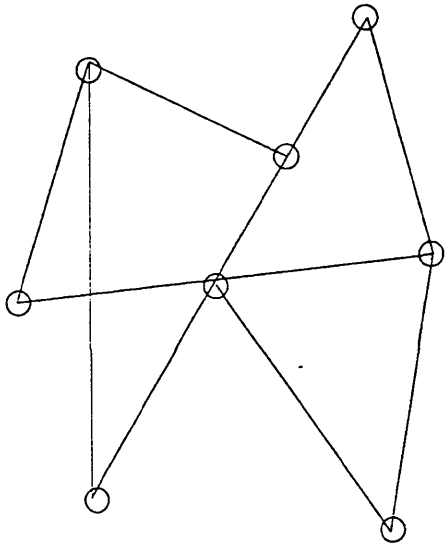
Star



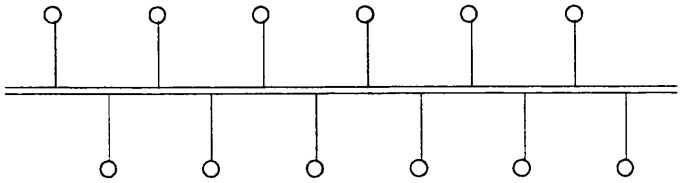
Tree



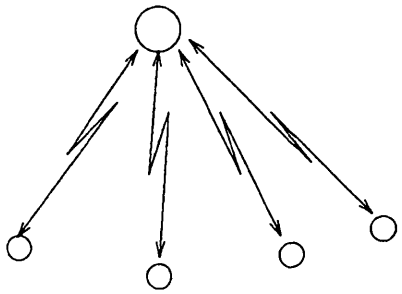
Ring



Mesh



Bus



Satellite

Figure 2.6. Network Topology

meaning of the data packet passed through the communication subnetwork. Because of the increasing complexity of the communication network, it is necessary to divide the network into several sub-systems with clearly defined interface between them. Each subsystem has its own function in the overall system. Therefore it requires its special protocol to fulfill this function.

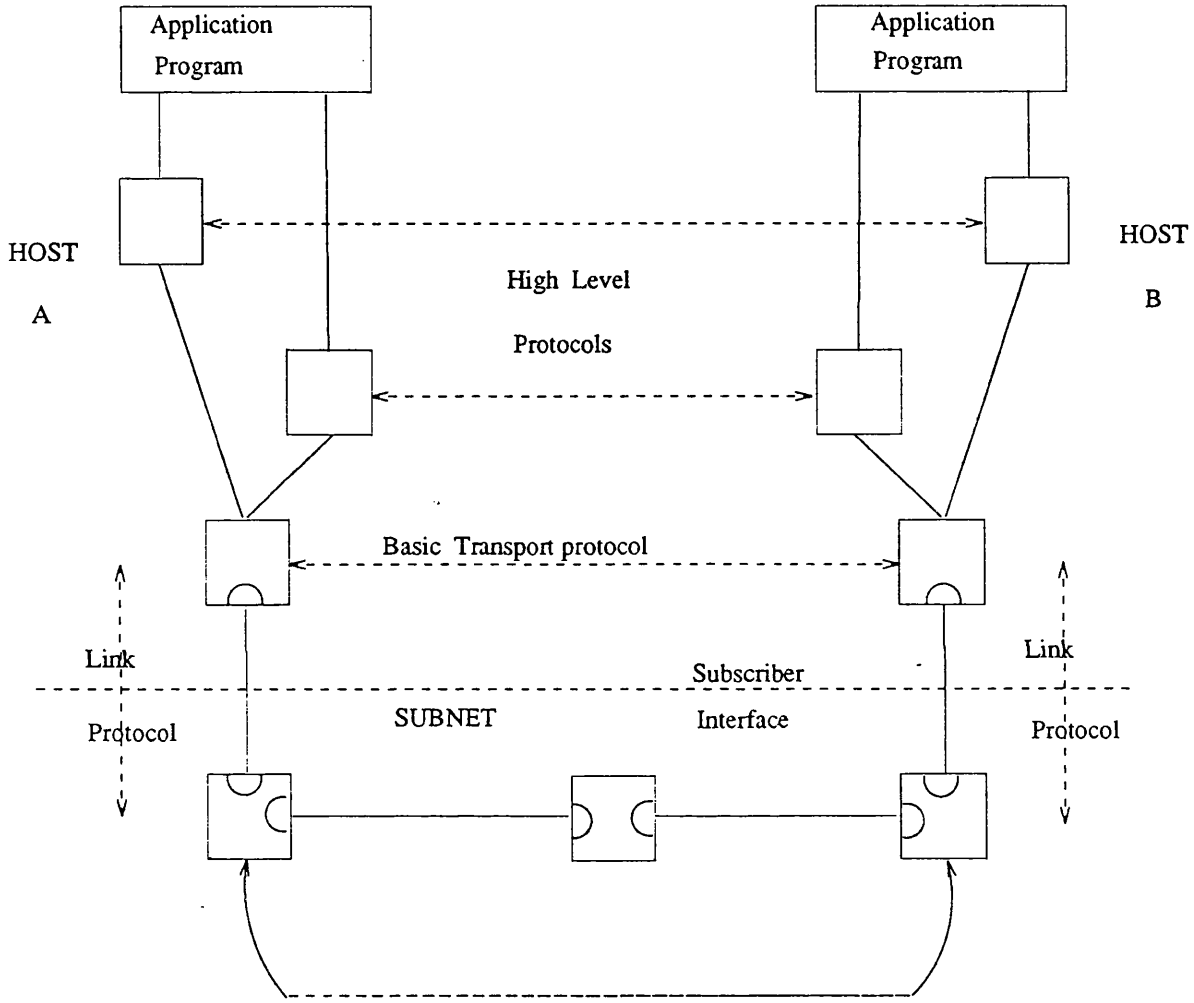


Figure 2.7. Inter-related Network Protocol

The basic hierarchies of protocols are shown in figure 2.7. Under the subscriber interface messages are passed over the transmission lines between switching centres by using the network end-to-end protocol. Acrossing the subscriber interface there exists a link protocol to carry messages across the interface without error. The subscriber interface is designated to transfer messages reliablely between hosts by using the basic transport protocol, which is a network independent protocol. All the hosts communicate with each other using the same transport level service without concerning about the underlying communication subnet. The transport level is a very good starting point for performance evaluation of the network, because

it is the basis of all the high level protocols. It clearly separates the communication functions and the user application functions. Therefore it is a natural boundary for performance evaluation.

The high level protocols are application dependent. With the establishment of the transport protocol, it is convenient to build various high level protocols, such as distributed data concurrency control protocol, file transfer protocol, remote job entry protocol, terminal protocol, etc. The distributed database concurrency control protocol and its performance will be studied in details in chapters 3, 4 and 5.

Store and Forward Packet Switching

Modern computer network techniques allow computers to be used to control the switch of messages. A computer system is thus responsible for switching messages by moving the addressed message from the input buffers to the destination output queue or moving a pointer to the buffer. As shown in figure 2.8, four channels are connected to the computer switch. Each channel has an input buffer and an output queue. There is also a longer-term message store used in case of overload.

We can now construct a computer network containing computer resources, terminal concentrators, multiplexers and store-and-forward packet switching centre as shown in figure 2.9. Traffic is generated from terminals and then routed to packet switching centres which provide all the store-and-forward message switching functions such as routing, reassembly, buffering, acknowledgement etc.

2.3.2 Model Definition and Evaluation

The objective of this section is to define the performance model for point-to-point computer network evaluation. The starting point for defining the model is to study the behavour of a store-and-forward message switching centre. As shown in figure 2.8, a message switching centre contains a pair of queues for each communication channel. The input queue is used for message buffering. The queue of interest here is the output queue, which stores and submits message for transmission in a FCFS queueing discipline. This phenomenon can be naturally considered as a FCFS queueing centre. Kleinrock introduced the network evaluation model under the assumption that message arrives to the ith channel according to a Poisson process with mean λ_i (msg/s) and the message length is exponentially distributed with mean l_i (bits)[36]. The ith channel capacity is defined as c_i (bits/s), the mean transmission time of a message at the ith channel is equal to $1/\mu_i = l_i/c_i$ (s). The switching centre with channel i is modeled as a $M/M/1$ system (i.e. a single server queue with exponential interarrival and service time) with arrival rate λ_i and mean service time $1/\mu_i$. The mean time of a message waiting in the switching centre is given by

$$\bar{R}_i = \frac{1}{\mu_i - \lambda_i} \tag{2.12}$$

and the mean queueing length is given by

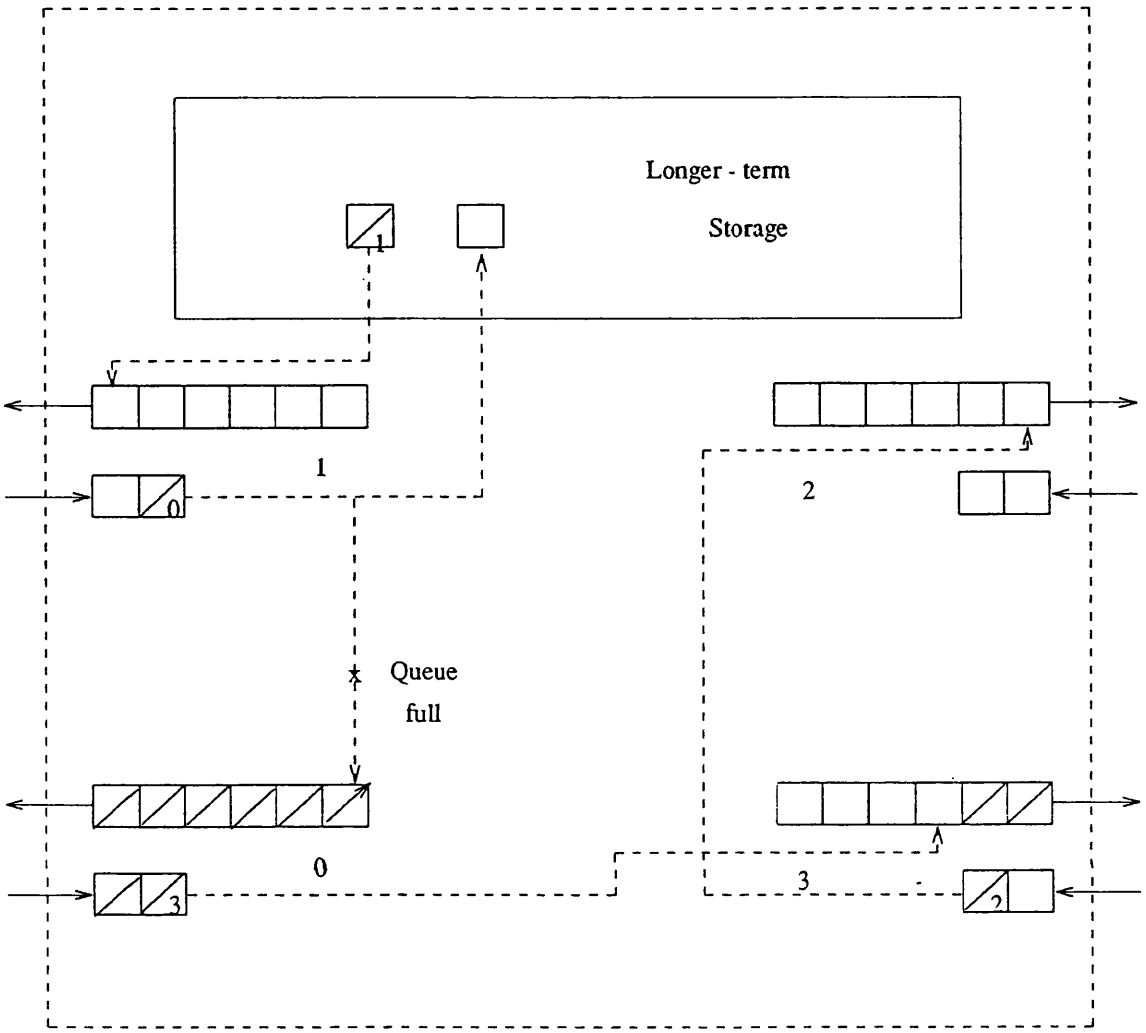


Figure 2.8. Store-and-Forward Queueing System of a Message Switch

$$\bar{n}_i = \frac{\rho_i}{1-\rho_i} \tag{2.13}$$

where

$$\rho_i = \frac{\lambda_i}{\mu_i}$$

For the M-channel, N-node model of the message switching communication network shown in figure 2.9, each store-and-forward message switching centre is considered as a node, and the path between any two nodes is considered as a channel which is modeled by an independent server. Suppose messages are generated from external sources in terms of a Poisson process with mean γ_{jk} for those originated at node j and terminated at node k. The overall message traffic is equal to

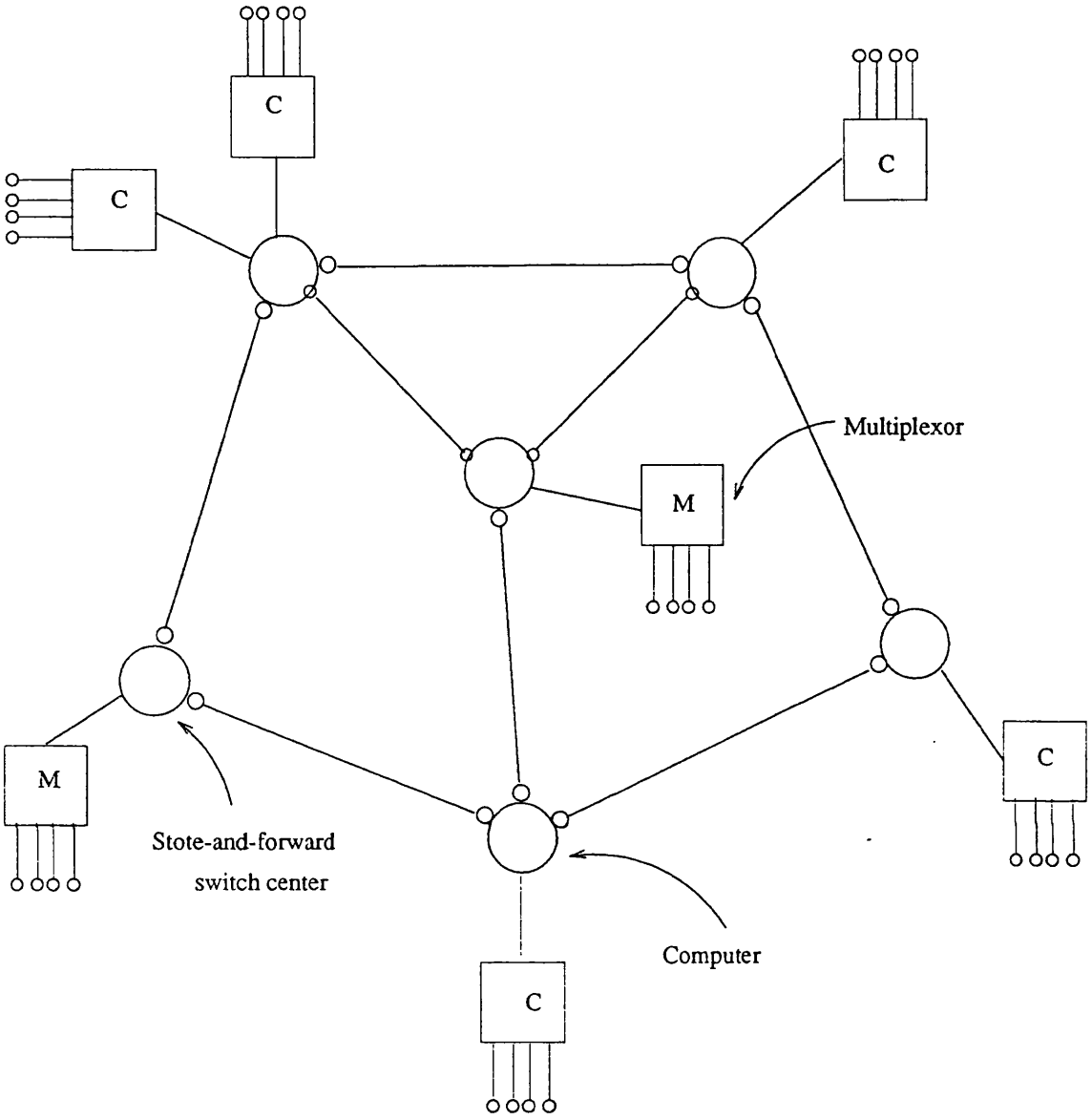


Figure 2.9. Computer Communication Network

$$\gamma = \sum_{j=1}^{N} \sum_{k=1}^{N} \gamma_{jk} \tag{2.14}$$

Kleinrock also derived the relation between λ and γ. Suppose the path between nodes j and k is denoted by π_{jk}. Any channel ch_i which is included in the path π_{jk} is represented as $ch_i \in \pi_{jk}$. The average number of messages passing channel ch_i is equal to the sum of the messages of all paths that traverse channel ch_i. That is

$$\lambda_i = \sum_{j,k : ch_i \in \pi_{jk}} \gamma_{jk} \tag{2.15}$$

A message originated at node j and terminated at node k has to traverse various communication channels along the path π_{jk}. The average delay for this message is therefore equal to the sum of the delays at all the traversed channels. That is

$$\sum_{i:ch_i \in \pi_{jk}} \bar{R}_i \tag{2.16}$$

The overall average message delay is obtained by

$$\bar{R} = \sum_{j=1}^{N} \sum_{k=1}^{N} \frac{\gamma_{jk}}{\gamma} \sum_{i:ch_i \in \pi_{jk}} \bar{R}_i \tag{2.17}$$

This may be rewritten by exchanging the order of the summations

$$\bar{R} = \sum_{i=1}^{M} \frac{\bar{R}_i}{\gamma} \sum_{j,k:ch_i \in \pi_{jk}} \gamma_{jk} \tag{2.18}$$

From equation (2.15), it becomes

$$\bar{R} = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} \bar{R}_i \tag{2.19}$$

The above method is suitable only for networks with Poisson arrivals and exponential services and can only be used to model the performance of a network in isolation. In order to model a distributed database consisting of not only computer networks but also databases, there is a need to introduce new modeling method to release the Poisson and exponential assumption. Instead of considering a communication channel as $M/M/1$ service centre, we can more accurately treat it as a $G/G/1$ service centre (i.e. a single server queue with general interarrival and service times). The whole computer network can be modeled as a queueing network where the communication channel between node j and k is modeled by a $G/G/1$ service centre with mean service time equal to $1/\mu_{jk}$. Thus the communication channels can be consistently modeled in the same way as other components of the distributed database system, such as cpu, i/o, etc. The method to solve the queueing network with general service time will be introduced later.

2.4 Database and File Systems Evaluation

In file system evaluation, previous research mainly concentrates on obtaining the mean access time relating to i/o hardware devices. However the access time depends on not only the characteristic of the i/o hardware devices but also the database file organizations. Moreover the mean measurement is not sufficient to represent the characterstics of the service time of i/o device. In the following sections we intend to provide the distribution functions of the access time for various database file organizations.

2.4.1 Secondary Storage Structure

Secondary storage is used to store large amount of data. It has slow access speed but large and cheap storage compared with main memory. Information systems such as databases depend heavily on secondary storage. The efficiency of secondary storage management will determine the performance of the database system built on top of it.

There are several types of secondary storage namely tape, disk and drum. The access time and storage cost are the main characteristics of the secondary storage. Generally speaking, the higher the access speed, the higher the cost. A computer system usually consists of several types of secondary storage in order to increase the efficiency of the system. Drum is used to store fast access data; disk has slower access speed but bigger storage, and tape has the slowest access speed which stores massive data such as backup files.

The most popular and efficient secondary storage is disk. A number of disks can be packed together to form a diskpack. Each disk in a diskpack has two sides. The sides at the top and the bottom of the diskpack are not used. A diskpack of 11 disks has 20 usable sides. On each side of a disk there are a number of circles called tracks. The tracks with the same radius on all the disks form a cylinder. Reading and writing on a disk cylinder involve moving the mechanical access arm to the cylinder and then rotating the diskpack to read or write data on the cylinder.

Drums are similar to diskpacks. They are constructed with a storage surface which forms one cylinder with a lot of tracks on it. Each track is associated with a head for reading and writing. There is no mechanical access arm movement like diskpack. Therefore the drum gives a good access speed but provides less storage since only one cylinder is used.

As the disk and drum are the most important secondary storages used by a computer in real-time, we will discuss them in detail and evaluate them in terms of queueing networks. The disk or drum service can be modeled by a service centre with FCFS queueing discipline. A job, requiring the service of a disk or drum will enter the service centre and demand i/o service. The service time of the job in the queueing centre is determined by the the access time of the disk or drum. We use $B(t)$ and $b(t)$ to define respectively the service time distribution and density function of the various i/o manipulations, such as seek, rotational latency, record and block transfer etc.

Seek Time

The seek time of a diskpack is the time spent in moving the access arm to the required cylinder. The initial startup time is denoted by s_c and the inter-track movement time by δ. If access arm has to move i cylinders before reaching the required position, the seek time is given by

$$sk_i = s_c + i\delta \tag{2.20}$$

If we consider a diskpack with n_c cylinders, the probability of the access arm traveling i cylinders is

$$p_i = \frac{2(n_c - i)}{n_c(n_c - 1)} \tag{2.21}$$

Thus the average seek time is given by

$$\begin{aligned} sk &= \sum_{i=1}^{n-1} p_i \cdot sk_i \\ &= \sum_{i=1}^{n_c-1} \frac{2(n_c - i)}{n_c(n_c - 1)} (s_c + i\delta) \end{aligned} \tag{2.22}$$

and the distribution function of seek time $B_{sk}(t)$ is given by

$$\begin{aligned} B_{sk}(t) &= Prob\{s_c + i\delta \le t\} \\ &= \sum_{k=0}^{i} p_k \\ &= \frac{2n_c i - i(i+1)}{n_c(n_c - 1)} \end{aligned}$$

Since $i = \frac{t - s_c}{\delta}$, we have

$$B_{sk}(t) = \frac{t - s_c}{\delta} \frac{2n_c - (\frac{t - s_c}{\delta} + 1)}{n_c(n_c - 1)}$$

$$= \frac{(t - s_c)(2n_c\delta - t + s_c - \delta)}{n_c(n_c - 1)\delta^2} \tag{2.23}$$

$B_{sk}(t)$ is the normal distribution function with discrete increasement at $t = s_c + i\delta$ $(i = 0, \ldots, n_c - 1)$.

Rotational Latency

For a diskpack the rotational latency is the time delay to allocate the right block after the track position is reached. This latency is mainly caused by the rotational delay of the diskpack. The time needed for one disk revolution is given by

$$2rl = \frac{60000}{rpm} (ms) \tag{2.24}$$

where rpm is the number of disk revolution per min.

Based on the assumption of uniform rotational latency within range $2rl$, the distribution function of rotational latency $B_r(t)$ is given by

$$B_r(t) = Prob\{latency \le t\} = \frac{t}{2rl} \tag{2.25}$$

Records and Blocking

Record is the unit for data storage at file or database level, while blocking refers to the way of fitting records into blocks. Records may have a fixed length or variable lengths. The length of records is denoted by Rec (char) and the length of blocks is denoted by Blk (char). If every record is fitted into only one block, the blocking is called unspanned; otherwise it is called spanned.

A disk consists of a number of blocks with gap Gap (char) between them. The blocking is best measured by the blocking factor Bfr introduced by Wiederhold[83], which gives the number of records expected within a block. For the unspanned disk, we have

$$Bfr = \left\lfloor \frac{Blk}{Rec} \right\rfloor \tag{2.26}$$

The waste of a disk is due to two factors: one is caused by the gap between blocks; the other is caused by unspanned blocking. They are denoted by W_G (char/record) and W_R (char/record) respectively. The waste space in one block due to unspanned blocking is half a record size on average; i.e. $\frac{1}{2}Rec/Bfr$. The total waste space W_a (char/record) is therefore equal to

$$\begin{aligned} W_a &= W_G + W_R \\ &= \frac{Gap}{Bfr} + \frac{1}{2}\frac{Rec}{Bfr} \end{aligned} \tag{2.27}$$

In order to derive the distribution of Bfr, the record length distribution has to be given. The record length distribution tends to be normal if the record length does not vary significantly (A normal distribution occurs when events are sampled independently with a fixed probability of occurrence; the record length falls into this category naturally). A normal density function has the form of

$$f_{record}(x) = f_{normal}(x) = \frac{1}{\sqrt{(2\pi)}\,\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.28}$$

where μ is the mean and σ is the standard deviation. Taking the record length as a example, μ is the average record length and σ is the variation range of the record length. Since the blocking factor Bfr is not proportional to the record length R, the distribution of $Bfr = \left\lfloor Blk/Rec \right\rfloor$ is distorted from the distribution of the record length. Let us denote the density function of Bfr as

$$f_{Bfr}(y) = Prob\{\mathbf{y} = \frac{Blk}{\mathbf{X}}\} \tag{2.29}$$

where x is the record length variable. With the variable transform $\mathbf{y} = \frac{Blk}{\mathbf{X}}$, the distribution function of

Bfr becomes

$$F_{Bfr}(y) = Prob\{y \le \frac{Blk}{x}\}$$
$$= \int_{\frac{Blk}{y}}^{Blk} f_{record}(x)dx \qquad (2.30)$$

Transfer Rate

The record transfer time for disk is given by

$$t_R = \frac{Rec}{t}(ms) \qquad (2.31)$$

where t (char/ms) is the raw transfer rate. Since the raw transfer rate t is a constant, the density function of the record transfer time is given by

$$b_{t_R}(y) = \frac{d}{dy}\int_0^{ty} f_{record}(x)dx$$
$$= t f_{record}(ty) \qquad (2.32)$$

The block transfer time is

$$t_B = \frac{Blk}{t} \quad (1/ms) \qquad (2.32)$$

Since Blk and t are both constants, the block transfer time is deterministic. That is

$$b_{t_B}(x) = \delta(t_B) = \begin{cases} 1 & \text{if } x = t_B \\ 0 & otherwise \end{cases} \qquad (2.34)$$

A random block access time includes seeking a sylinder, rotating to the track and reading a block, i.e.

$$t_{rB} = sk + rl + t_B \qquad (2.35)$$

Its density function can be derived by the convolution of density functions of the tree variables. That is

$$b_{t_{rB}} = b_{sk}(x) * b_r(x) * b_{t_B}(x) \qquad (2.36)$$

2.4.2 File Organizations in Databases

The important criteria for information storage in databases are fast access, convenient update and economic storage. Although there are virtually unlimited types of file organizations to meet these criteria. They can be easily classified into a few basic types. Here we discuss the most widely used file organizations, namely, sequential, indexed and direct files.

The basic quantitative descriptions of a file organization are the record length Rec (char); the file length $File$ (char) and the time needed to fetch an arbitrary record T_F (ms). The mean measures of the above qualitative descriptions are given by Wiederhold[83]. Here we intend to obtain their service time distribution functions, which will be used later.

2.4.2.1 Sequential File

In a sequential file organization records are ordered according to their keys. The key of a record can be a particular attribute or the combination of several attributes in the record. All keys are unique and comparable in a sequential file.

Since records are stored sequentially in the order of their keys, inserting new records in a sequential file is always difficult. The common technique to handle the insertion is to collect insertion transactions in an overflow log file. Once the log file becomes big, a batch updating is performed by reorganizing the sequential file.

Fetch Record

Fetching a record in a sequential file can be performed most efficiently by the binary search method[38]. The searching starts from the middle of the sequential file. If a record is not found, the searching continues to one of two equally partitioned parts according to the order of the key. The searching procedure continues recursively until the required key is found. The number of searches has a logarithmic function with respect to the number of blocks n_r/Bfr, where n_r is the number of records. The expected number of block accesses for the binary search is given by

$$\log_2\left\lceil\frac{n_r}{Bfr}\right\rceil$$

From equation (2.35), the overall binary fetch time is given by

$$T_F = \log_2\left\lceil\frac{n_r}{Bfr}\right\rceil(sk+rl+t_B) \tag{2.37}$$

The density function of t_F is

$$b_{t_F}(x) = b_{t_{rB}}\left\lceil\frac{x}{\log_2(n_r/Bfr)}\right\rceil \tag{2.38}$$

2.4.2.2 Indexed File

An indexed file provides the means to allocate a record by referring the index of a particular attribute of the record. The main advantage of the indexed file is that it can provide indexes for many attributes of a record, and therefore obtain fast and flexible access path from various attributes. The most well known and widely used indexed file organization is the B-tree file as shown in figure 2.10.

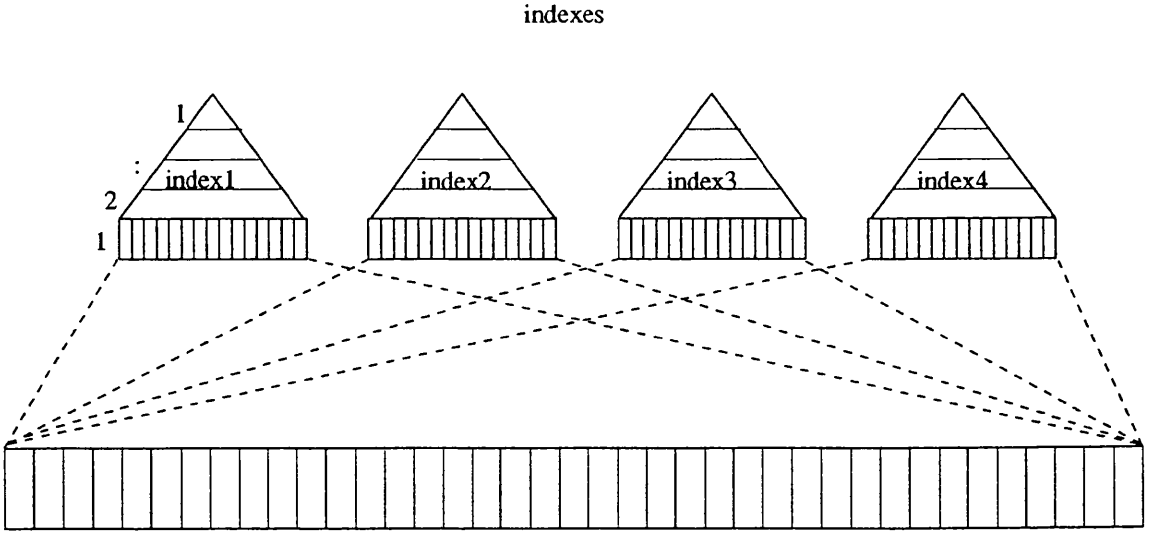


Figure 2.10. Indexed File Organization

The B-tree indexed file is composed of a number of index blocks with length y, defined as the number of index records in a block. The number of index records in an index block is always greater than $y/2$ and less than y. Once an insertion causes an index block to exceed y index records, the index block will be split into two with each block containing half of the index records. The index blocks at each level of the index can be split in the same way. So the splitting can be done recursively from bottom to top when necessary.

Yao[8-4] has pointed out that the effective number of index records in a block y_{eff} is between $y/2$ and y, and in a steady state of a B-tree file the effective fanout y_{eff} is given by

$$y_{eff} \rightarrow \ln 2 \cdot y = 0.69y \tag{2.39}$$

The number of levels of an index l is determined by the effective fanout y_{eff} and the number of indexable records n'_r. That is

$$l = \left\lceil \log_{y_{eff}} n'_r \right\rceil \tag{2.40}$$

Fetch Record

Fetching record in an indexed file involves accessing an index with l levels and accessing the actual record. We have

$$T_F = (l+1)(sk+rl+t_B) \tag{2.41}$$

with

$$b_{t_F}(x) = b_{t_B}(\frac{x}{l+1}) \tag{2.42}$$

If each index can be kept on one cylinder, the seek time is reduced. Thus

$$T_F = 2sk+(l+1)(rl+t_B) \tag{2.43}$$

with

$$b_{t_F}(x) = b_{sk}(\frac{x}{2})*\delta[(l+1)(rl+t_B)] = b_{sk}\left\lceil\frac{x-(l+1)(rl+t_B)}{2}\right\rceil \tag{2.44}$$

2.4.2.3 Direct File

A direct file is used when the addresses of records can be given, and the storage media can provide direct access facility. The addressing of record is obtained by arithmetic transformation of a key. The most well known technique is hashing. It transforms a key into a uniformly distributed address within the range of the direct file as shown in figure 2.11. The unit of addressing is called a bucket, which consists of one or more slots. Each slot can hold one record. The key-to-address transform may produce an identical address for different keys, which is called collision. Collision will occur frequently if the addressable file space is not big enough. One technique to solve this problem is to increase the free space in the direct file.

Upon inserting a new record into a direct file, a key to address transformtion is performed to generate a random address of a bucket. If there is no collision with this bucket address, the record is stored in the first slot of the bucket. If collision occurs and there is still some free slots in the bucket, the record is stored in the first free slot in the bucket. Otherwise the record has to be stored in a separate area called overflow file. An overflow file stores records in the form of a sequential chain. For the sake of simplicity, we only study the case of one slot per bucket.

Collision Probability

For a direct file with m slots and n records, the probability of j records being assigned to one slot is given by the binomial distribution[83]

$$P_c(j) = \frac{n!}{j!(n-j)!}\left(\frac{1}{m}\right)^j\left(1-\frac{1}{m}\right)^{n-j} \tag{2.45}$$

In the case of j records being assigned to a slot, it requires $j/2$ number of accesses on average to obtain one of the j records. Thus the average number of accesses for each record is given by

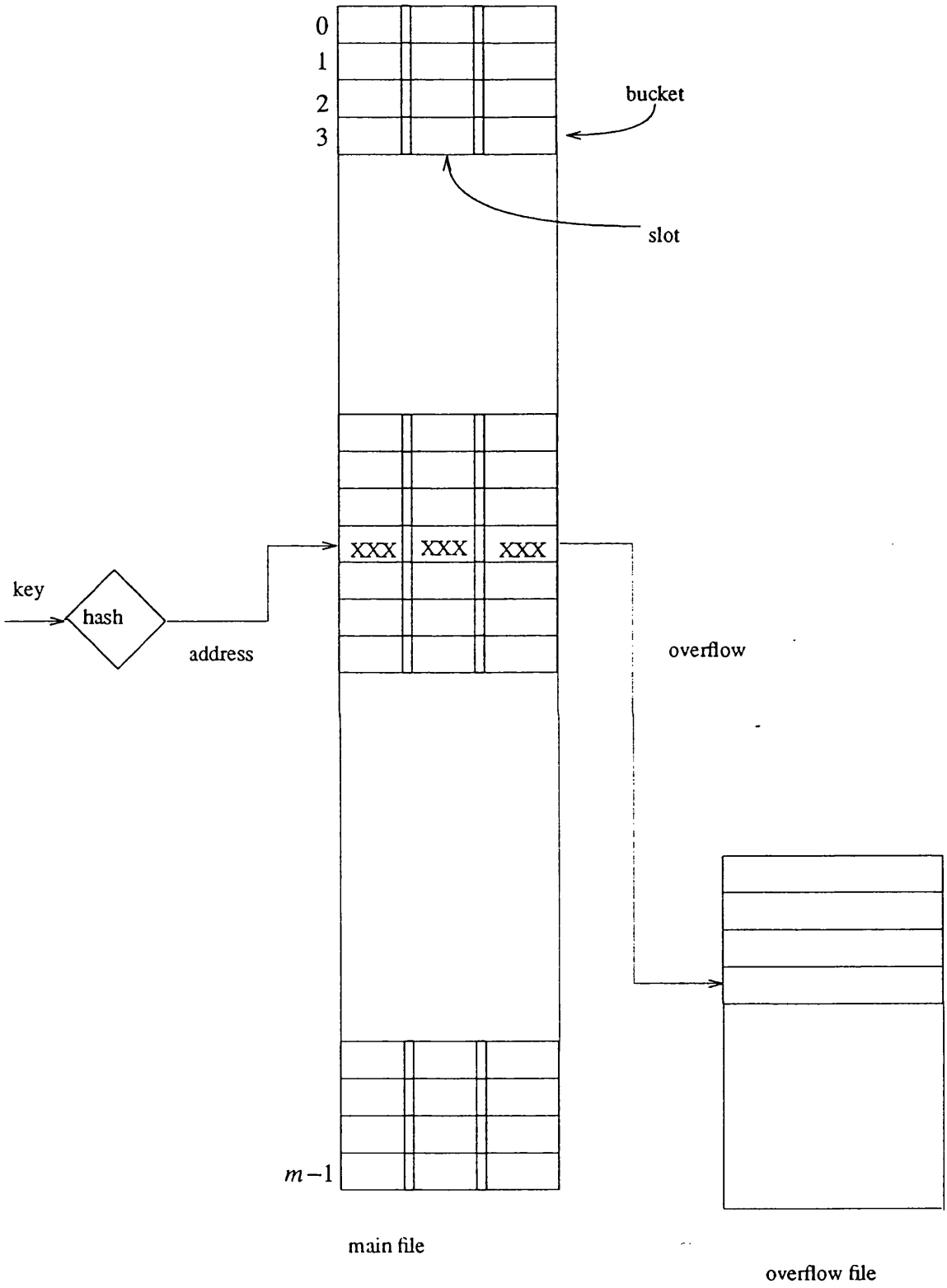


Figure 2.11. Direct File Organization

$$n_a = \sum_{j=0}^{n} \frac{j}{2} P_c(j)$$
$$= \frac{1}{2} \sum_{j=0}^{n} \frac{n-j+1}{m-1} P_c(j-1) \quad (2.46)$$

At $j=0$, $P_c(j-1)=0$ and $n \gg j$ and $m \gg 1$, it becomes

$$n_a = \frac{1}{2} \sum_{j=1}^{n} \frac{n}{m} P_c(j-1)$$
$$= \frac{1}{2} \frac{n}{m} [1-P_c(n)] \quad (2.47)$$

Since

$$P_c(n) = \left[\frac{1}{m}\right]^n \quad (2.48)$$

is the probability that all n records fall into one slot, which is a very rare event and can be neglected. Hence

$$n_a = \frac{1}{2} \frac{n}{m} \quad (2.49)$$

Fetch Record

Fetching record in a direct file involves an access to the direct address of the bucket and n_a number of accesses to the overflow chain. That is

$$T_F = (1+n_a)(sk+rl+t_B) \quad (2.50)$$

$$b_{t_F}(x) = b_{t_B}\left(\frac{x}{1+n_a}\right) \quad (2.51)$$

The derived distribution functions can be used to evaluate the service time distribution of database or i/o access in the following chapters.

2.5 Queueing Network Models of Database Oriented Computer System

In this section we shall discuss the performance evaluation models of computer systems with multiprogramming in virtual memory. The components of a computer system, i.e. cpu and i/o, which we have discussed in the previous sections, provide the basis for modeling a multiprogramming computer system. There are some available theoretical and empirical methods, such as Jackson's queueing network model[33], operational analysis[9] etc for the evaluation of multiprogramming computer system. But the results obtained so far are not sufficient to analyze the database oriented computer system. This is due to the

assumption that the cpu and i/o service times are exponentially distributed. This assumption is only approximately justifiable for cpu bound applications, but is not suitable for i/o bound database oriented applications. The latter means that queues are usually formed at the i/o devices rather than at cpu service centre. The distribution functions of i/o service time are proven more likely to be uniform than exponential. The aim here is to develop performance models for the database oriented multiprogramming computer systems.

2.5.1 Model Definition of Computer Systems

Modern computers very often employ virtual memory technique to manage data storage. The addressing space of the main memory is called physical addressing. An alternative to physical addressing is the use of relative addressing which covers the entire file domain maintained by secondary storage such as drum and disk etc. An application program is free from the actual location of the required data. Instead it only deals with the relative address of the data. The technique to load the required data from the secondary storage device into the main memory is called paging. The way of loading and discarding data in the main memory is called paging discipline. A page is the unit of data transfer.

A paged multiprogramming computer system has two main tasks: memory allocation and scheduling. Memory allocation by paging is to move address space required by a user into the main memory. Scheduling is to control an appropriate multiprogramming level by job scheduling algorithm. The purpose of controlling the multiprogramming level is to avoid deterioration of system performance when main memory is overloaded.

The two tasks of paged multiprogramming computer system can be naturally decomposed into two levels, i.e., job scheduling level to control multiprogramming and resource allocation level to control paging and computing.

At the higher job scheduling level a job from a user terminal is placed into the job scheduling queue waiting to be multiprogrammed. Only a fixed number of jobs, N, are allowed to be executed at the lower level of multiprogramming computer, where N is also the level of multiprogramming. The job scheduling discipline could be batch, round-robin, last-come-first-serve, etc as studied in section 2.2.

At the lower level, a job is always in one of the three states: ready, running and suspended. When the required address space is allocated by paging, the job is said to be in a ready state. The ready job is placed on the cpu scheduling queue to wait for execution. A job being executed in cpu is said to be in the running state. The running will be terminated until either the job's service is completed or a paging is required. The cpu execution will be sliced into small slots as studied in section 2.2.1. A job requiring additional page allocation is said to be in the suspended state. The suspended job is placed in the i/o device queue to wait for the allocation of the required page to the main memory. While a suspended job is waiting for i/o operation, the cpu is relinquished to process an other ready job in the cpu scheduling queue if there is any. Since a database-oriented computer system is most commonly i/o bound, queues will be expected to form

at the i/o devices.

We shall now develop the model for a multiprogramming computer system with virtual memory. Two levels of activities can be modeled by an outer model for the job scheduling queue and an inner model for the cpu and i/o queues. The inner model represents the multiprogrammed computer with virtual memory; while the outer model evaluates the control of multiprogramming level. They are illustrated in figures 2.12 and 2.13 respectively.

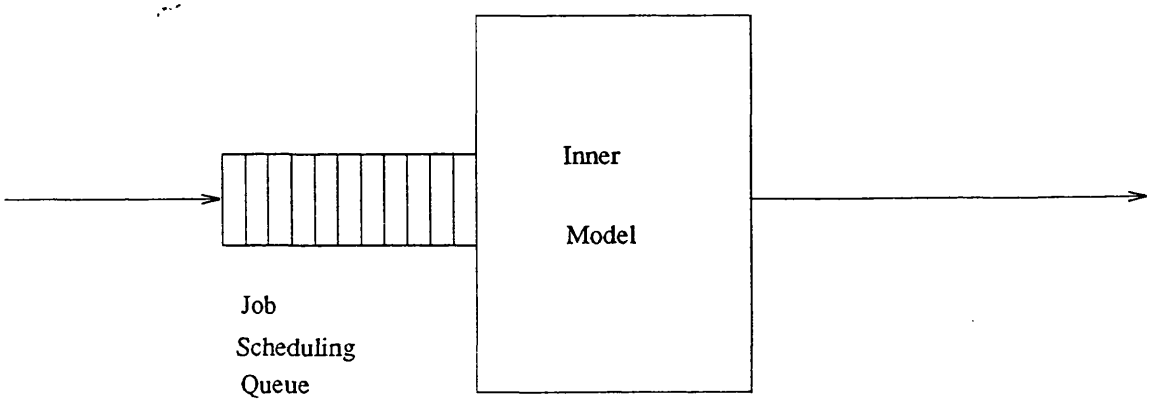


Figure 2.12. Outer Model of Multiprogramming Control

The inner model consists of one cpu service centre and m i/o service centres. All of them have the FCFS queueing discipline. Generally speaking the service time distributions of cpu and i/o devices are independent and identically distributed. They form a network of queues with a feedback from i/o service centres to the cpu service centre. The probability of feedback p_{fb}, determined by the mean number of paging activities of a job n_p, is given by

$$p_{fb} = \frac{n_p - 1}{n_p} = 1 - \frac{1}{n_p} \tag{2.52}$$

Usually $n_p \gg 1$; therefore $p_{fb} \rightarrow 1$. That means the number of activities between the service centres inside the inner model is much greater than that between inner and outer service centres.

The outer model resembles an aggregated service centre with whatever queueing discipline is used, i.e. batch, round-robin, last-come-first-serve etc.

2.5.2 Decomposition Approach with Exponential Service Times

A number of modeling methods have been introduced to solve the above multiprogramming computer system based on the assumption of exponential service time distribution. One well known method is the decomposition approach introduced by Courtois[16]. The method aims to decompose a complex system into a number of groups with the assumption that interactions among groups are much weaker than those within

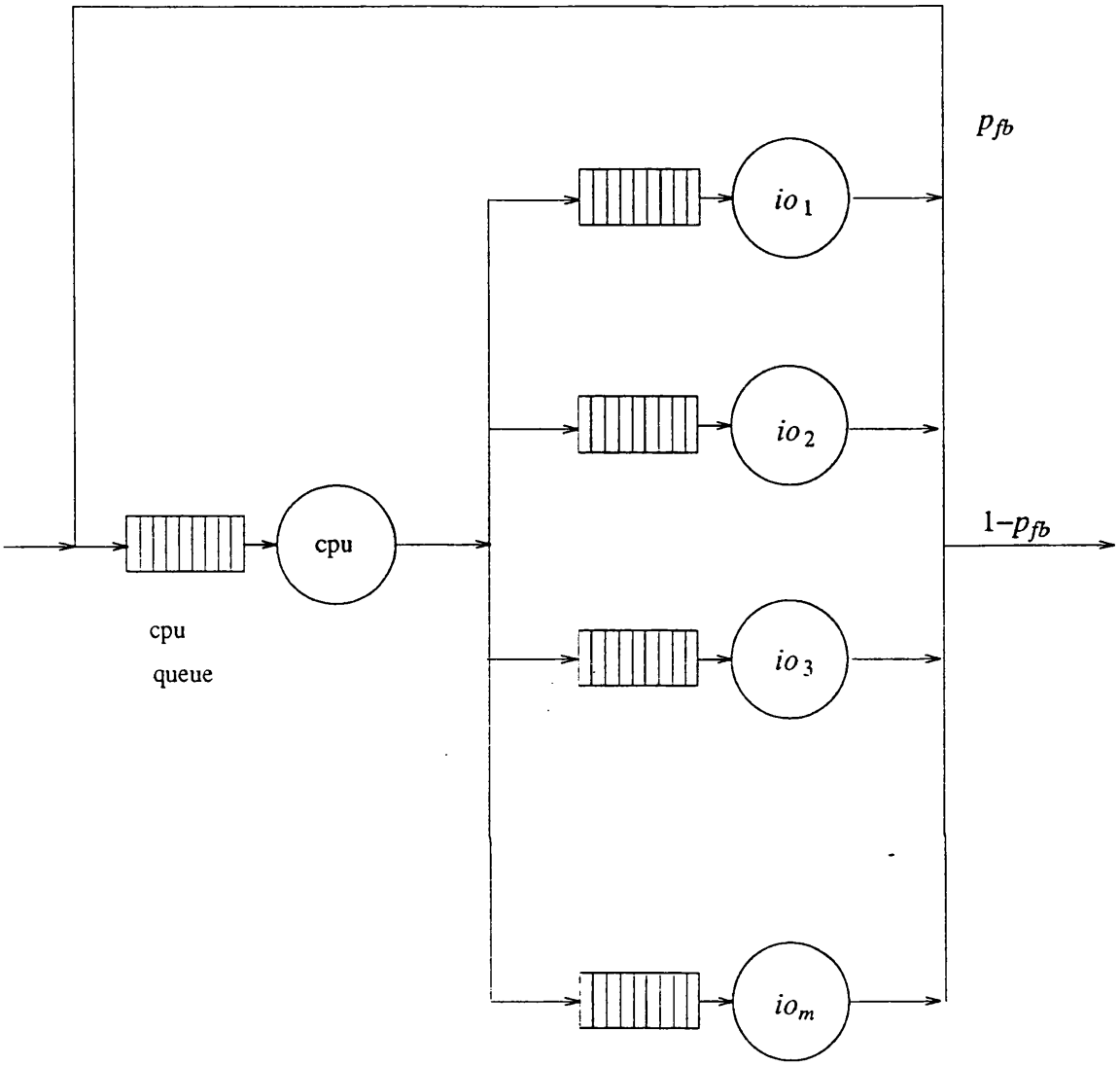


Figure 2.13. Inner Model of a Multiprogramming Computer System

groups. Thus the system is considered to be merely completely decomposable. The evaluation of such a system can then be tackled by

- modeling the interactions within groups as if interactions among groups do not exist.
- modeling the interactions among groups without referring to the interactions within groups.

This evaluation technique is justified by the fact that

- A local equilibrium is reached by the strong interactions within each group almost independent of the other groups.
- The weak interactions among groups make themselves felt and the whole system moves towards a global equilibrium maintaining approximately the relative equilibrium value attained by the state

variable of each group at the end of its running period.

We now study the inner and outer model of a multiprogramming computer system by using the decomposition method. Under the assumption of exponential service time distribution, the inner model can be easily modeled by the closed network of queues with population N equal to the multiprogramming level. As shown in figure 2.14, there are m number of service centres in the system. That is one cpu centre and $(m-1)$ i/o service centres. The well known Jackson's queueing network[33] model can be used here.

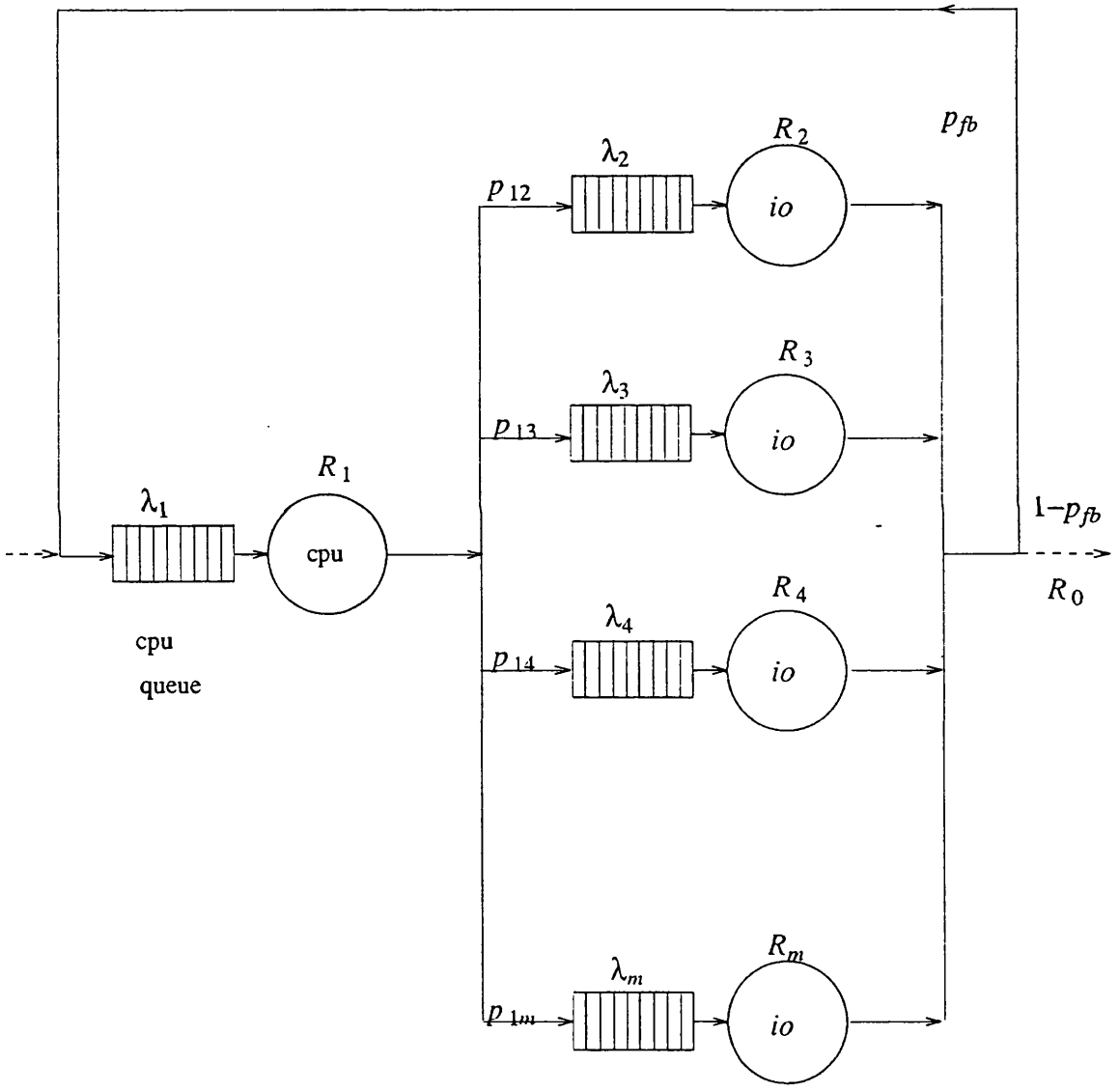


Figure 2.14. Queueing Network Model of a Decomposed Computer System

The m service centres of the inner model are denoted by $R_1, R_2, \cdots, R_m$. The interaction point from the inner model to the outer model is denoted by R_0. The service time of the ith service centre R_i is exponentially distributed with mean $1/\mu_i$. A job in the inner system has a probability p_{ij} of walking from

centre R_i to centre R_j.

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix} \tag{2.53}$$

is the Markov matrix which defines the probability of the random walk. Under the assumption of the inner system being decomposable, a closed system of queues, with negligible probability of leaving the inner system, can be formed.

The matrix $\mathbf{P}$ has the property of

$$\sum_{i=1}^{m} p_{ji} = 1\,, \quad j=1, \cdots, m \tag{2.54}$$

The Markov matrix defining the inner model is given by

$$\mathbf{P} = \begin{bmatrix} 0 & p_{12} & \cdots & p_{1m} \\ 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \tag{2.55}$$

where

$$\sum_{j=2}^{m} p_{1j} = 1$$

A useful parameter of the closed queueing network is the relative throughput $e_i, i=1, \cdots, m$, which can be derived from

$$e_i = \sum_{j=1}^{m} e_j p_{ji} \tag{2.56}$$

A closed queueing network with exponential service centres has a product form solution

$$P(n_1, n_2, \cdots, n_m) = \frac{p_1(n_1) p_2(n_2) \cdots p_m(n_m)}{G(N)} \tag{2.57}$$

where $P(n_1, n_2, \cdots, n_m)$ are the steady state probabilities of a network state with m service centres, $P_j(n_j)(j=1, \cdots, m)$ are a steady state probabilities of the state of service centre R_j with n_j jobs, $G(N)$ is a normalization factor of the product form solution

$$G(N) = \sum_{n_1 + \cdots + n_m = N} X_1(n_1) X_2(n_2) \cdots X_m(n_m) \tag{2.58}$$

and $X_j(n_j)$ are factors proportional to the steady state probabilities of n_j number of jobs in service centres R_j in isolation. Since the service centres of the inner model are single server service centres, $X_j(n)$ are given by

$$X_j(n) = \left[\frac{e_j}{\mu_j}\right]^n \quad n=0, \cdots, N \tag{2.59}$$

The equation for $G(N)$ can be simplified as

$$G_j(n) = G_{j-1}(n) + \frac{e_j}{\mu_j} G_j(n-1) \ , \quad j=2, \cdots, m: n=1, \cdots, N \tag{2.60}$$

The interarrival rate or throughput of centre R_i, denoted by Λ_i, is given by

$$\Lambda_j = \frac{e_j G(N-1)}{G(N)} \tag{2.61}$$

The utilization of the server at service centres R_j is given by

$$U_j = \frac{\Lambda_j}{\mu_j} \tag{2.62}$$

The mean queue length of the service centre R_j is

$$\bar{n}_j = \sum_{i=1}^{N} \frac{(e_j/\mu_j)^i G(N-i)}{G(N)} \tag{2.63}$$

From Little's formula the mean response time of service centre R_j is

$$\bar{R}_j = \frac{\bar{n}_j}{\Lambda_j} \tag{2.64}$$

An important system parameter is the marginal distribution of queue length, which can be derived by

$$P_j(n) = \\ = \frac{(e_j/\mu_j)^n [G(N-n) - (e_j/\mu_j) G(N-n-1)]}{G(N)} \tag{2.65}$$

with a special case of $n=N$. We have

$$P_j(N) = \left[e_j/\mu_j\right]^N \frac{[G(0) - (e_j/\mu_j) G(-1)]}{G(N)} \tag{2.66}$$

Since $G(0)=1$ and $G(i)=0$ for $i<0$,

$$P_j(N) = \frac{(e_j/\mu_j)^N}{G(N)} \tag{2.67}$$

Now we shall derive the state dependent aggregated service rate $\mu(n)$ of the inner model. After the local equilibrium is reached at the inner model, the probability of a job moving from the service centre R_j of the inner model to the point R_0 at the outer model is defined by p_{j0}, $j=2, \cdots, m$, where

$$p_{j0} \ll p_{j1}$$

and

$$p_{10}=0$$

The probability of feedback to cpu after paging is given by

$$p_{fb} = 1-\frac{1}{n_p} \tag{2.68}$$

Therefore the probability of a job leaving the inner model from service centre R_j $(j=2, \cdots, m)$ is given by

$$\begin{aligned} p_{j0} &= p_{1j}p_{fb} \\ &= (1-\frac{1}{n_p})p_{1j} \end{aligned} \tag{2.69}$$

We have

$$p_{fb} = \sum_{j=2}^{m} p_{j0} \tag{2.70}$$

At the multiprogramming level of n the aggregated service rate of the inner model is equal to

$$\mu(n) = \sum_{j=2}^{m} \mu_j[1-P_j(n)]p_{j0}\ , \quad n=1,...,N;\mu(0)=0 \tag{2.71}$$

The outer model can be easily evaluated as a single server service centre with state dependent service rate

$$\mu(n) = \begin{cases} \mu(n) & \text{if } n<N \\ \mu(N) & \text{if } n \geq N \end{cases} \tag{2.72}$$

2.5.3 Diffusion Approximation Approach with Non-exponential Service Time

The application of the conventional Jackson's queueing network solution is limited by the assumption of the exponential distribution of service time. In general, the assumption is not well justified, especially in the case of a database oriented computer system. In such a system transactions are i/o bound and the i/o

service time tends to be uniformly distributed rather than exponentially distributed. This leads us to employ the diffusion approximation approach to model the non-exponential servers in a computer system. We shall still use the decomposition method of the inner and outer model. But here we only concentrate on solving the inner model by using diffusion approximation method; while the technique for solving the outer model is the same.

The diffusion approximation method can obtain highly accurate results when the load of traffic in the queueing network is heavy[39], as for the database oriented computer system whose work load to the i/o service centre is considerably heavy. This method can provide the exact solution in the case of exponential service time distribution. The basic idea of the method is to approximate a discrete-state process (e.g. a random walk) by a diffusion process with continuous path. The discrete-state of interest in queueing network is the number of customers in each state. The probability of a discrete-state n is denoted by $P(n)$. It can be transformed to a continuous time-space, i.e., $P(n) \rightarrow f(x,t)$, where $f(x,t)$ denotes the probability of x customers at time t.

We first briefly present the diffusion approximation for a single queue, while the detailed derivations can be found in Lavenberg[42]. Let $A(t)$ and $D(t)$ be the cumulative number of arrivals and departures up to time t respectively. Then the number of customers in the service centre at time t is given by

$$Q(t) = A(t) - D(t) \tag{2.73}$$

The change in queue length between time t and $t+\Delta t$ is

$$Q(t+\Delta t) - Q(t) = [A(t+\Delta t) - A(t)] - [D(t+\Delta t) - D(t)] \tag{2.74}$$

or

$$\Delta Q(t) = \Delta A(t) - \Delta D(t) \tag{2.75}$$

Let the interarrival time and service time be both independent and identically distributed (i.i.d) with (mean, variance) given by (μ_a, σ_a^2) and (μ_s, σ_s^2) respectively. Then according to the central limit theorem, we can show that if Δt is sufficiently large, many events will take place between time t and $t+\Delta t$. If $Q(t)$ does not become zero in this interval, then $\Delta Q(t)$ should be approximately normally distributed with mean

$$E[\Delta Q(t)] = (1/\mu_a - 1/\mu_s)\Delta t \tag{2.76}$$

and variance

$$Var[\Delta Q(t)] = (c_a/\mu_a + c_s/\mu_s)\Delta t \tag{2.77}$$

where $c_a = \sigma_a^2/\mu_a^2$ and $c_s = \sigma_s^2/\mu_s^2$ are the coefficients of interarrival time and service time respectively.

The above results lead to the following continuous process $X(t)$ as an approximation of a discrete-state process $Q(t)$: it is defined as such that its incremental change $\Delta X(t) = X(t+\Delta t) - X(t)$ is

normally distributed with mean $\beta \cdot \Delta t$ and variance $\alpha \cdot \Delta t$, where

$$\alpha = \frac{c_a}{\mu_a} + \frac{c_s}{\mu_s} \tag{2.78}$$

and

$$\beta = \frac{1}{\mu_a} - \frac{1}{\mu_s} \tag{2.79}$$

The density function of $X(t)$

$$f(x,t)dx = Pr\{x \le X(t) < x+dx\} \tag{2.80}$$

satisfies the Kolmogorov forward diffusion equation. The solution of the diffusion equation is then given by

$$f(x) = \begin{cases} -\gamma e^{\gamma x} & \text{if } x \ge 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.81}$$

where

$$\begin{aligned} \gamma &= \frac{2\beta}{\alpha} \\ &= \frac{2(1/\mu_a - 1/\mu_s)}{c_a/\mu_a + c_s/\mu_s} \\ &= \frac{-2(1-\rho)}{c_a\rho + c_s} \end{aligned} \tag{2.82}$$

and $\rho = \mu_s/\mu_a$ is the utility of the service centre.

Let $\hat{p}(n)$ denote the diffusion approximation to the probability that the queue length is n in steady-state, then

$$\begin{aligned} \hat{p}(0) &= 1-\rho \\ \hat{p}(n) &= \rho(1-\hat{\rho})\hat{\rho}^n \quad n \ge 1 \end{aligned} \tag{2.83}$$

where $\hat{\rho} = e^{\gamma}$.

Now we shall derive the diffusion approximation equations for the closed queueing network of a computer system. Applying the single queue solution to the closed queueing network, we can obtain some interesting results. We shall consider a closed network of queues consisting m single server service centres as shown in figure 2.15. The service times at centre i are i.i.d with mean μ_i, variance σ_i^2 and coefficient $c_i + \sigma_i^2/\mu_i^2$, $i = 1, \cdots, m$. The routing path of a customer within the closed system is specified by the Markov chain matrix probability p_{ij} satisfying

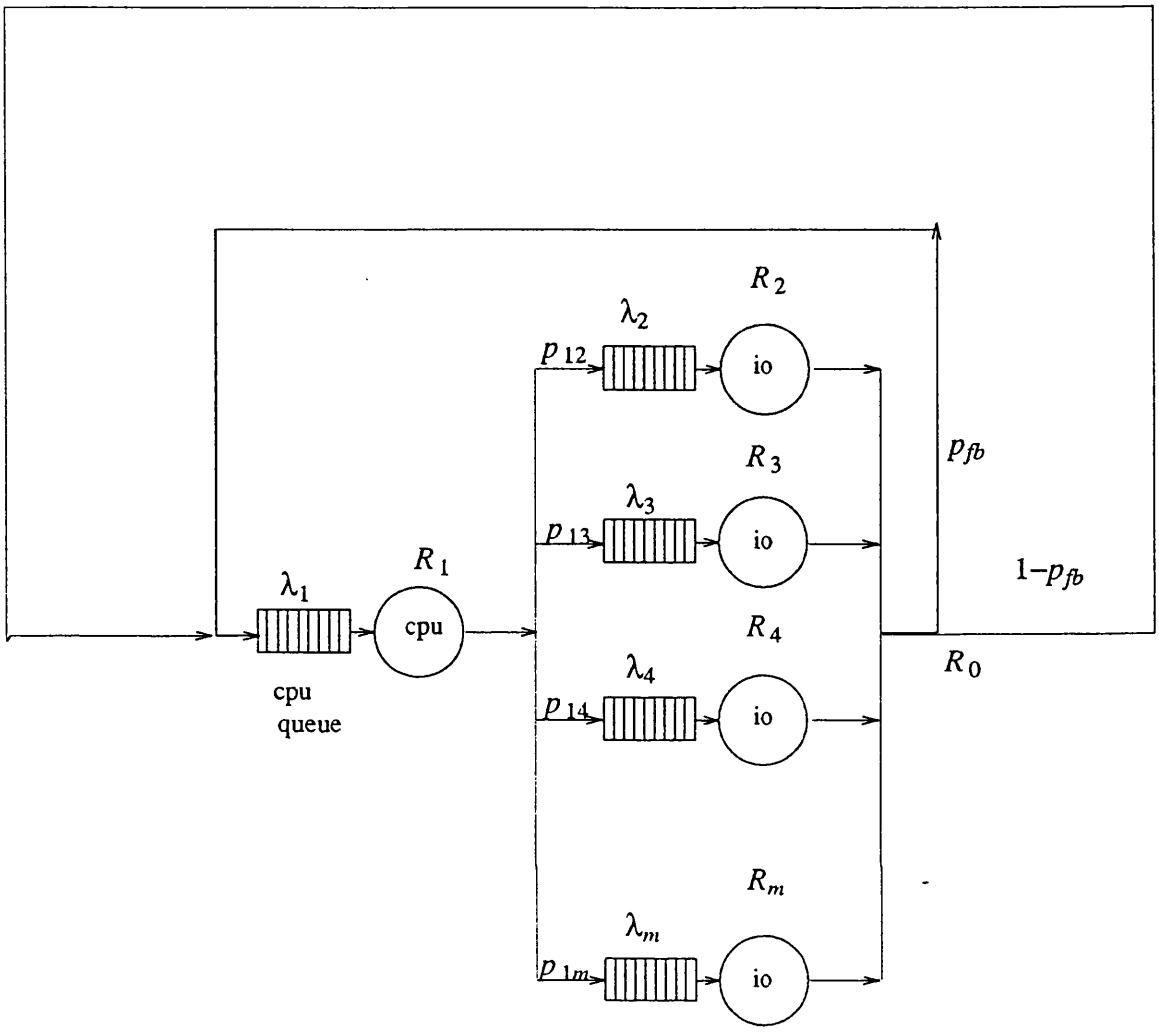


Figure 2.15. Closed Queueing Network

$$\sum_{j=1}^{m} p_{ij} = 1 \tag{2.84}$$

The relative throughput of service centre i is given by

$$e_i = \sum_{j+1}^{m} e_j p_{ji} \tag{2.85}$$

Similar to the single queue solution, the parameters α, β and γ are defined as matrixes $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ respectively. If $p_{ii}=0$, $\boldsymbol{\alpha}$ is a $m \times m$ matrix given by

$$\boldsymbol{\alpha} = \sum_{k=0}^{m} \frac{c_k}{\mu_k} \mathbf{V}_k \cdot \mathbf{V}_k' = \mathbf{W} \tag{2.86}$$

where $\mathbf{V}_k$ is an m - dimensional column vector whose kth element is unity and ith element $(i \neq k)$ is $-p_{ki}$.

That is

$$\mathbf{V}_k' = [-p_{k1}, -p_{k2}, \cdots, 1, \cdots, -p_{km}] \tag{2.87}$$

Matrix $\mathbf{W}$ is a non negative definite matrix of $m \times m$ whose coefficients are

$$w_{ij} = \begin{cases} \sum_{k=0}^{m} p_{ki}(1-p_{ki})/\mu_k & i=j \\ -\sum_{k=0}^{m} p_{ki}p_{kj}/\mu_k & i \neq j \end{cases} \tag{2.88}$$

β is an m-dimensional column vector in which the ith element is given by

$$\beta_i = \sum_{k=0}^{m} p_{ki}/\mu_k - 1/\mu_i \quad i=1, \cdots, m \tag{2.89}$$

γ is an m-dimensional vector given by

$$\gamma = 2\alpha^{-1}\beta \tag{2.90}$$

Here we shall introduce a novel method to solve the closed queueing network of diffusion approximation. The closed queueing network has an approximate steady state discrete probability $\hat{P}(n_1, n_2, \cdots, n_m)$ given by

$$\hat{P}(n_1, n_2, \cdots, n_m) = \frac{P_1(n_1)P_2(n_2)\cdots P_m(n_m)}{G(N)} \tag{2.91}$$

where $P_j(n_j)$ is a factor corresponding to the steady-state probability of the state of the service centre i in isolation and $G(N)$ is a normalizing constant given by

$$G(N) = \sum_{n_1+\cdots+n_m=N} X_1(n_1)X_2(n_2)\cdots X_m(n_m) \tag{2.92}$$

and N is the number of customers in the closed queueing network, and

$$X_j(n_j) = (e^{\gamma_j})^{n_j} = e^{n_j\gamma_j} \tag{2.93}$$

Noticing the clear resemblance of the above form to the product form of Jackson's closed queueing network, we can similarly obtain the mean measurements and marginal steady-state probability $\hat{P}_j(n)$ of queueing length at service centre j.

Invoking the convolution algorithm we have

$$G_j(n) = G_{j-1}(n) + e^{\gamma_j}G_j(n-1) \quad j=2, \cdots, m;\ n=1, \cdots N \tag{2.94}$$

The steady-state probability is given by

$$\hat{P}_j(n) = \frac{e^{n\gamma_j}[G(N-n)-e^{\gamma_j}G(N-n-1)]}{G(N)} \tag{2.95}$$

and

$$\hat{P}_j(N) = \frac{e^{N\gamma_j}}{G(N)} \tag{2.96}$$

The mean queueing length is given by

$$\bar{n}_j = \sum_{n=1}^{N} \frac{e^{n\gamma_j}G(N-n)}{G(N)} \tag{2.97}$$

The throughput is given by

$$\Lambda_j = \frac{e_j G(N-1)}{G(N)} \tag{2.98}$$

The mean response time is given by

$$\bar{R}_j = \frac{\bar{N}_j}{\Lambda_j} \tag{2.99}$$

Referring to the inner and outer model of multiprogramming computer system introduced in sections 2.5.1 and 2.5.2, we can easily derive the aggregated service rate of the inner model with the multiprogramming level n. That is

$$\mu(n) = \sum_{j=2}^{m} \mu_j[1-\hat{P}_j(n)]p_{j0} \quad n=1,...,N; \quad \mu(0)=0 \tag{2.100}$$

where p_{jo} is the routine probability of a job leaving the inner model from service centre j.

$$p_{j0} = (1-\frac{1}{n_p})p_{1j} \tag{2.101}$$

The outer model of the multiprogramming computer system can be evaluated with the given $\mu(n)$ by the decomposition method introduced in section 2.5.2. The above results will be used in the following chapters when evaluating computer systems.

2.6 Summary

In this chapter, the methods to evaluate the major components of distributed databases, i.e. cpu, i/o and computer network, have been presented. Decomposition and diffusion approximation methods have been used to model database-bound multiprogramming computer systems.

Chapter 3. Concurrency Control Model of Centralized DB

3.1 Introduction

The problems associated with the evaluation of centralized databases have been extensively studied in recent years. In particular, the locking problem of centralized databases has received much attention and substantial research efforts have been made to evaluate the performance of locking mechanisms. However the following problems still have not been solved. The most difficult one associated with locking is to evaluate the waiting time upon locking conflicts. This is mainly because of the stochastic nature of the waiting time, which depends on the behavior of the departure process of transaction execution. Another problem associated with locking is the queueing discipline, which often means that the lock manager can apply preemptive priority scheduling over the transaction executions. The difficulty of evaluating locking performance of a multiple class transaction system is also widely recognized; this is due to the fact that the overall system performance depends on not only different service demands but also different lock conflict rates and waiting times of multiclass transactions. In this chapter a unified analytic model is introduced to solve the above problems. The model can apply to both open and closed database systems with either single class or multiple class transactions[49].

Section 2 represents the specification of the concurrency control and locking algorithm. Section 3 introduces the analytic model to evaluate the waiting time of the blocked transactions and priority queueing in an open system. Section 4 applies the waiting model to a closed system. Section 5 introduces the method to evaluate multiclass transactions. Section 6 validates the analytic results with the simulation results obtained by Ries and Stonebraker[68].

3.2 Model Specification

3.2.1 Basic Two Phase Locking

Bernstein and Goodman[8] have given a very thorough survey of the concurrency control algorithms in database systems. It is therefore appropriate to provide only a brief discussion here, while the detail can be found in their paper.

Two phase locking is based on the concept of detecting and preventing conflicts between concurrent operations. The unit of data is protected by locks. A read transaction has to own a shared lock before actually performing the operation. The write transaction has to own an exclusive lock before writing data to the database. A shared lock can be owned by many read transactions; while an exclusive lock can only belong to one write transaction.

A transaction is said to be in a lock conflict state if one of the following situations happen: for read-write synchronization two locks conflict if one is a shared lock and the other is an exclusive lock, and both locks are on the same data item; for write-write synchronization two locks conflict if both locks are on the same item and are exclusive locks.

The correct execution of locking is governed by a concurrency control algorithm. With two phase locking, a transaction requests locks in a growing phase and releases locks in a shrinking phase. Upon lock conflict, a collision resolution algorithm needs to be used to deal with the scheduling of the blocked transactions. If the required locks are denied in the growing phase, the transaction will be scheduled for retry according to the collision resolution algorithm used. When locks are released at the shrinking phase, the blocked transactions waiting for the locks are dealt with according to the collision resolution algorithm.

3.2.2 Collision Resolution Scheduling Algorithm

A number of collision resolution scheduling algorithms have been introduced[28,83]. They can be mainly classified into two categories: the fixed waiting algorithm and the scheduled waiting algorithm. The fixed waiting collision resolution algorithm is to restart a blocked transaction after a fixed period regardless of the availability of the locks, while the scheduled waiting algorithm is to free blocked transactions as soon as the required locks become available. As stated in chapter 2, most researches have been carried out in evaluating the fixed waiting collision resolution algorithm of the centralized system. The scheduled waiting algorithm is not well studied. It is the purpose of this chapter to address this problem.

The scheduled waiting algorithm is as follows. When a transaction encounters conflict locks at the growing phase, it enters a blocked transaction queue and all the conflict locks are recorded. When locks are released at the shrinking phase, the blocked transactions and their conflict locks are checked against the released locks. The blocked transactions will be freed if all the required locks become available. The freed transactions then compete for their locks at their next growing phases. The more detailed description of the algorithm can be found in Ries and Stonebraker's paper[68], in which they used a simulation model to evaluate the performance of a centralized database.

The implementation of scheduled waiting algorithm involves setting up a hash table, in which each item is consist of a lock and a list of the transactions waiting for the lock. Setting up locks involves checking the hash table for the corresponding lock items; if all the locks are available, the transaction can grant the locks by locking them in one atomic operation. Once a transaction is completed the database management system releases the locks in the hash table. The blocked transactions associated with the released locks are therefore freed[28].

3.3 The Scheduled Waiting Model in an Open System

A database system can usually be accessed by a number of users simultaneously via local or remote terminals. We can consider users' requests as a steady stream of jobs arriving to the database. Each of the

jobs requires certain amount of database service. Such a system can be naturally considered as an open database system in which transactions arrive at the source of the system and depart at the sink as shown in figure 3.1. As introduced in chapter 2, a database system is composed of cpu and i/o devices. They can be modeled by a queuing centre using decomposition method, introduced in section 2.5. A transaction requiring locking and execution operations will be processed by the computer system.

In an open database system, a transaction first requests locks at c_l. If all locks are granted, the transaction is executed at c_e, and releases the locks and leaves the system; otherwise the transaction is said to be blocked and put into the blocked queue at b. The blocked transaction in the blocked queue will be freed when the required locks are released. It then re-enters c_l to request locks again. The lock request and transaction execution share the same computer resources. The locking operations have priority over the execution operations, which means that the lock request operations are processed by a computer with higher priority over the transaction execution operations.

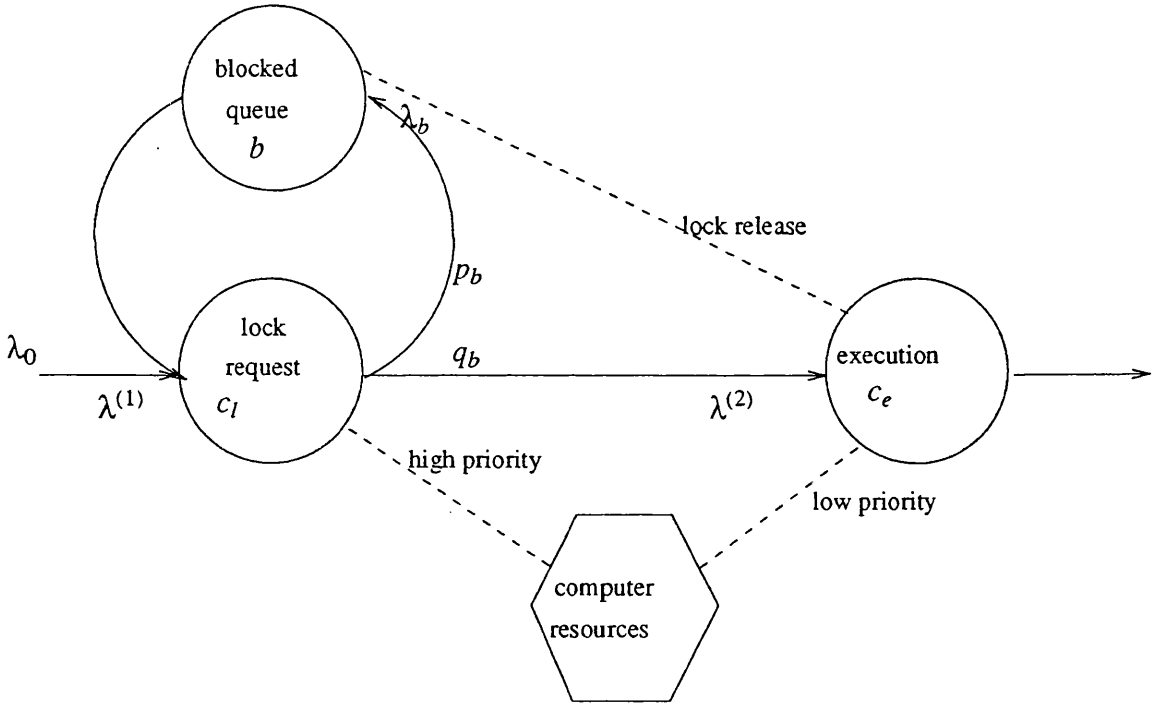


Figure 3.1. Locking Model of an Open Database System

We shall now derive the throughput of the locking model shown in figure 3.1, where the throughput at c_l, c_e and b are denoted by $\lambda^{(1)}$, $\lambda^{(2)}$ and λ_b respectively. The interarrival rates at the source is denoted by λ_0. Let us define p_b as the probability that a transaction can not grant all the required locks, therefore must enter the blocked queue at b; and $q_b=1-p_b$ is the probability that a transaction can grant all the requred locks. It can be seen that under steady state

$$\lambda^{(1)} = \lambda_0 + \lambda_b \tag{3.1a}$$

$$\lambda^{(2)} = \lambda_0 \tag{3.1b}$$

$$\lambda_b = p_b \lambda^{(1)} \tag{3.1c}$$

$\lambda^{(1)}$ and λ_b can be rewritten in terms of λ_0 as

$$\lambda^{(1)} = \frac{\lambda_0}{1-p_b} \tag{3.1d}$$

and

$$\lambda_b = \frac{p_b \lambda_0}{1-p_b} \tag{3.1e}$$

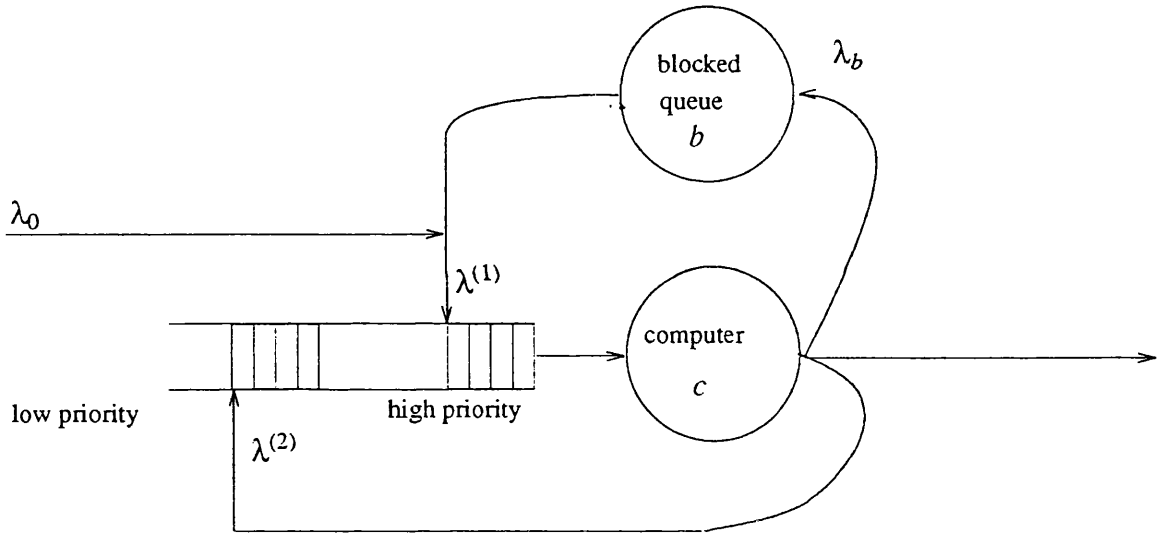


Figure 3.2. Queueing Network Model of an Open System

Since lock request and transaction execcution share the same computer resources as shown in figure 3.1, we can reconstruct the locking model into a queueing network model with priority queueing discipline as shown in figure 3.2. The computer resource is represented as a single server service centre. Lock request is served with first class priority; while transaction execution is served with a second class priority. The arrival rate of the first and second class transactions are given by $\lambda^{(1)}$ and $\lambda^{(2)}$ respectively. Since the waiting in the blocked queue does not require computer resources, it can be modeled as an infinite-server service centre. That is each blocked transaction is served immediately for a period determined by the availability of locks.

As discussed before, the problem associated with evaluating locking performance is very complicated because of its stochastic nature. The waiting time analysis can not be isolated from the rest of the system.

It depends on the system throughput, the number of locks already held, and the number of locks in the waiting centre. The stochastic nature of the blocking process is strongly related with the departure process of the transaction execution, i.e. the second class priority customer in fugure 3.2. The prime goal here is to obtain the characteristics of the departure process of the second class customer.

To enable meaningful analysis, we assume that the interarrival process is Poisson. The service time is independent and identically distributed. It is appropriate to treat the computer service centre c as a preemptive resume priority for $M/G/1$ with two priority levels. Strictly speaking, the computer service centre c is not an independent $M/G/1$ queue; the whole model should be treated as a queueing network because of the effect of feedback. However under the following assumptions the computer service centre c may reasonably be approximated by an independent $M/G/1$ queue with two priority levels. Firstly we assume the lock overhead is small. In a real system the lock table is usually maintained in a hash table either in main memory or on disk. The time spent to fetch certain items from the hash table is very small compared to the time spent in the transaction execution operations[83]. Secondly we assume that the probability of lock conflict is small. In a real database system with a scheduled waiting collision resolution algorithm, only the transactions in the execution state hold locks. The transactions in the blocked queue do not hold any locks. Furthermore only those blocked transactions which can grant all the required locks are allowed to re-enter the system; others must wait until such moment comes. Therefore the probability of lock conflict is small in most cases.

In figure 3.2, transactions first arrive with a Poisson distribution and enter the computer queue as a high priority customer. Since we assume that the lock overhead is small, the interarrival times of transactins are much greater than the service times for lock request, which means that there is almost no queueing of first priority customers. Suppose τ_n denotes the epoch of the nth arriving high priority customer and S_n denotes the service time of the nth customer of first priority. Since there is almost no queuing for these customers, the departure epoch of the nth high priority customer is approximately equal to $\{\tau_n+S_n\}$. According to Doob[21], if τ_n is Poisson and S_n is independent and identically distributed, the instants $\{\tau_n+S_n\}$ forms a Poisson process. We can therefore say that the output process of the high priority customer is approximately Poisson.

Since we assume that the probability of lock conflict is very small, the effect of feedback should be very weak. Futhermore since the output process of the high priority customer is approximately Poisson and the blocked queue is an infinite-server service centre, the output process of the blocked queue is also Poisson. Therefore it also forms a Poisson input to the computer service centre.

The input process of the low priority customer, i.e. the transaction execution customer, is formed by the ouput process of the high priority customer, i.e. lock request customer. It is therefore approximately Poisson. The low priority customer, i.e. transaction execution, is only served once and will not cause any feedbacks.

As introduced in section 2.5 a computer system with a compound cpu and i/o can be modeled by a single server queue. Therefore an open database system can be represented by a the $M/G/1$ preemptive resume priority queue with two priority levels. The characteristics of particular interest are the departure process of the second priority customer; since this process determines the entire waiting process at the waiting centre.

The following equations are derived to obtain the Laplace-Stieltjes transform of the distribution function of departure process of the second priority customers.

We shall describe the ith priority levels as (i). If $i<j$, the customer of ith priority level will have priority service over those of jth level. The distribution of the interarrival time for the ith priority customer is supposed to be exponentially distributed with mean and variance $\lambda^{(i)}$; the distribution of the service times of the ith priority customer is indicated by $H^{(i)}(t)$ with mean

$$\mu^{(i)} = \int_0^\infty tH^{(i)}(t)dt \tag{3.2}$$

and second moment

$$\sigma^{(i)} = \int_0^\infty t^2 H^{(i)}(t)dt. \tag{3.3}$$

Let us define $\chi_{l_n}^{(i)}$ as the stochastic process of the number of customers of the ith priority level in the system at the time t when the nth customer arrives. We shall first prove in the following theorem that the departure process of the second priority customer is Markovian, which means that the time interval of two successive departures of the second priority customers, i.e. transaction execution customers, is independent and identically distributed. This property is very important to a stochastic variable as it is a necessary condition of a proper distribution function used in queueing networks. The proof can be found in appendix B.

Theorem 3.1: The process $\{\chi_t^{(2)}, t\in[0,\infty)\}$ which is regenerative with respects to the renewal precess of the interdeparture time of the second priority customer is Markovian.

In order to obtain the distribution function of the departure process of the second priority customer we need to employ the concept of completion time of a customer, which was introduced by Cohen[15]. This is the time between the epoch when the service of a customers begins and the first epoch thereafter at which the server becomes available to another customer. The completion time of a second priority customer is denoted by c_i.

The Laplace-Stieltjes transform of the completion time c_i of a second priority customer with preemptive resume priority discipline is given by[15]

$$E\{e^{-sc_i}\} = \sum_{k=0}^{\infty} \int_0^{\infty} e^{-st} \frac{(\lambda^{(1)}t)^k}{k!} e^{-\lambda^{(1)}t} dH^{(2)}(t) \{\mu^{(1)}(s, 1)\}^k$$

$$= \beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(1)}(s, 1))\} \quad , \qquad \text{Re}s \geq 0 \ , \tag{3.4}$$

with

$$\beta^{(2)}(s) = \int_0^{\infty} e^{-st} dH^{(2)}(t) \tag{3.5}$$

and $\mu^{(1)}(s, 1)$ defined as the Laplace-Stieltjes transform of the busy period of the first priority level. $\mu^{(1)}(s, 1)$ is the zero with the smallest absolute value of

$$z = \beta^{(1)}\{s+(1-z)\lambda^{(1)}\} \ , \quad \text{Re } s \geq 0 \tag{3.6}$$

with

$$\beta^{(1)}(s) = \int_0^{\infty} e^{-st} dH^{(1)}(t) \tag{3.7}$$

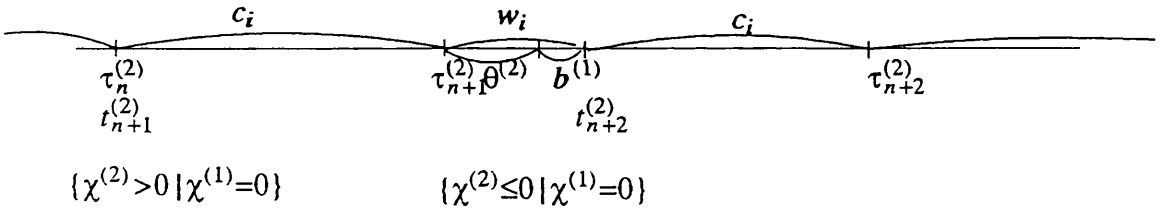


Figure 3.3. Epochs of the Second Priority Customers

We further introduce the concept of pre-waiting time of the second priority customer, denoted by w_i. This is the time between the epoch at which the server becomes idle and the first epoch thereafter at which the service begins to serve a second priority customer. This equals to the idle period of the system plus the residual busy period of the first priority customer. In order to have a clearer picture of the $M/G/1$ queue with two priority levels, figure 3.3 illustrates the epochs of the second priority customers. $\tau_n^{(2)}$ denotes the departure epoch of the nth second priority customer and $t_n^{(2)}$ denotes the epoch at which the nth second priority customer starts to be served. If before epoch $\tau_n^{(2)}$ the $(n+1)$th second priority customer has already arrived, i.e. $\{\chi^{(2)}>0 \mid \chi^{(1)}=0\}$, the server starts to serve the $(n+1)$th customer immediately. This period, i.e. from $t_{n+1}^{(2)}$ to $\tau_{n+1}^{(2)}$, is exactly the completion time of the second class customer[15]. If however at the epoch $\tau_{n+1}^{(2)}$ the $(n+2)$th second priority customer has not yet arrived, i.e. $\{\chi^{(2)}\leq 0 \mid \chi^{(1)}=0\}$, the server will spend a pre-waiting time w_i to wait for the arrival of the $(n+2)$th second priority customer and the completion time of the first priority customers before the service to the $(n+2)$th second priority customer begins. Since from the epoch $\tau_{n+1}^{(2)}$ to the first epoch the $(n+2)$th

second priority customer arrives is the residual interarrival time of the second priority customer, denoted by $\theta^{(2)}$. The period from the epoch of the arrival of the $(n+2)$th customer, i.e. $\tau_{n+1}^{(2)}+\theta^{(2)}$ to epoch of the completion of service of the first priority customer is the the residual busy period of the first priority customer, denoted by $b^{(1)}$. Thus the pre-waiting time w_i is composed of $\theta^{(2)}$ and $b^{(1)}$.

Since the interarrival process of the second priority customer is Poisson, the residual interarrival time is equal to the Poisson interarrival time with mean $1/\lambda^{(2)}$. The Laplace-Stieltjes transform of $\theta^{(2)}$ is

$$E\{e^{-s\theta^{(2)}}\} = \frac{\lambda^{(2)}}{s+\lambda^{(2)}} \tag{3.8}$$

The residual busy period $b^{(1)}$ starts with a initial waiting time $v_\tau^{(1)}$ which is equal to the virtual waiting time of the $\chi_t^{(1)}$ - system at the epoch τ at which the second priority customer arrives. It is due to Cohen that[15]

$$\begin{aligned} E\{e^{-sb^{(1)}}\} &= \int_{t=0}^{\infty}\int_{\sigma=0}^{\infty} \exp[-\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\}\sigma]\, d_\sigma Pr\{v_t^{(1)}<\sigma \mid v_0^{(1)}=0\} e^{-\lambda^{(2)}t} d\lambda^{(2)}t \\ &= \frac{\lambda^{(2)}-s+\lambda^{(1)}[\mu^{(1)}(s,1)-\mu^{(1)}(\lambda^{(2)},1)]}{[\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)](1-s/\lambda^{(2)})} \end{aligned} \tag{3.9}$$

Now we can form the pre-waiting time of the customer of second priority level. That is

$$w_i = \theta^{(2)}+b^{(1)} \tag{3.10}$$

From (3.8), (3.9) and (3.10), we have

$$\begin{aligned} E\{e^{-sw_i}\} &= E\{e^{-s\theta^{(2)}}\}\cdot E\{e^{-sb^{(1)}}\} \\ &= \frac{\lambda^{(2)}}{s+\lambda^{(2)}}\cdot\frac{\lambda^{(2)}-s+\lambda^{(1)}[\mu^{(1)}(s,1)-\mu^{(1)}(\lambda^{(2)},1)]}{[\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)](1-s/\lambda^{(2)})} \end{aligned}$$

if Re $s\geq 0$.

The following derivations aim to obtain the distribution function of the departure time $\tau_{n+1}^{(2)}-\tau_n^{(2)}$ of the second priority customer which can be presented by c_i and w_i. As we can see from figure 3.3, if a second priority customer leaves the system with at least one second priority customer in the queue, i.e. $\{\chi^{(2)}>0 \mid \chi^{(1)}=0\}$, the interdeparture time $\tau_{n+1}^{(2)}-\tau_n^{(2)}$ is equal to the completion time of the next second priority customer. If however the nth second priority customer leaves system empty, i.e. $\{\chi^{(2)}\leq 0 \mid \chi^{(1)}=0\}$ the interdeparture time between the nth and $(n+1)$th departures is the sum of the completion and pre-waiting time of the next arriving second priority customer. Thus the inter-departure time can be given by

$$\tau_{n+1}-\tau_n = c_i + \begin{cases} w_i & \text{if } \{\chi^{(2)}=0 \mid \chi^{(1)}=0\} \\ 0 & \text{if } \{\chi^{(2)}>0 \mid \chi^{(1)}=0\} \end{cases} \tag{3.11}$$

This can be rewritten as

$$\begin{aligned} Pr\{\tau_{n+1}-\tau_n<t\} &= Pr\{\chi^{(2)}=0 \mid \chi^{(1)}=0\}Pr\{c_i<t\}*Pr\{w_i<t\} \\ &\quad + (1-Pr\{\chi^{(2)}=0 \mid \chi^{(1)}=0\})Pr\{c_i<t\} \end{aligned} \tag{3.12}$$

For the stationary process it follows that

$$\lim_{t\to\infty} Pr\{\chi^{(2)}=0 \mid \chi^{(1)}=0\} = \frac{1-a}{1-a^{(1)}} \quad \text{if } a<1 \tag{3.13}$$

and

$$\lim_{t\to\infty} Pr\{\chi^{(2)}>0 \mid \chi^{(1)}=0\} = \frac{a^{(2)}}{1-a^{(1)}} \quad \text{if } a<1 \tag{3.14}$$

where

$$a^{(1)} = \lambda^{(1)}\mu^{(1)} \tag{3.15a}$$

$$a^{(2)} = \lambda^{(2)}\mu^{(2)} \tag{3.15b}$$

and

$$a = a^{(1)}+a^{(2)}. \tag{3.15c}$$

Let us denote $g^*(s)$ as the Laplace-Stieltjes transform of the density function of the departure process of the second priority customer. We have

$$\begin{aligned} g^*(s) &= E\{e^{-s\{\tau_{n+1}^{(2)}-\tau_n^{(2)}\}}\} \\ &= \frac{1-a}{1-a^{(1)}}E\{e^{-sc_i}\}E\{e^{-sw_i}\}+\frac{a^{(2)}}{1-a^{(1)}}E\{e^{-sc_i}\} \\ &= \frac{1-a}{1-a^{(1)}}\cdot\beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\}\cdot\frac{\lambda^{(2)}}{s+\lambda^{(2)}}\cdot\frac{\lambda^{(2)}-s+\lambda^{(1)}[\mu^{(1)}(s,1)-\mu^{(1)}(\lambda^{(2)},1)]}{[\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)](1-s/\lambda^{(2)})} \\ &\quad + \frac{a^{(2)}}{1-a^{(1)}}\beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\} \\ &= \frac{\beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\}}{1-a^{(1)}}\cdot \\ &\quad \left[(1-a)\left[\frac{\lambda^{(2)}}{\lambda^{(2)}+s}\right]\left[\frac{\lambda^{(2)}}{\lambda^{(2)}-s}\right]\left[\frac{\lambda^{(2)}-s+\lambda^{(1)}[\mu^{(2)}(s,1)-\mu^{(1)}(\lambda^{(2)},1)]}{\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)}\right]+a^{(2)}\right] \end{aligned} \tag{3.16a}$$

Let us denote μ_σ as the mean interdeparture time of the second priority customers. We have

$$\mu_\sigma = -\frac{d}{ds}g^*(0)\Big]_{s=0} \tag{3.16b}$$

For exponential service time, the mean service time of the inter-departures process of the second priority customer can be further derived as follows. Since

$$\beta^{(2)}(s) = \frac{1}{\mu^{(2)}s+1}$$

$$\frac{d}{ds}\mu^{(1)}(s,1)\Big]_{s=0} = \frac{d}{ds}\left[\frac{1}{2a^{(1)}}(\mu^{(1)}s+a^{(1)}+1-\sqrt{(\mu^{(1)}s+a^{(1)}+1)^2-4a^{(1)}})\right]_{s=0}$$

$$= -\frac{\mu^{(1)}}{1-a^{(1)}} \tag{3.17}$$

and

$$\mu^{(1)}(s,1)\Big]_{s=0} = \frac{1}{2a^{(1)}}(\mu^{(1)}s+a^{(1)}+1-\sqrt{(\mu^{(1)}s+a^{(1)}+1)^2-4a^{(1)}})\Big]_{s=0}$$

$$= 1 \tag{3.18}$$

we can obtain

$$\mu_\sigma = \frac{\mu^{(2)}}{1-a^{(1)}} + \frac{1-a}{(1-a^{(1)})^2}\cdot\frac{1}{\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)} \tag{3.19}$$

where

$$\mu^{(1)}(\lambda^{(2)},1) = \frac{1}{2a^{(1)}}\left[\lambda^{(2)}\mu^{(1)}+a^{(1)}+1-\sqrt{(\lambda^{(2)}\mu^{(1)}+a^{(1)}+1)^2-4a^{(1)}}\right] \tag{3.20}$$

In order to build the database waiting model we also need to obtain the residual inter-departure time of the second priority customer. It is due to Cohen that[15] for the stationary process, the Laplace-Stieltjes transform of the residual lifetime density function for the inter-departure process of the second priority customer is given by

$$g^*_\zeta(s) = \frac{1}{\mu_\sigma}\cdot\frac{1-g^*(s)}{s} \tag{3.21}$$

From the Laplace-Stieltjes transform of the density function of the inter-departure process of the second priority customer $g^*(s)$ we can easily obtain its mean. That is

$$\mu_{\sigma_\xi} = -\frac{1}{\mu_\sigma}\lim_{s\to 0}\frac{d}{ds}\left[\frac{1-g^*(s)}{s}\right]$$
$$= \frac{1}{2\mu_\sigma}\frac{d^2}{ds^2}g^*(0) \qquad (3.22)$$

The above results are very important and crucial to the evalution of scheduled waiting in a blocked queue. It is the most important step in the derivation of the blocking model. It reveals the mean and distribution of the departure process of the transaction execution job. Since the release of locks from the completed transactions triggers the release of the blocked transactions. The above results, therefore, define not only the mean waiting time but also the stochastic nature of the blocked transactions, which has not been solved before. Previous researches can only evaluate fixed waiting time from approximate estimation, as in Tay's paper, or even from empirical measurement, as given by Irani. Our result, on the other hand, provides exact evaluation of blocking with scheduled waiting collision resolution algorithm.

Blocking Model

We shall now derive the formulae for the waiting model of open database systems based on the inter-departure process of the second priority customer. The waiting time of the blocked transaction is closely related to the inter-departure time of the leaving transactions. We shall first derive the waiting time in the blocked queue and then the probability of locking conflict.

The performance model of a database system, as discussed in section 3.2 is constructed by an open network of queues with one computer service centre and one blocked transaction waiting centre. Each transaction starts with going through the computer server with a preemptive resume priority to request, set or release locks. Once a transaction gets all the required locks, it goes straight back to the computer server to be executed with a second class priority. After the execution the transaction finally departs from the computer server and releases all locks. From the point of view of the computer server each departure of its second priority customer will release r locks, where r is the average number of locks held by one transaction upon execution. Therefore the waiting time for each release of r locks in the blocked queue is equal to the inter-departure time of the second priority customer in the computer service centre.

From theorem 3.1 it follows that the departure epochs of the second priority customer are the regenerative points of the $\chi_i^{(2)}$- process. Consequently, the inter-departure times of the second priority customer are independent and identically distributed variables with the Laplace-Stieltjes transform of the density function given by $g^*(s)$ in equation (3.16a).

As shown in figure 3.4, the waiting time for each lock release can be naturally modeled by a waiting phase. A blocked transaction goes through one or more waiting phases before finally obtaining all the required locks. It should be noted that the service time of the first waiting phase is the residual time of the interdeparture process of the second priority customer in the computer service centre, because the initial arrival of a blocked transaction in the waiting centre is independent of the starting point of the departure

process. In the subsequent waiting phases the waiting time is exactly equal to the interdeparture time of the leaving transactions. Thus the service times of the start phase and the subsequent phases in the waiting model are independent and identically distributed with their Laplace-Stieltjes transform of the distribution functions given by $g^*_\zeta(s)$ and $g^*(s)$ respectively.

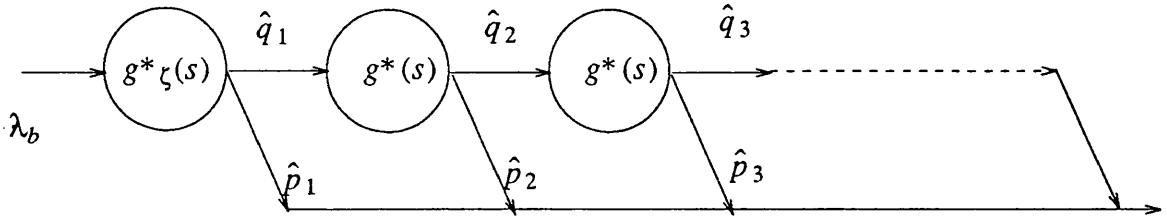


Figure 3.4. A Phase Method of Waiting Time in the Blocked Queue

The above model of waiting time in the blocked queue can be further aggregated into an single infinite-server service centre, which can be easily evaluated in the numerical calculation of the overall model. As shown in figure 3.4, a blocked transaction first enters a start phase which is defined by $g^*_\zeta(s)$. At the end of the first phase, r locks are released by a leaving transaction. Then the blocked transactions either leave the waiting centre with probability $\hat{p}_1$ or enter the second phase of waiting with probability $\hat{q}_1$ and so on. That is upon leaving the jth phase, a blocked transaction enters the $j+1$th phase with probability $\hat{q}_j$ and leaves the blocked queue with probability $\hat{p}_j$. We shall now determine the probability $\hat{p}_j$ of phase j. Let us define L as the total number of granules in the database. (Granular is the unit of data, which can be locked.) Before a lock release, there are N_l locks held on average. At the epoch of the completion of each phase there are r locks released from a pool of N_l locks. Therefore the probability of a transaction to successfully granting r required locks becomes

$$\hat{q}_j = (1-\frac{N_l-r}{L})^r \tag{3.23a}$$

and

$$\hat{p}_j = 1-\hat{q}_j$$

Since all $q_j(j=1, \cdots)$ are equal,

$$\hat{q}_j = \hat{q} = (1-\frac{N_l-r}{L})^r, \tag{3.23b}$$

and

$$\hat{p}_j = 1-\hat{q} \tag{3.23c}$$

Based on Cox's method of phases[18] the service time of the waiting centre has the Laplace-Stieltjes transform function in the form of

$$f_b^*(s) = \hat{p}_1 g_\zeta^*(s) + \hat{q}_1 \hat{p}_2 g_\zeta^*(s) g^*(s) + \hat{q}_1 \hat{q}_2 \hat{p}_3 g_\zeta^*(s) g^*(s) + \cdots$$
$$= \sum_{j=1}^{\infty} \left[\prod_{i=1}^{j-1} \hat{q}_i \right] \hat{p}_j g_\zeta^*(s) [g^*(s)]^{j-1} \tag{3.24}$$

Substituting (3.23a), (3.23b) and (3.23c) into equation (3.24), it becomes

$$f_b^*(s) = \sum_{j=1}^{\infty} \hat{q}^{j-1} (1-\hat{q}) g_\zeta^*(s) [g^*(s)]^{j-1}$$
$$= \frac{(1-\hat{q}) g_\zeta^*(s)}{1-\hat{q}\, g^*(s)}$$

Substituting (3.21) into the above equation, it can be rewritten as

$$f_b^*(s) = \frac{(1-\hat{q})(1-g^*(s))}{\mu_\sigma s (1-\hat{q}\, g^*(s))}$$

Since the mean interdeparture time of the leaving transactions equals to the mean interarrival time of the transactions, i.e. $\lambda_0 = 1/\mu_\sigma$, we have

$$f_b^*(s) = \frac{\lambda_0 (1-\hat{q})(1-g^*(s))}{s(1-\hat{q}\, g^*(s))} \tag{3.25}$$

The mean service time of the waiting centre is given by

$$\mu_b = -\frac{d}{ds} f_b^*(s) \Big]_{s=0} \tag{3.26}$$

Now we need to to obtain the probabilities of lock conflict, when a transaction first acquire locks. Two parameters are used to define the lock conflict rates: q, the probability of successfully granting one lock, and p_b, the probability of lock conflict upon requesting r locks. We also have

$$q_b = 1 - p_b \tag{3.27}$$

Under steady state, the number of locks held is taken as the mean number of transactions in execution state, i.e. $\bar{n}^{(2)}$, multipled by the mean number of locks required by each transaction. Thus the total number of locks held in the system is therefore

$$N_l = r \cdot \bar{n}^{(2)} \tag{3.28}$$

Considering a database as a universe containing a total number of L granules with N_l locks already held, a transaction requests one more lock will have a probability

$$\frac{\binom{N_l-1}{L-1}}{\binom{N_l}{L}} \tag{3.29}$$

of being conflict with N_l locks already held. Therefore the probability of successfully granting one lock becomes

$$q = 1 - \frac{\binom{N_l-1}{L-1}}{\binom{N_l}{L}} = 1 - \frac{N_l}{L} = 1 - \frac{r \cdot \bar{n}^{(2)}}{L} \tag{3.30}$$

The probability of a transaction successfully granting all the required r locks can be written as

$$q^r = \left(1 - \frac{r \cdot \bar{n}^{(2)}}{L}\right)^r$$

and the probability of lock conflict upon requesting r locks is therefore

$$\begin{aligned} p_b &= 1 - q^r \\ &= 1 - \left(1 - \frac{r \cdot \bar{n}^{(2)}}{L}\right)^r \end{aligned} \tag{3.31}$$

and

$$\begin{aligned} q_b &= 1 - p_b \\ &= \left(1 - \frac{r \cdot \bar{n}^{(2)}}{L}\right)^r \end{aligned} \tag{3.32}$$

We shall now derive the mean measures of the open database system. A lock manager always receives the first priority service with preemptive queueing discipline. Therefore it has the same characteristics as the $M/G/1$ queue. The mean waiting time of a $M/G/1$ queue is equal to[15]

$$W^{(1)} = \frac{1}{2} \frac{a^{(1)}\mu^{(1)}}{1 - a^{(1)}} \frac{\sigma^{(1)}}{(\mu^{(1)})^2} \tag{3.33a}$$

where $\sigma^{(1)}$ is the second moment of the distribution $H^{(1)}(t)$. Therefore the mean response time of a lock manager is given by

$$R^{(1)} = \frac{1}{2} \frac{a^{(1)}\mu^{(1)}}{1 - a^{(1)}} \frac{\sigma^{(1)}}{(\mu^{(1)})^2} + \mu^{(1)} \tag{3.33b}$$

It is due to Cohen[15] that the mean waiting time of the second priority customer with a preemptive resume

queueing discipline is given by

$$W^{(2)} = \frac{1}{2(1-a^{(1)})(1-a)}\left\{\frac{\sigma^{(1)}}{\lambda^{(1)}} + \frac{\sigma^{(2)}}{\lambda^{(2)}}\right\} \tag{3.34a}$$

It is obvious that the mean response time of a transaction execution customer is equal to the mean waiting time plus the mean service time of the second priority customer. That is

$$R^{(2)} = \frac{1}{2(1-a^{(1)})(1-a)}\left\{\frac{\sigma^{(1)}}{\lambda^{(1)}} + \frac{\sigma^{(2)}}{\lambda^{(2)}}\right\} + \mu^{(2)} \tag{3.34b}$$

Consequently the mean queueing length of a lock manager is given by

$$\bar{n}^{(1)} = a^{(1)} + \frac{(\lambda^{(1)})^2\sigma^{(1)}}{2(1-a^{(1)})} \tag{3.35}$$

and the mean queueing length of a transaction execution customer is given by[15]

$$\bar{n}^{(2)} = \frac{1}{1-a^{(1)}}\left\{a^{(2)} + \frac{1}{2(1-a)\lambda^{(2)}}\left[\frac{\sigma^{(1)}}{\lambda^{(1)}} + \frac{\sigma^{(2)}}{\lambda^{(2)}}\right]\right\} \tag{3.36}$$

Now we have derived all the equations necessary to evaluate the mean measures of the waiting model in an open database system. It should be noted that if the service time distribution of the computer server is exponential, the mean measures obtained from the above equations are exact, if the computer service time is independent and identically distributed, the above results are approximate. Since the approximation is made only on the assumptions of small lock overhead and lock conflict rate, it can be easily justified by most real database systems. Furthermore the assumptions only effect the distribution characteristics of the output process of the first priority customer rather than its means, the evaluation of the overall system performance is accurate at least to the first order level.

As discussed in the introduction, the analytic models of blocking introduced by others only give an approximate estimation of fixed waiting[78,77]. In this section we have derived the model for not only evaluating the scheduled waiting in the blocked queue with two priority levels but also providing accurate mean and distribution fuction of waiting time in the blocked queue. We have introduced a method which can solve one of the most difficult and fundamental problems of blocking, i.e. evaluating the interdeparture process of transaction execution. By obtaining the characteristic of the interdeparture process, the waiting time in the blocked queue is evaluated by using the method of phases. The evaluation therefore has taken many more fundamental factors into account, such as the stochastic characteristics of the transactions, the service time and interarrival times of the transactions, the queueing discipline, the collision resolution algorithm, the structure of locks, etc. It represents the blocking phenomenon much more accurately. It can be seen in the validation section that the analytic model produces a very accurate ag reement with the simulation results.

3.4 The Scheduled Waiting Model in a Closed System

A closed database is a system with a fixed number of transactions repeatedly competing for hardware and data resources. The hardware resources are cpu and i/o devices; and the data resource is a software resource defined as a collection of conceptual data divided by a number of granules.

For the closed system with two priority levels, even if the service time distribution is exponential the conventional queueing retwork method introduced by Baskett[6] is still not applicable. We need to introduce novel method to solve the problem. In this section we shall introduce an approximation method based on the open database model introduced in the previous section.

In order to compare our analytic results with the simulation results obtained by Ries and Stonebraker[68] we built a closed database system according to the specification given by Ries and Stonebraker. In their model the computer system is composed of a cpu and an i/o device. They can be respectively modeled as the single server service centres with two priority level FCFS queueing discipline. In the closed database each transaction starts with receiving cpu and i/o services to request locks. If all required locks are granted, the transaction succeeds with requesting cpu and i/o services for transaction execution. Otherwise, it goes to a blocked queue to wait according to the scheduled waiting collision resolution algorithm. After completing the execution, the transaction releases all its locks and starts again. Here we suppose the think time is zero. However it is not difficult to include it. As assumed before, the lock manager has preemptive power over transaction management for the cpu and i/o resources.

The whole system illustrated in figure 3.5 can be modeled by a closed queueing network with a total number of transactions equal to N, where N can also be interpreted as the multiprogramming level.

We shall now derive the throughputs of the locking model shown in figure 3.6, where the throughput at c_l, c_e and b are $\lambda^{(1)}$, $\lambda^{(2)}$ and λ_b respectively. Let us define $\lambda_{cpu}^{(i)}$ and $\lambda_{io}^{(i)}$ as the interarrival rate of the ith priority customer at cpu and io service centre respective and λ_0 is the overall throughput of the closed system. It can be seen that under steady state

$$\lambda_{cpu}^{(1)} = \lambda_{io}^{(1)} = \lambda_0 + \lambda_b \tag{3.37a}$$

$$\lambda_{cpu}^{(2)} = \lambda_{io}^{(2)} = \lambda_0 \tag{3.37b}$$

$$\lambda_b = p_b \lambda_{io}^{(1)} \tag{3.37c}$$

$\lambda_{cpu}^{(1)}$, $\lambda_{io}^{(1)}$ and λ_b can be rewritten in terms of λ_0 as

$$\lambda_{cpu}^{(1)} = \lambda_{io}^{(1)} = \frac{\lambda_0}{1-p_b} \tag{3.37d}$$

and

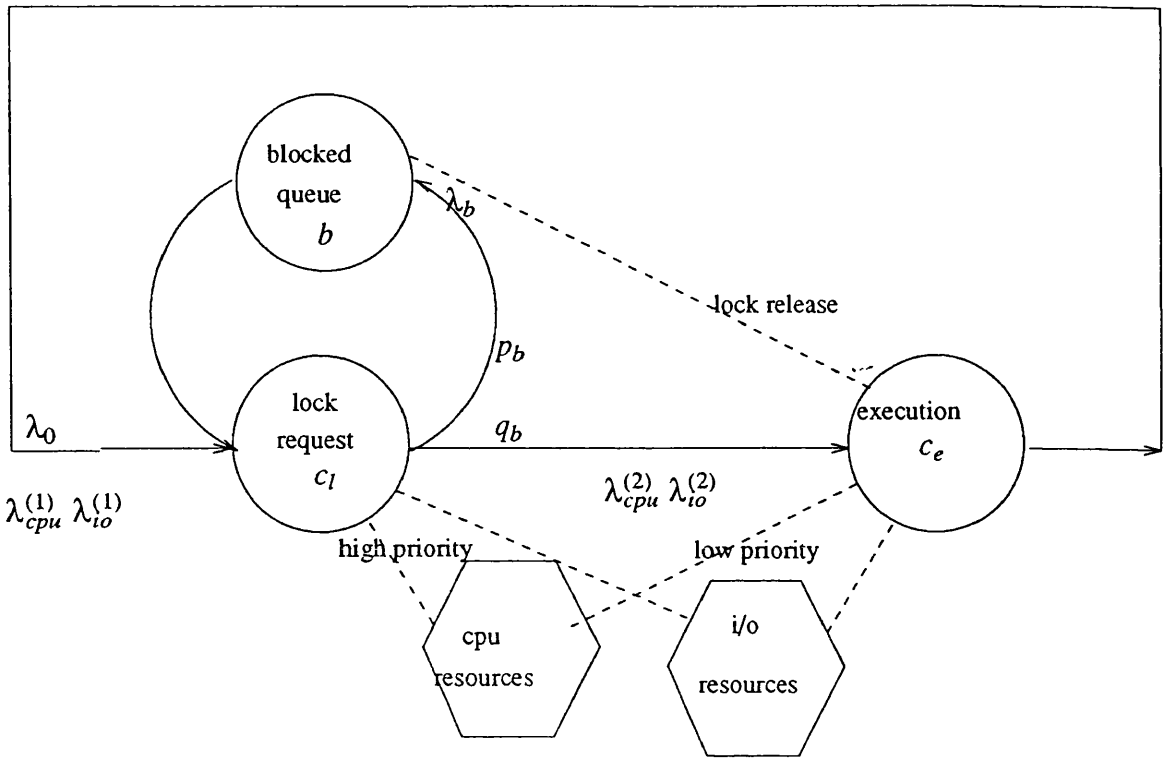


Figure 3.5. Locking Model in a Closed System

$$\lambda_b = \frac{p_b \lambda_0}{1-p_b} \tag{3.37e}$$

Since lock request and transaction execcution share the same computer resources, i.e. cpu and i/o devices, as shown in figure 3.5, we can reconstruct the locking model into a queueing network model with priority queueing discipline as shown in figure 3.6. The cpu and i/o resources can be represented as single server service centres. Lock request is served with first class priority; while transaction execution is served with a second class priority. Since the waiting in the blocked queue does not require computer resources, it can be modeled as an infinite-server service centre. That is each blocked transaction is served immediately for a period determined by the availability of locks.

Let us further define the e_i as the relative throughputs of service centre i in respect to the throughput of the closed system, i.e. λ_0. We have

$$\lambda_i = e_i \lambda_0 \tag{3.38}$$

Substituting the above equation into equations (3.37b), (3.37d) and (3.37e) we have

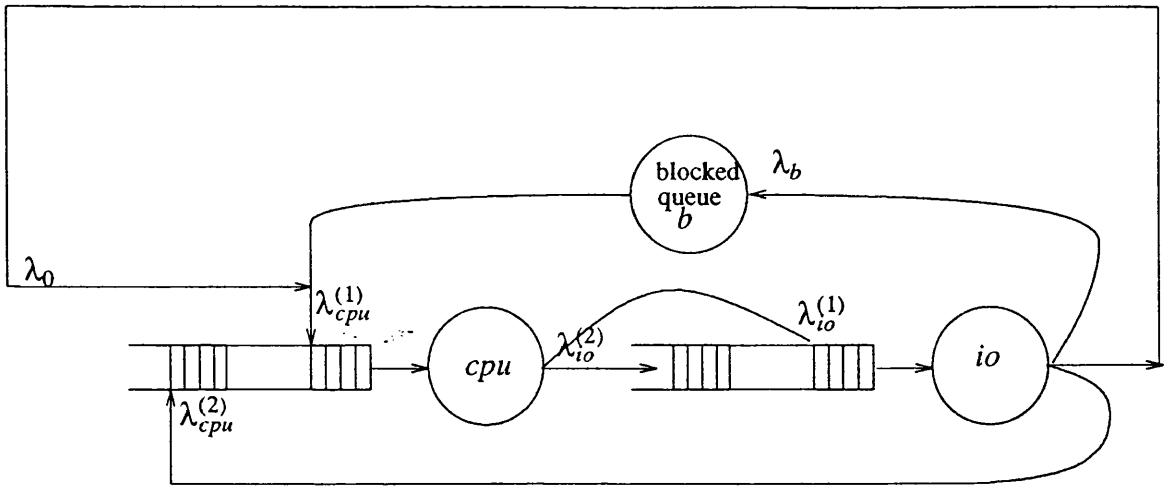


Figure 3.6. Queueing Network Model of a Closed system

$$e_{cpu}^{(2)} = e_{io}^{(2)} = 1 \tag{3.39a}$$

$$e_{cpu}^{(1)} = e_{io}^{(1)} = \frac{1}{1-p_b} \tag{3.39b}$$

$$e_b = \frac{p_b}{1-p_b} \tag{3.39c}$$

We shall apply the method introduced in section 3.3 in the open database system to evaluate the waiting time in the blocked queue. In the closed system the stochastic nature of the blocking process is determined by the departure process of the i/o transaction execution centre. As described previously in figure 3.6, transactions are executed in the service centers with second priority; while lock requests are processed with first priority. Due to the complexity of the closed queueing network with priority queueing discipline, we assume that the service times are exponentially distributed. In this section we mainly concentrate on solving the problem of priority queueing of a closed database system. The Laplace-Stieltjes transform of the distribution function of the inter-departure process of the leaving transactions can be obtained from equation (3.16a); and the Laplace-Stieltjes transform of the distribution density function of the blocking delay in the closed system can be obtained from equation (3.25).

We shall first prove that under the condition of low lock overhead, i.e. $a^{(1)} \to 0$ and exponential service time distribution of cpu and i/o servers, the inter-departure distribution of the second class priority customer is approximately exponentially distributed with mean equal to $1/\lambda^{(2)}$. That is

$$\hat{G}(t) = 1 - e^{-\lambda^{(2)} t} \tag{3.40}$$

We first obtain the busy period distribution of the first class priority customer. From equation (3.6) we have

$$\mu^{(1)}(s,1) = \frac{1}{2a^{(1)}}[\mu^{(1)}s+a^{(1)}+1-\sqrt{(\mu^{(1)}s+a^{(1)}+1)^2-4a^{(1)}}]$$
$$= \frac{1}{2a^{(1)}} \cdot \frac{(\mu^{(1)}s+a^{(1)}+1)^2-(\mu^{(1)}s+a^{(1)}+1)^2+4a^{(1)}}{(\mu^{(1)}s+a^{(1)}+1)^2+(\mu^{(1)}s+a^{(1)}+1)^2-4a^{(1)}}$$
$$\approx \frac{2}{(\mu^{(1)}s+a^{(1)}+1)^2+(\mu^{(1)}s+a^{(1)}+1)^2}$$
$$= \frac{1}{(\mu^{(1)}s+a^{(1)}+1)^2}$$
$$\approx \frac{1}{\mu^{(1)}s+1} \tag{3.41}$$

We then have

$$\beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\} = \beta^{(2)}\{s+\lambda^{(1)}(1-\frac{1}{\mu^{(1)}s+1})\}$$
$$= \beta^{(2)}\{\frac{s(\mu^{(1)}s+1+a^{(1)})}{\mu^{(1)}s+1}\} \tag{3.42}$$

Based on the assumption of $a^{(1)} \to 0$, we can obtain

$$\beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(2)}(s,1))\} \approx \beta^{(2)}\{s\} \tag{3.43}$$

Since the service centre is assumed to have an exponential service time, it is obvious that

$$\beta^{(2)}\{s\} = \int_0^\infty e^{-st} d_t(1-e^{-t/\mu^{(2)}})$$
$$= \frac{1}{\mu^{(2)}s+1} \tag{3.44}$$

Substituting the above results into equation (3.16a), we obtain

$$\hat{g}^*(s) = \frac{1}{(1-a^{(1)})(\mu^{(2)}s+1)}\left[(1-a)\cdot\frac{\lambda^{(2)}}{\lambda^{(2)}+s}\cdot\frac{1+a^{(1)}/[(\mu^{(1)}s+1)(\mu^{(1)}\lambda^{(2)}+1)]}{1+a^{(1)}/(\mu^{(1)}\lambda^{(2)}+1)}+a^{(2)}\right] \tag{3.45}$$

Noticing $a^{(1)} \to 0$, we can rewrite this equation as

$$\begin{aligned}
\hat{g}^*(s) &\approx \frac{1}{(1-a^{(1)})(\mu^{(2)}s+1)}\left[(1-a)\cdot\frac{\lambda^{(2)}}{\lambda^{(2)}+s}+a^{(2)}\right] \\
&= \frac{1}{(1-a^{(1)})(\mu^{(2)}s+1)}\left[\frac{(1-a^{(1)}-a^{(2)})\lambda^{(2)}+a^{(2)}\lambda^{(2)}+a^{(2)}s}{\lambda^{(2)}+s}\right] \\
&\approx \frac{1}{(1-a^{(1)})(\mu^{(2)}s+1)}\frac{\lambda^{(2)}(1+\mu^{(2)}s)}{\lambda^{(2)}+s} \\
&= \frac{\lambda^{(2)}}{(1-a^{(1)})(\lambda^{(2)}+s)} \\
&\approx \frac{\lambda^{(2)}}{\lambda^{(2)}+s}
\end{aligned}$$

Therefore approximately

$$\hat{G}(t) = 1-e^{-\lambda^{(2)}t}$$

This result is very useful. It means that the output process of the leaving transactions is approximately Poisson; therefore the interarrival process of the transactions in a closed database is always approximately Poisson. We can thus treat the cpu and i/o service centres in the closed system as $M/M/1$ queue with two priority levels. This result can be used in not only scheduled waiting but also fixed waiting. Previous researches only gives an estimation of fixed wiating time on the basis of deduction as in Tay's work. The above result provides the mean and distribution function at the accuracy of first order. The result also has an extremely simple and desirable form which can be easily used in the overall model.

The mean service time of $\hat{g}^*(s)$, i.e. μ_σ, can be directly obtained from equation (3.40). In order to prove the consistency of the approximation, we can also derive the mean time between departures from equation (3.19). We have

$$\begin{aligned}
\mu_\sigma &= \frac{\mu^{(2)}}{1-a^{(1)}}+\frac{1-a}{(1-a^{(1)})^2}\cdot\frac{1}{\lambda^{(2)}+\lambda^{(1)}(1-\mu^{(1)}(\lambda^{(2)},1))} \\
&= \frac{\mu^{(2)}}{1-a^{(1)}}+\frac{1-a}{(1-a^{(1)})^2}\cdot\frac{1}{\lambda^{(2)}}\cdot\frac{\mu^{(1)}\lambda^{(2)}+1}{\mu^{(1)}\lambda^{(2)}+1+a^{(1)}}
\end{aligned}$$

As $a^{(1)}\to 0$, we use

$$\frac{\mu^{(1)}\lambda^{(2)}+1}{\mu^{(1)}\lambda^{(2)}+1+a^{(1)}}\to 1$$

to obtain

$$\mu_\sigma = \frac{\mu^{(2)}}{1-a^{(1)}} + \frac{1-a}{(1-a^{(1)})^2} \cdot \frac{1}{\lambda^{(2)}}$$
$$= \frac{1}{\lambda^{(2)}} \cdot \frac{(1-a^{(1)})^2 + a^{(1)}(1-a)}{(1-a^{(1)})^2}$$
$$= \frac{1}{\lambda^{(2)}} = \frac{1}{\lambda_{io}^{(2)}} \tag{3.46}$$

Now we can evaluate the overall performance model of the closed system. This is a closed queueing network composed of a high priority cpu and i/o and a low priority shadow cpu and i/o service centres with FCFS queueing discipline; and a waiting centre with infinite-server queueing discipline. The cpu and i/o service centres are assumed to be exponentially distributed, while the infinite-server service centres have general service time distribution with their Laplace-Stieltjes transforms given by $f^*(s)$.

Next we shall determine the probability q and p_b which are defined in the previous section. Under steady state, the mean number of locks held is taken as the mean number of transactions in execution state multipled by the mean number of locks required by each transaction. Since the total number of transactions in execution centres is equal to $\bar{n}_{cpu}^{(2)} + \bar{n}_{io}^{(2)}$, the total number of locks held in the system is given by

$$N_l = r \cdot (\bar{n}_{cpu}^{(2)} + \bar{n}_{io}^{(2)}) \tag{3.47}$$

From equation (3.30), the probability of successfully granting one lock becomes

$$q = 1 - \frac{N_l}{L} = 1 - \frac{r \cdot (\bar{n}_{cpu}^{(2)} + \bar{n}_{io}^{(2)})}{L} \tag{3.48}$$

From the above equation the probability of a transaction successfully granting all the required r locks can be derived as

$$q^r = \left[1 - \frac{r \cdot (\bar{n}_{cpu}^{(2)} + \bar{n}_{io}^{(2)})}{L} \right]^r \tag{3.49}$$

and the probability of lock conflict upon requesting r locks can be given by

$$p_b = 1 - q^r$$
$$= 1 - \left[1 - \frac{r \cdot (\bar{n}_{cpu}^{(2)} + \bar{n}_{io}^{(2)})}{L} \right]^r \tag{3.50}$$

We can now solve the problem of the closed queueing network shown in figure 3.6. The numerical solution to the above closed queueing network with $M/M/1$ and IS service centres can be found in Sevcik's approximation method in which the priority queueing centres can be treated as the single server service centres[71]. A service centre is divided into two. One is the high priority server; the other is the low priority server called shadow server. Because the queueing discipline is preemptive resume, the high priority customers will not be delayed by the low priority customers. Therefore the service capacity of the

high priority server is the same as the original server. That is

$$C^{(1)} = C \ (unit/s) \tag{3.51}$$

If the service demand of the high class customer is denoted as $D^{(1)}$ $(unit)$, the mean service rate of the high class customer is equal to

$$\mu^{(1)} = \frac{C^{(1)}}{D^{(1)}} = \frac{C}{D^{(1)}} \tag{3.52}$$

The service capacity of low class customer is reduced from the total capacity by the amount used by the high class customers. Therefore

$$C^{(2)} = C - C \cdot U^{(1)} \tag{3.53}$$

where $U^{(1)}$ is the utility of high class customer. The mean service rate of the shadow server of low class customers is equal to

$$\mu^{(2)} = \frac{C^{(2)}}{D^{(2)}} = \frac{C(1-U^{(1)})}{D^{(2)}} \tag{3.54}$$

Without losing generality, the total system capacity is set to $C=1$ $(unit/s)$. Thus

$$\mu^{(1)} = \frac{1}{D^{(1)}} \tag{3.55a}$$

$$\mu^{(2)} = \frac{1-U^{(1)}}{D^{(2)}} \tag{3.55b}$$

By using this method the cpu service centre can be decomposed into two single server service centres $cpu^{(1)}$ and $cpu^{(2)}$ with FCFS queuing discipline and mean service times equal to

$$\mu_{cpu}^{(1)} = \frac{1}{D_{cpu}^{(1)}} ; \quad \mu_{cpu}^{(2)} = \frac{1-U_{cpu}^{(1)}}{D_{cpu}^{(2)}} \tag{3.56}$$

Similarly the i/o service centre is decomposed with mean service times given by

$$\mu_{io}^{(1)} = \frac{1}{D_{io}^{(1)}} ; \quad \mu_{io}^{(2)} = \frac{1-U_{io}^{(1)}}{D_{io}^{(2)}} \tag{3.57}$$

where $D_{cpu}^{(1)}$ and $D_{io}^{(1)}$ are the service demand of lock management, and $D_{cpu}^{(2)}$ and $D_{io}^{(2)}$ are the service demand of transaction execution. The blocking centre can be modeled by using an infinite-server service centre with service time μ_b given by equation (3.26).

The mean response times of cpu and i/o service centres are given by

$$R_i(N) = \mu_i(1+\bar{n}_i(N-1)) \quad i \in \{FCFS \ station\} \tag{3.58}$$

and that of the waiting centre is given by

$$R_i(N) = \mu_\sigma \quad i \in \{infinite-server\ station\} \tag{3.59}$$

other mean measures for all the service centres are given by

$$\lambda_0(N) = \frac{N}{\sum_{i=1}^{m} e_i R_i(N)} \tag{3.60}$$

$$\bar{n}_i(N) = \lambda_0(N) e_i R_i(N) \tag{3.61}$$

$$U_i = e_i \lambda_0 \mu_i \tag{3.62}$$

where m is the number of service centres in the system.

The value of the relative throughput and service time of each service centre can be obtained by solving equations (3.38), (3.39a-c) and (3.46) respectively using the iterative method. The corresponding numerical algorithm is given as follows:

1. Initialization:

$$\mu_{cpu}^{(1)} = \frac{1}{D_{cpu}^{(1)}}; \quad \mu_{io}^{(1)} = \frac{1}{D_{io}^{(1)}}$$

$$\mu_{cpu}^{(2)} = \frac{1}{D_{cpu}^{(2)}}; \quad \mu_{io}^{(2)} = \frac{1}{D_{io}^{(2)}}; \quad \mu_b = \mu_{io}^{(2)}$$

$$q = q_0$$

2. Iteration

   Solve p_b from

$$p_b = 1 - q^r$$

   Define / redefine e_i

$$e_{cpu}^{(1)} = e_{io}^{(1)} = \frac{1}{1-p_b}$$

$$e_{cpu}^{(2)} = e_{io}^{(2)} = 1;$$

$$e_b = \frac{p_b}{1-p_b};$$

   Call core subroutine to obtain $n_{cpu}^{(2)}$, $n_{io}^{(2)}$ and λ_0.

   Redefine μ_i:

$$\mu_{cpu}^{(2)} = \frac{1-u_{cpu}^{(1)}}{D_{cpu}^{(2)}}; \quad \mu_{io}^{(2)} = \frac{1-u_{io}^{(1)}}{D_{io}^{(2)}};$$

$$\mu_b = \lambda_0;$$

Obtain a new q from

$$q = \frac{1-r(n_{cpu}^{(2)}+n_{io}^{(2)})}{L};$$

If $|q-q_0| > \varepsilon$ then repeat 2 else exit.

It should be noted that the solution of the above equations lies upon the core subroutine of the recursive solution of the mean value analysis method. Core algorithm of mean value analysis to obtain $\bar{n}_{cpu}^{(1)}$, $\bar{n}_{cpu}^{(2)}$, $\bar{n}_{io}^{(1)}$, $\bar{n}_{io}^{(2)}$ and λ_o is given by Lavenberg[42].

In this section, we have extended the waiting model to closed system by introducing an approximation method. We have obtained an interesting and useful approximation result, i.e. under small lock overhead, the departure process of the transaction exection centre is approximately Poisson and its mean only depends on the overall throughput of the closed queueing network. This new result not only simplifies the evaluation but also makes it possible to treat the service centre as $M/M/1$ queue with two priority levels in a closed queueing network.

3.5 The Scheduled Waiting Model With Multiple Classes

In the previous sections we have studied the behavior of the waiting model with only single class transactions. For existing commercial database systems, there are usually more than one type of transaction. Therefore it is desirable to release the single class restriction when evaluating the system performance. This, however, is difficult as widely recognized. It is because that the queueing theory for multiple class customers with two priority levels is not available. Here we will introduce a novel approximation method based on the methods introduced in the previous two sections and the mean value analysis method introduced by Bard[5].

The multiclass model differs from single class model in the way that the service demands and lock requests for different classes of transactions are not identical. This can be modeled by our multiclass queueing network model when we factor in the service times, the lock conflict rates and blocking delays of different transaction classes. Instead of using a mean conflict rate and blocking delay for all the transaction classes, we introduce a method to obtain lock conflict rates from the correspondent number of locks required by each transaction. Furthermore each transaction class reserves its own waiting centre to model the waiting time in the blocked queue.

We shall derive the multiclass model from the single class model illustrated in figure 3.6. The corresponding multiclass model is shown in figure 3.7. The notions of multiclass model in figure 3.7 are

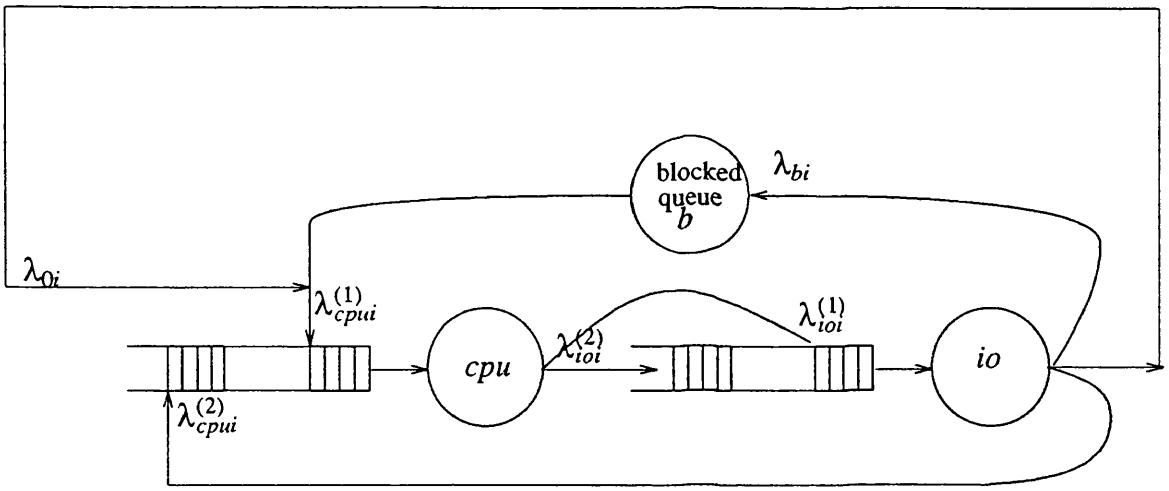


Figure 3.7. Locking Model with Multiple Classes

quite similar to those of figure 3.6. Suppose there are c transaction classes in the system. We shall use the subscript ki to define the mean value of class i transaction in service centre k. Thus $\mu_{ki}, e_{ki}, \bar{n}_{ki}, R_{ki}$, and U_{ki} are respectively defined as the mean service time, the relative throughput, the mean queueing length, the mean response time and the mean utilization of class i transaction in service centre k.

A transaction of class i first enters $cpu^{(1)}$ and $io^{(1)}$ service centres requiring lock service of $\mu^{(1)}_{cpui}$ and $\mu^{(1)}_{ioi}$; it then enters waiting centre bi with probability p_{bi} and reenters service centres $cpu^{(2)}$ and $io^{(2)}$ for transaction execution with probability $1-p_{bi}$, where p_{bi} is the probabilty of lock conflict upon requesting r_i locks from class i customer. Upon entering the waiting centre bi the transaction is served by an infinite-server service center with mean service time μ_{bi}. Upon entering the $cpu^{(2)}$ and $io^{(2)}$ for transaction execution it receives $\mu^{(2)}_{cpui}$ and $\mu^{(2)}_{ioi}$ service. The whole transaction with class i is thus completed and will start again in the closed system.

Now we shall derive the equations to solve this locking model with multiple transaction classes. Firstly the probability of successfully requesting one lock from class i customer needs to obtained. Under steady state, the number of locks held by a class i transaction is equal to the number of locks required by the transaction multiplied by the mean number of class i transactions in the execution state. Thus the number of locks held by class i transaction is given by

$$N_{li} = r_i \cdot (\bar{n}^{(2)}_{cpui} + \bar{n}^{(2)}_{ioi}). \tag{3.63}$$

And the total number of locks held by all the transactions is therefore

$$N_l = \sum_{i=1}^{c} N_{li} \tag{3.64}$$

where c is the total number of transaction classes. The probability of conflict upon requiring one lock is therefore equal to the total number of locks held divided by the total number of granules in the system. Thus the probability of successfully granting one lock can be given by

$$q = 1 - \frac{N_l}{L}$$
$$= 1 - \frac{\sum_{i=1}^{c} r_i \cdot (\bar{n}_{cpui}^{(2)} + \bar{n}_{ioi}^{(2)})}{L}$$

A transaction of class i requires to grant r_i locks before starting execution. The probability of successfully granting r_i locks is q^{r_i}. Therefore the probability of lock conflict upon requesting r_i locks from class i transaction is

$$p_{bi} = 1 - q^{r_i}$$
$$= 1 - \left[1 - \frac{\sum_{i=1}^{c} r_i \cdot (\bar{n}_{cpui}^{(2)} + \bar{n}_{ioi}^{(2)})}{L}\right]^{r_i} \tag{3.65}$$

In the blocked queue transactions of different classes wait for different time intervals. Therefore c parallel infinite-server service centres are required to model the waiting delays. Let μ_{bi} denote the mean waiting time of the class i transaction in the waiting centre. Now we shall determine the Laplace transform of the waiting time distribution of the blocked transactions of class i customer. It can be easily derived from equation (3.23a) that the probability of a class i transaction successfully getting r_i required locks in the blocked queue is given by

$$\hat{q}_i = \left[q + \frac{\bar{r}}{L}\right]^{r_i}$$

and

$$\hat{p}_i = 1 - \hat{q}_i$$

where

$$\bar{r} = \sum_{i=1}^{c} r_i$$

Therefore the Laplace-Stieltjes transform of the blocking delay distribution of class i transaction becomes

$$f_{bi}^{*}(s) = \sum_{j=1}^{\infty} \hat{q}_i^{j-1} \hat{p}_i g_{\zeta}^{*}(s)[g^{*}(s))]^{j-1}$$
$$= \frac{\lambda_0(1-\hat{q}_i)(1-g^{*}(s))}{s(1-\hat{q}_i g^{*}(s))}$$

Following the method introduced in section 3.3, we can obtain the overall multiclass model. The relative throughputs for class i transactions at each service centre in figure 3.7 are given by

$$e_{cpui}^{(2)} = e_{ioi}^{(2)} = 1 \tag{3.66a}$$

$$e_{cpui}^{(1)} = e_{ioi}^{(1)} = \frac{1}{1-p_{bi}} \tag{3.66b}$$

$$e_{bi} = \frac{p_{bi}}{1-p_{bi}} \tag{3.66c}$$

The overall model can be solved by applying Sevcik's shadow server priority queueing network method. Suppose for class i transaction $D_{cpui}^{(1)}$ and $D_{ioi}^{(1)}$ are the service demand of lock management, and $D_{cpui}^{(2)}$ and $D_{ioi}^{(2)}$ are the service demands of transaction execution. The mean service time of class i transaction at each service centre can be given by

$$\mu_{cpui^{(1)}} = \frac{1}{D_{cpui}^{(1)}} ; \quad \mu_{ioi}^{(1)} = \frac{1}{D_{ioi}^{(1)}} \tag{3.67}$$

$$\mu_{cpui^{(2)}} = \frac{1-U_{cpu}^{(1)}}{D_{cpui}^{(2)}} ; \quad \mu_{ioi}^{(2)} = \frac{1-U_{io}^{(1)}}{D_{ioi}^{(2)}} \tag{3.68}$$

where

$$U_{cpu}^{(1)} = \sum_{i=1}^{c} U_{cpui}^{(1)} \tag{3.69}$$

and

$$U_{io}^{(1)} = \sum_{i=1}^{c} U_{ioi}^{(1)} \tag{3.70}$$

Applying Bard's approximation method of multiclass queueing network, we can obtain the mean measures by using iteration method[5].

1. Throughput:

$$\lambda_{ki}(\underline{N}) = \lambda_{0i}(\underline{N}) e_{ki}; \tag{3.71}$$

2. Utilization:

$$U_{ki}(\underline{N}) = \mu_{ki}\lambda_{ki}(\underline{N}); \tag{3.72}$$

3. Mean queue length:

$$\bar{n}_{ki}(\underline{N}) = \lambda_{ki}(\underline{N})R_{ki}(\underline{N}); \tag{3.73}$$

4. Mean response time:

$$R_{ki}(\underline{N}) = \begin{cases} \mu_{ki} & \text{infinite-server station} \\ \mu_{ki} + \sum_{j=1}^{c} \mu_{kj}\bar{n}_{kj}(\underline{N}-\underline{1}_i) & \text{FCFS station} \end{cases} \tag{3.74}$$

where $\underline{N} = (N_1, N_2, \cdots, N_c)$ is the total number of transactions in each class, and $\underline{1}_i = (0, \cdots, 1, \cdots, 0)$ is a vector of a 1 in the ith component and zeros in the rest components. The solution of the system can be obtained iteratively.

In this section, we have extended the waiting model to multiclass transactions by introducing different lock conflict rates, lock sizes, block delays and service times. An interesting phenomenon has been observed. This is the transactions with smaller service time circulate within the closed network more quickly than others. This multiclass model is superior to other models in te way that it not only inheritates all the characteristics of the scheduled waiting model of single class but also includes many important characteristics of multiclass transactions; thus provides an accurate evaluation model for multiclass transactions, while in Tay's model. only fixed waiting is considered. And in Thomasian's model, only a much simplified iterative solution is introduced.

3.6 Valiation by Simulation

The main purpose of this section is verify the analytic model with the simulation results. Figure 3.8 illustrates the simulation model used by Ries and Stonebraker. The detailed description of the simulation model can be found in their paper[68]. They simulated locking in a centralized database system with the following disciplines:

1. Scheduled waiting collision resolution algorithm is used;
2. Locking operations have preemptive resume priority over transaction execution operations;
3. Locking is based on lock script;
4. Transactions are of multiclass type.

The results obtained from the multiclass analytic model in the closed system are compared with the simulation results obtained by Ries and Stonebraker[68]. Three groups of results are compared, i.e. mean

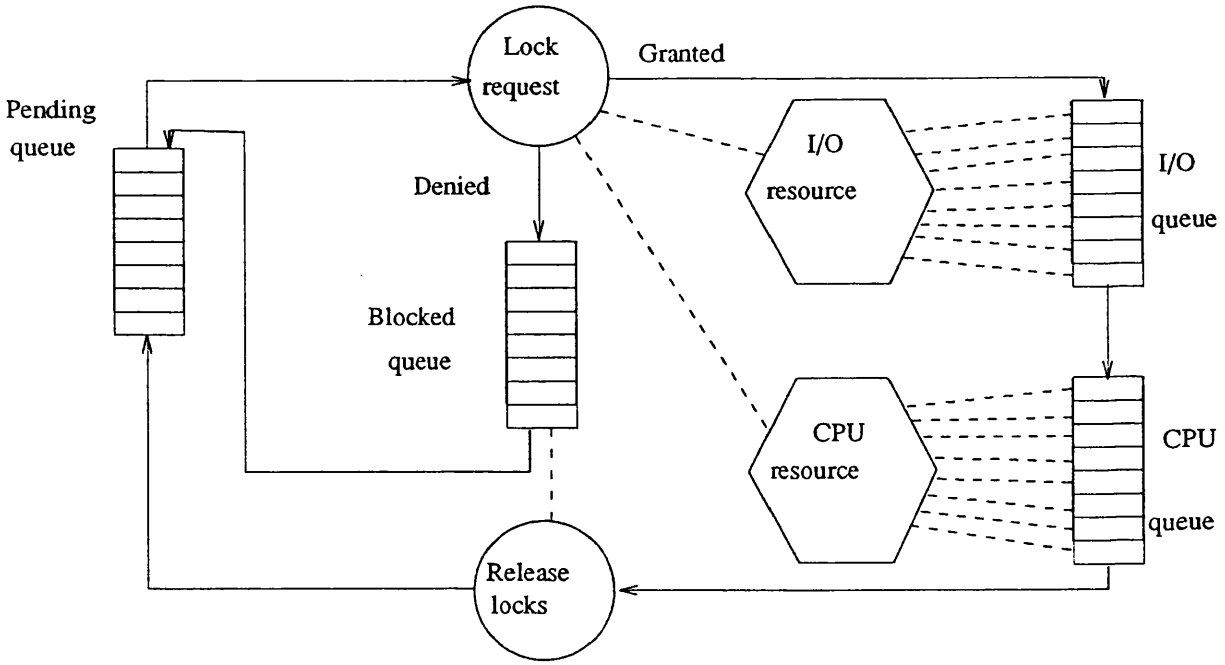


Figure 3.8. Ries and Stonebraker's Simulation Model

response time of transactions, useful cpu time and useful i/o time vs. number of granulars, as shown in figures 3.9, 3.10, 3.11 and 3.12. For the useful cpu and i/o times the differences are mostly around 2% as shown in table 3.2 and 3.3. Only at the two extreme cases, i.e. $no\_of\_granula=1$ and $no\_of\_granula=2500$ and 5000 the difference becomes greater. This is because that in the former case the feedback effect becomes greater, while in the latter case the lock overheads at i/o are no longer small. For the mean response time the differences given in table 3.1 are quite small for small number of granules and becomes slightly bigger for larger granulars.

The analytic results are very close to the simulation results, which validates the analytic model of the scheduled waiting and verifies the assumptions and the approximations made in the analytic model. Ries and Stonebraker's simulation model is built at a very detailed level, such as using lock script, prioritizing lock operations, using scheduled waiting collision resolution algorithm and simulating multiclass transactions. Since our analytic model factors in accurately all these detailed characteristics of the system, a very good agreement between the two models has been achieved. Our analytic model is therefore well validated.

3.7 Summary

In this chapter, we have introduced an analytic model of blocking with scheduled waiting in a centralized database. Many novel modeling methods have been introduced. The model reveals the distribution of the departure process of the second priority customer in a $M/G/1$ queue with two priority

levels. With this crucial result, the scheduled waiting in a blocked queue can be modeled succesfully. The model is further extended to a closed database and multiple transaction classes. The accuracy of the analytic model has been well validated by Ries and Stonebraker's simulation result. Very close agreement has been achieved.

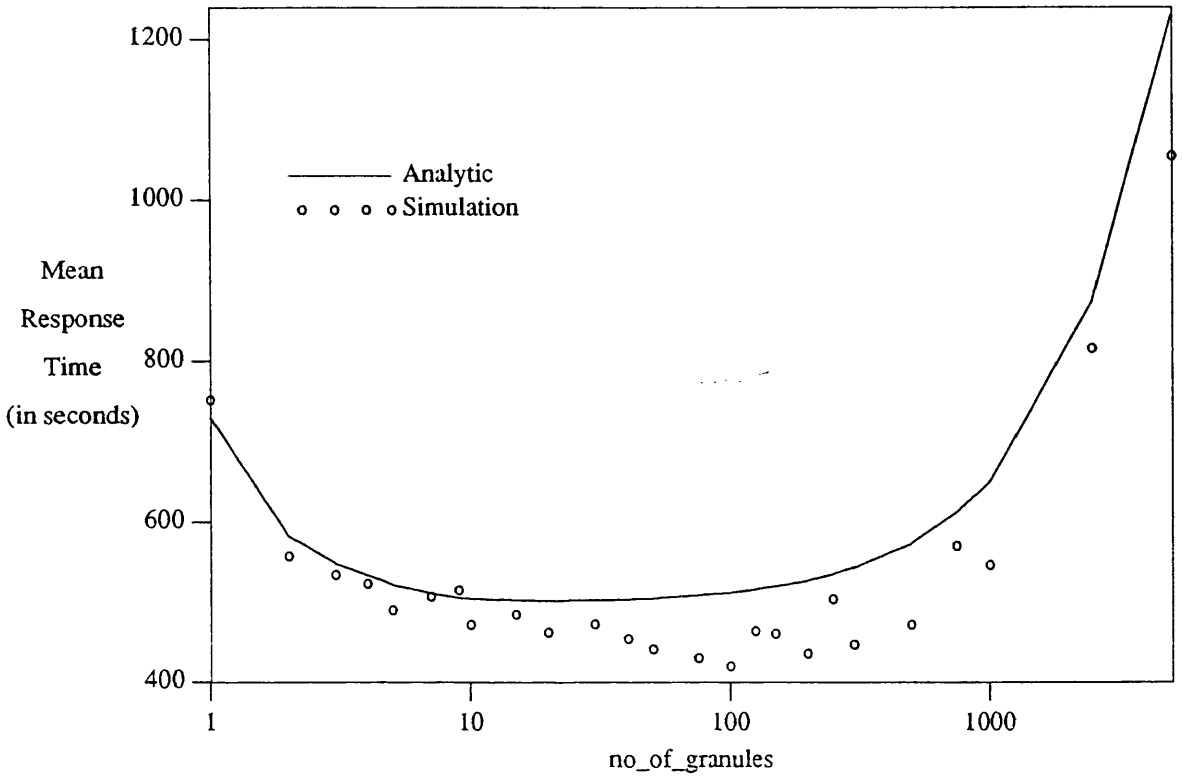


Figure 3.9. Mean Response Time of the Multiclass System

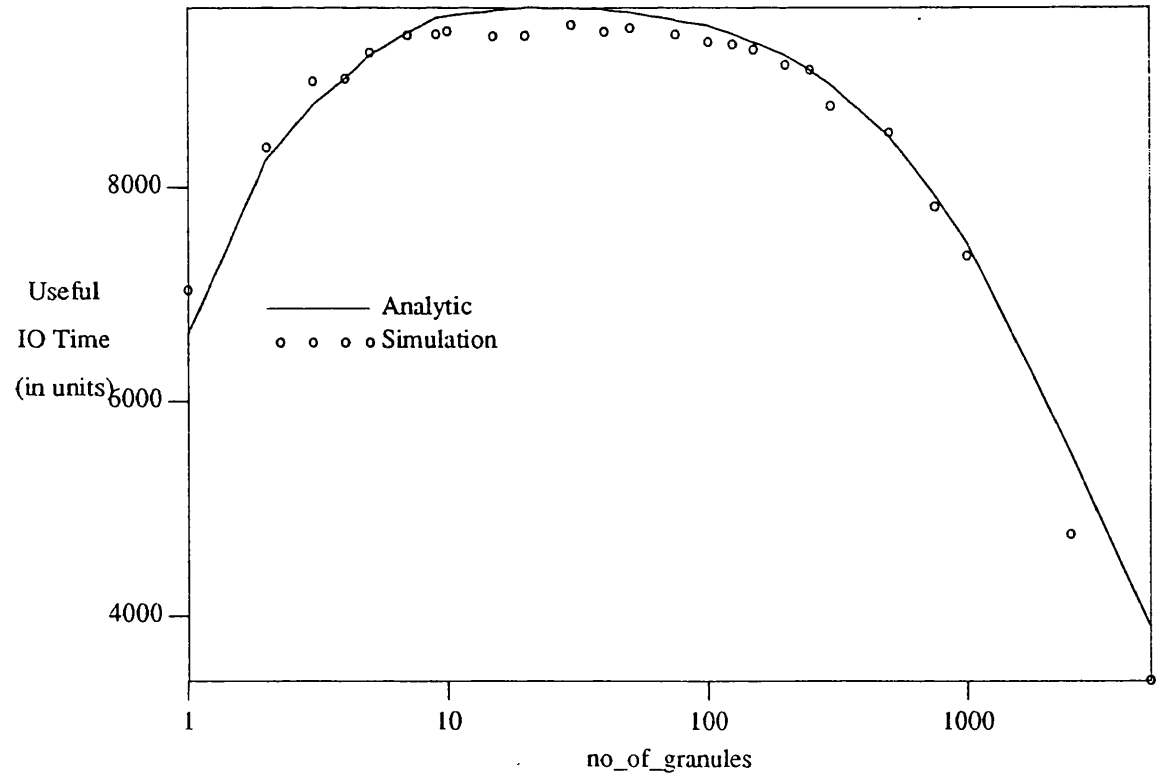


Figure 3.10. Useful IO Time of the Multiclass System

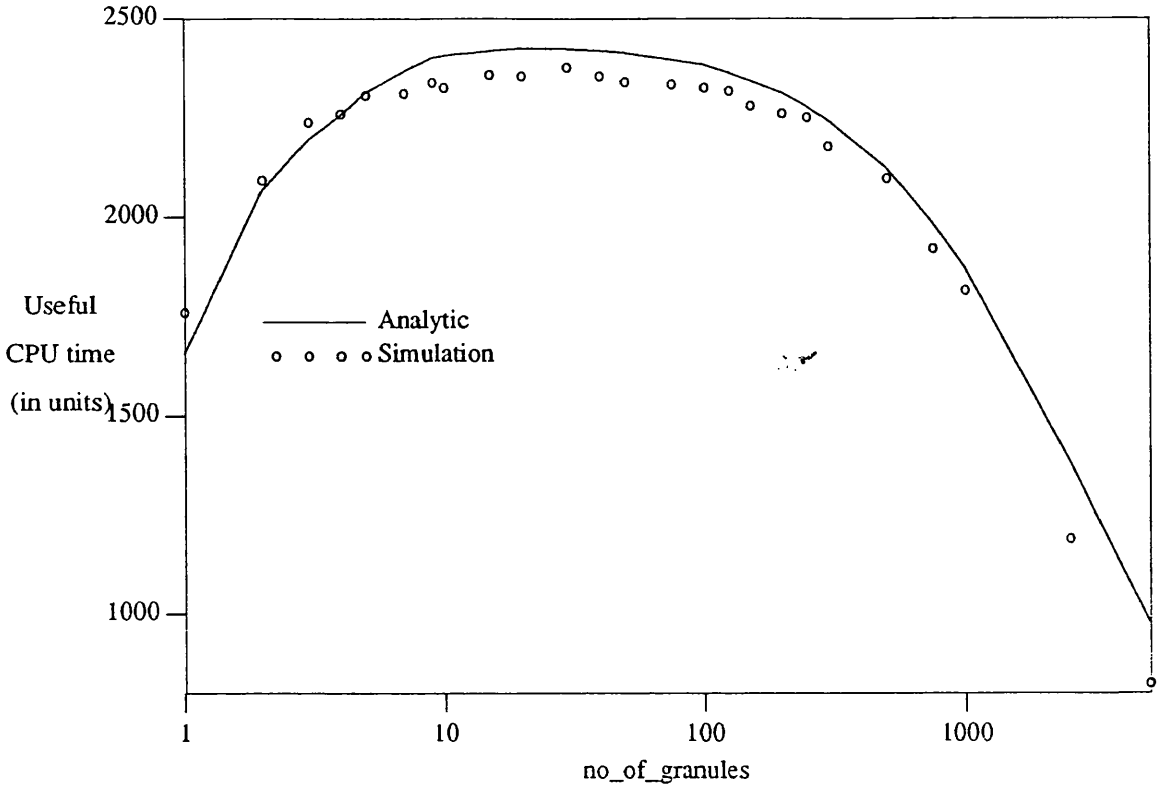


Figure 3.11. Useful CPU Time of the Multiclass System

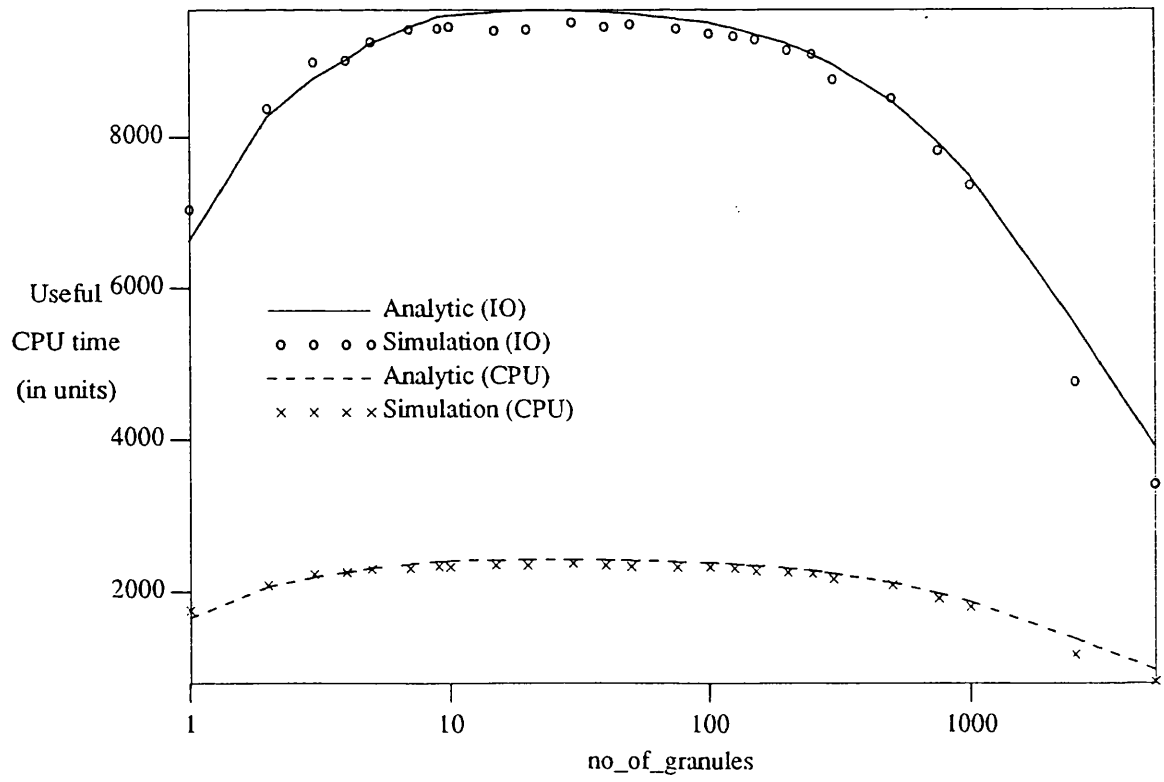


Figure 3.12. Useful CPU and IO Time of the Multiclass System

Table 3.1. The Comparison of Mean Response Time of the Multiclass System

no_of_granules	simulation	analytic	%differences
1	751.9	730.6	2.9
2	557.2	582.1	4.3
3	534.4	548.4	2.6
4	523.1	533.8	2.0
5	490.3	521.5	6.0
7	506.7	511.2	0.9
9	515.1	505.0	2.0
10	472.3	504.0	6.3
15	484.2	502.5	3.6
20	462.7	501.7	7.8
30	472.7	502.3	5.9
40	454.2	503.5	9.8
50	441.5	504.8	12.5
75	430.5	508.9	15.4
100	420.4	511.6	17.8
125	463.3	515.9	10.2
150	460.4	520.1	11.5
200	435.7	527.0	17.3
250	504.0	535.0	5.8
300	447.1	542.9	17.6
500	472.1	573.2	17.6
750	570.1	612.5	6.9
1000	546.0	648.9	15.9
2500	815.8	873.0	6.6
5000	1055.0	1231.1	14.3

Table 3.2. The Comparison of Useful I/O Time of the Multiclass System

no_of_granules	simulation	analytic	%differences
1	7041.9	6618.4	6.4
2	8377.0	8264.2	1.4
3	9002.3	8772.3	2.6
4	9030.3	9023.5	0.1
5	9273.9	9249.8	0.3
7	9438.5	9463.2	0.3
9	9449.1	9601.2	1.6
10	9476.2	9623.5	1.5
15	9425.6	9669.6	2.5
20	9438.0	9698.6	2.7
30	9534.3	9693.9	1.6
40	9472.7	9675.7	2.1
50	9504.1	9652.3	1.5
75	9448.4	9577.8	1.4
100	9378.3	9528.7	1.6
125	9351.7	9450.0	1.0
150	9304.1	9371.8	0.7
200	9159.7	9248.5	1.0
250	9110.5	9107.9	0.0
300	8768.2	8971.7	2.3
500	8517.2	8489.3	0.3
750	7820.6	7933.8	1.4
1000	7359.8	7480.1	1.6
2500	4764.2	5530.7	13.9
5000	3408.6	3913.6	12.9

Table 3.3. The Comparison of Useful CPU Time of the Multiclass System

no_of_granules	simulation	analytic	%differences
1	1759.9	1654.6	6.4
2	2091.9	2066.1	1.2
3	2237.4	2193.1	2.0
4	2258.9	2255.9	0.1
5	2304.9	2312.5	0.3
7	2309.9	2365.8	2.4
9	2337.4	2400.3	2.6
10	2324.9	2405.9	3.4
15	2358.4	2417.4	2.4
20	2354.9	2424.7	2.9
30	2377.5	2423.5	1.9
40	2354.9	2418.9	2.6
50	2339.9	2413.1	3.0
75	2332.5	2394.5	2.6
100	2324.9	2382.2	2.4
125	2316.5	2362.5	1.9
150	2280.0	2343.0	2.7
200	2260.0	2312.1	2.3
250	2250.0	2277.0	1.2
300	2177.5	2242.9	2.9
500	2097.5	2122.5	1.2
750	1920.0	1983.5	3.2
1000	1815.0	1870.0	2.9
2500	1190.0	1385.3	14.1
5000	825.0	978.4	15.7

Chapter 4. Concurrency Control Model of DDB

A distributed database system inherits most of the characteristics of centralized databases such as locking, parallel transaction execution and multiprogramming. However some additional features of distributed databases, such as concurrency control and locking among distributed transactions, significantly increase the complexity of the systems. In this chapter we aim to define an analytic model which can represent the characteristics of the distributed database concurrency control and locking mechanisms. In a distributed database system transactions are controlled by a concurrency control protocol to guarantee the consistency and integrity of the system. The basic idea of a concurrency control protocol is to provide a communication and control agreement between a coordinating process and a number of participating processes. There is a wide variety of concurrency control protocols such as the basic two phase locking (2PL), the primary copy 2PL, the majority consensus 2PL, centralized 2PL etc[8, 12]. In this chapter we shall first study the most well known two phase locking protocol, i.e. the basic 2PL. The other three 2PL protocols will be studied in the next chapter.

Section 4.1 presents a formal specification of the basic 2PL and its communication structure. Section 4.2 introduces a novel method to systematically define a two phase commit protocol with access pattern matrix, communication flow matrix, arrival rate matrix and Markov chain matrix. In section 4.3 a distributed database locking model with fixed waiting collision resolution algorithm is introduced. Section 4.4 introduces an extended diffusion approximation method to model a distributed database with general service time distributions. Section 4.5 extends the method introduced in chapter 3 to model the scheduled waiting in a distributed database. Section 4.6 builds a simulation model for distributed databases and validates the analytic model. In section 4.7 cases of performance results are analyzed and some useful conclusions are drawn at the end.

4.1 System Specification

Locking is strongly associated with distributed databases as with centralized databases. In a two phase locking protocol, transactions request locks in the growing phase before a common decision is reached; and releases locks in the shrinking phase after the implementation of the decision[8]. The basic 2PL is to explicitly detect and prevent conflicts between concurrent operations by implementing a growing phase for transactions to obtain locks and a shrinking phase for transactions to release locks. Transactions, such as $Read(X)$ for reading logical data item X and $Write(X)$ for updating X, are issued from a transaction manager TM. The physical copies of X, i.e. $\{x_1, x_2, \cdots, x_m\}$ are stored in local databases and managed by database managers DMs. A TM always acts as a coordinator of a transaction; while a DM as a participant. The basic 2PL protocol can be represented formally as follows:

Basic 2PL Protocol:

Phase 1:

1.1 The coordinator TM_k sends a $Prepare\ (X)$ and lock request (PRE) message to all the participants $\{DM_i, X \in DM_i\}$.

1.2 Each DM_i receives the message; and checks for the requested locks. If all locks are granted and it is willing to commit, then it writes a transaction's records in the log and sends a ready (RDY) answer message to TM_k. If it wants the transaction to be aborted, it sends an abort (ABT) message to TM_k.

Phase 2:

2.1 Upon receiving the answer message RDY or ABT from $\{DM_i, X \in DM_i\}$, TM_k sends a commit (CMT) message to all $\{DM_i, X \in DM_i\}$ if all of them have voted RDY, otherwise it writes an abort record and sends an ABT to all $\{DM_i, X \in DM_i\}$.

2.2 After receiving the command message from TM_k, each DM_i writes either a abort or commit record in log; then executes the transaction and release the locks; and sends the acknowledge (ACK) message to TM_k.

2.3 TM_k waits for ACK messages from all $\{DM_i, X \in DM_i\}$; then writes a completion record in log.

Figure 4.1 shows the communication structure for the basic 2PL. Two phases are shown in the figure. During the first phase of the basic 2PL, locks are requested from all participants, which corresponds to the growing phase. A decision about whether to commit or abort the transaction is made at the locked point. Upon receiving commit, all participants will execute transactions and release locks in the shrinking phase. After the transaction is executed by all participants, the locks will be released, which is related to the shrinking phase. The communication of a complete transaction need $4 \cdot m$ messages, where m is the number of participants involved.

4.2 Two Phase Commit Model Definition

The ultimate goal of this section is to define the Markov chain matrix of the queueing network model of the distributed databases with two phase locking protocol. Here a systematic way to obtain the Markov chain matrix is introduced. In section 4.2.1 the access pattern matrix, defined as the probabilities of node access, is derived. Section 4.2.2 introduces a method to define the communication flow rates between database nodes. Section 4.2.3 obtains the arrival rates of the queueing network from the communication flow rate matrix. In section 4.2.4 the final Markov chain matrix is derived from the communication flow rate matrix.

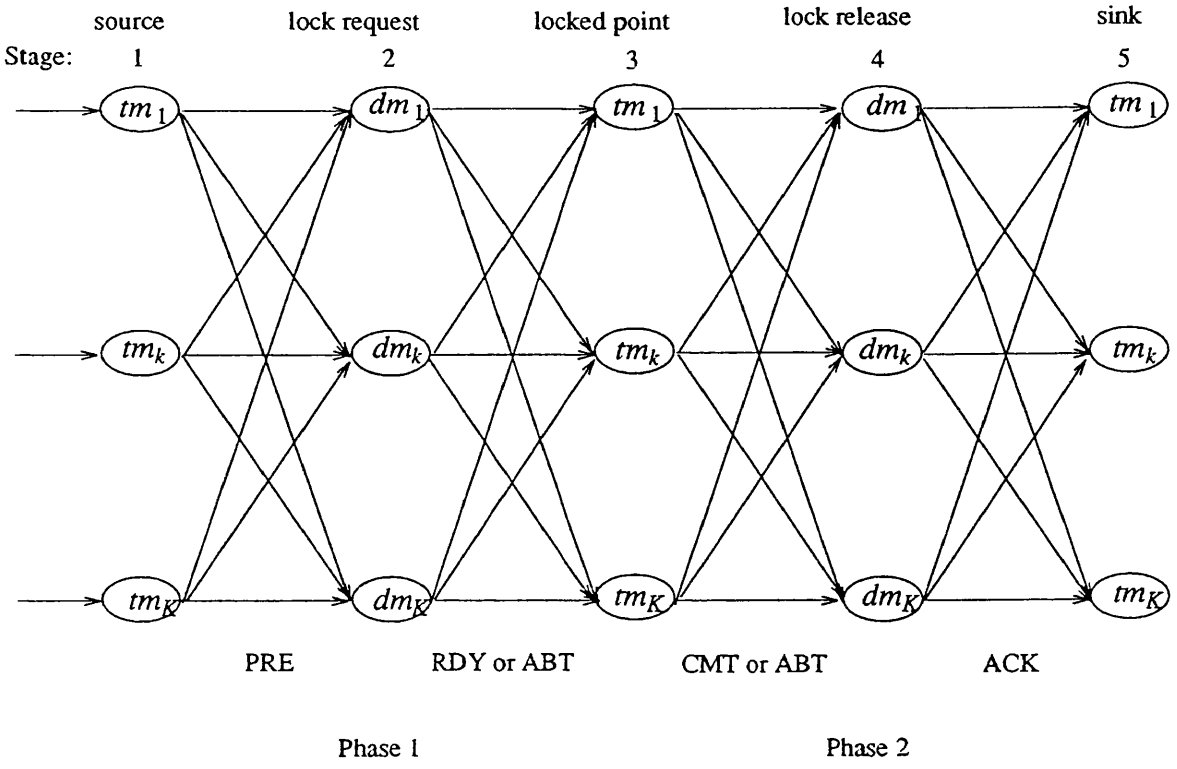


Figure 4.1. Communication Structure of 2PL

4.2.1 Access Pattern Matrix

In order to formally define the probability of node access, we introduce the concept of access pattern matrix. It is denoted by

$$\mathbf{W} = \begin{bmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1K} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \omega_{K1} & \omega_{K2} & \cdots & \omega_{KK} \end{bmatrix}$$

where the element ω_{ij} is the probability of a transaction accessing node DM_j from node TM_i and K is the number of nodes in the system. The access pattern matrix is determined by the protocol, the read-write ratio and the mean number of replicated copies. We shall first give a formal definition of the read and write rules of the basic 2PL. The basic 2PL requires that write operations have to access all the copies of a logical data item X while read operations need only to access one.

Definition 4.1: A $Read(X)$ transaction generated at TM_k is governed by the following rules:

- If $X \in DM_k$, access DM_k;
- If $\overline{X \in DM_k}$, access one of the $\{DM_j, X \in DM_j\}$ with probability $1/N_f$, where N_f denotes the mean number of replicated data copies in the system.

where node k is a local node and node j $(j \neq k)$ is a remote node.

Definition 4.2: A $Write(X)$ transaction generated at TM_k is governed by the following rule:

- Access all of the $\{DM_j, X \in DM_j\}$.

Let us denote N_{acc_k} as the mean number of database accesses generated by one transaction issued at TM_k. According to the read and write rules, $Read(X)$ accesses one data copy while $Write(X)$ accesses an average of N_f data copies. Thus we have

$$\begin{aligned} N_{acc_k} &= Prob\{Read\} \cdot 1 + Prob\{Write\} \cdot N_f \\ &= (1-\gamma_{rw_k}) + \gamma_{rw_k} N_f \end{aligned} \tag{4.1}$$

in which γ_{rw_k} is the read-write ratio given by

$$\gamma_{rw_k} = \frac{number\ of\ write\ transactions\ generated\ at\ TM_k}{total\ transactions\ generated\ at\ TM_k} \tag{4.2}$$

Suppose the total transactions generated by users at node k in one time unit are λ_k'. Thus the total number of database accesses generated at TM_k is given by

$$\begin{aligned} \lambda_k &= N_{acc_k} \lambda_k' \\ &= \left[1 + (N_f - 1)\gamma_{rw_k}\right] \lambda_k' \quad k = 1, \cdots, K \end{aligned} \tag{4.3}$$

The access probabilities dependent on the read-write ratio γ_{rw_k} and the mean number of replicated copies N_f. The probability of accessing a local data copy, i.e. from TM_k to DM_k, is given by

$$\begin{aligned} \omega_{kk} &= Prob\{local\ access\} \quad k = 1, \ldots, K \\ &= Prob\{Access_{kk}\} \quad k = 1, \ldots, K \end{aligned} \tag{4.4a}$$

and the probability of remote access, i.e. from TM_k to DM_k, is

$$\begin{aligned} \omega_{kj} &= Prob\{remote\ access\} \\ &= Prob\{Access_{kj}\} \quad k, j = 1, \ldots, K \text{ and } j \neq k \end{aligned} \tag{4.4b}$$

The distributed function of the locations of data copies is given by $Prob\{X \in DM_k\}$. Suppose the replicated data copies are distributed in database nodes with a uniform distribution. The probability of a data copy stored at DM_k is given by

$$Prob\{X \in DM_k\} = \frac{N_f}{K} \quad k = 1, \ldots, K \tag{4.5}$$

The access probability can then be derived by the following theorems.

Theorem 4.3: The probability of accessing a local data copy from TM_k is given by the access pattern matrix **W** with

$$\omega_{kk} = \frac{\frac{N_f}{K}}{1-\gamma_{rw_k}+N_f\gamma_{rw_k}} \quad k=1,...,K \tag{4.6}$$

Proof: We shall first give the conditional probability of ω_{kk}.

$$\begin{aligned}\omega_{kk} = &Prob\{Access_{kk} \mid X \in DM_k\} Prob\{X \in DM_k\} \\ &+ Prob\{Access_{kk} \mid \overline{X \in DM_k}\} Prob\{\overline{X \in DM_k}\}\end{aligned} \tag{4.7}$$

where $Prob\{Access_{kj} \mid X \in DM_k\}$ denotes the probability of access data copy at DM_j from TM_k under the condition that there is a data copy stored at DM_k. According to the read and write rules we have

$$\begin{aligned}Prob\{Access_{kk} \mid X \in DM_k\} &= \frac{Prob\{Read\}\cdot 1 + Prob\{Write\}\cdot 1}{N_{acc_k}} \\ &= \frac{1}{1-\gamma_{rw_k}+N_f\gamma_{rw_k}}\end{aligned} \tag{4.8}$$

and

$$Prob\{Access_{kk} \mid \overline{X \in DM_k}\} = 0 \tag{4.9}$$

Hence

$$\begin{aligned}\omega_{kk} &= \frac{1\cdot\frac{N_f}{K} + 0\cdot(1-\frac{N_f}{K})}{1-\gamma_{rw_k}+N_f\gamma_{rw_k}} \\ &= \frac{\frac{N_f}{K}}{1-\gamma_{rw_k}+N_f\gamma_{rw_k}}\end{aligned} \tag{4.10}$$

Theorem 4.4: The probability of accessing a remote data copy from TM_k is given by

$$\omega_{kj} = \frac{\gamma_{rw_k}\frac{N_f}{K}+(1-\gamma_{rw_k})(1-\frac{N_f}{K})\frac{1}{K-1}}{1-\gamma_{rw_k}+N_f\gamma_{rw_k}} \quad k,j=1,...,K \text{ and } j\neq k \tag{4.11}$$

Proof: The probability of accessing a remote data copy from TM_k is determined by the conditional probability of whether the copy is stored locally or remotely. That is

$$\begin{aligned}\omega_{kj} = & Prob\{Access_{kj} \mid X \in DM_k \cap X \in DM_j\} Prob\{X \in DM_k \cap X \in DM_j\} \\ & + Prob\{Access_{kj} \mid \overline{X \in DM_k} \cap X \in DM_j\} Prob\{\overline{X \in DM_k} \cap X \in DM_j\} \\ & + Prob\{Access_{kj} \mid X \in DM_k \cap \overline{X \in DM_j}\} Prob\{X \in DM_k \cap \overline{X \in DM_j}\} \\ & + Prob\{Access_{kj} \mid \overline{X \in DM_k} \cap \overline{X \in DM_j}\} Prob\{\overline{X \in DM_k} \cap \overline{X \in DM_j}\} \end{aligned} \quad (4.12)$$

The conditional probabilities about the locations of the data item X are given by

$$\begin{aligned} Prob\{X \in DM_k \cap X \in DM_j\} &= Prob\{X \in DM_k\} Prob\{X \in DM_j \mid X \in DM_k\} \\ &= \frac{N_f}{K} \cdot \frac{N_f - 1}{K - 1} \end{aligned} \quad (4.13a)$$

$$\begin{aligned} Prob\{\overline{X \in DM_k} \cap X \in DM_j\} &= Prob\{\overline{X \in DM_k}\} Prob\{X \in DM_j \mid \overline{X \in DM_k}\} \\ &= \left(1 - \frac{N_f}{K}\right) \cdot \frac{N_f}{K - 1} \end{aligned} \quad (4.13b)$$

$$\begin{aligned} Prob\{X \in DM_k \cap \overline{X \in DM_j}\} &= Prob\{X \in DM_k\} Prob\{\overline{X \in DM_j} \mid X \in DM_k\} \\ &= \frac{N_f}{K} \cdot \left(1 - \frac{N_f - 1}{K - 1}\right) \end{aligned} \quad (4.13c)$$

$$\begin{aligned} Prob\{\overline{X \in DM_k} \cap \overline{X \in DM_j}\} &= Prob\{\overline{X \in DM_k}\} Prob\{\overline{X \in DM_j} \mid \overline{X \in DM_k}\} \\ &= \left(1 - \frac{N_f}{K}\right) \cdot \left(1 - \frac{N_f}{K}\right) \end{aligned} \quad (4.13d)$$

and the conditional access probabilities are given by

$$Prob\{Access_{kj} \mid X \in DM_k \cap X \in DM_j\} = \frac{(1 - \gamma_{rw_k}) \cdot 0 + \gamma_{rw_k} \cdot 1}{1 - \gamma_{rw_k} + N_f \gamma_{rw_k}}; \quad (4.14a)$$

$$Prob\{Access_{kj} \mid \overline{X \in DM_k} \cap X \in DM_j\} = \frac{(1 - \gamma_{rw_k}) \frac{1}{N_f} + \gamma_{rw_k} \cdot 1}{1 - \gamma_{rw_k} + N_f \gamma_{rw_k}}; \quad (4.14b)$$

$$Prob\{Access_{kj} \mid X \in DM_k \cap \overline{X \in DM_j}\} = 0; \quad (4.14c)$$

$$Prob\{Access_{kj} \mid \overline{X \in DM_k} \cap \overline{X \in DM_j}\} = 0. \quad (4.14d)$$

Substituting the above equations into equation (4.12), we have

$$\omega_{kj} = \frac{\gamma_{rw_k}\frac{N_f}{K}\cdot\frac{N_f-1}{K-1} + \left[(1-\gamma_{rw_k})\frac{1}{N_f}+\gamma_{rw_k}\right]\left[1-\frac{N_f}{K}\right]\frac{N_f}{K-1}}{1-\gamma_{rw_k}+N_f\gamma_{rw_k}}$$
$$= \frac{\gamma_{rw_k}\frac{N_f}{K} + (1-\gamma_{rw_k})\left[1-\frac{N_f}{K}\right]\frac{1}{K-1}}{1-\gamma_{rw_k}+N_f\gamma_{rw_k}} \qquad (4.15)$$

Hence. []

4.2.2 Communication Flow Method

Here we introduce a novel method to represent the flow rate of transactions in the concurrency control model. We shall use the concept of stages to define the two phase commit protocol. As indicated in figure 4.1, stage 1 is defined as the source point of the system; stage 2 is the lock request point; stage 3 is the locked point; stage 4 is the lock release or transaction execution point and stage 5 is the sink point of the system. The communication flow matrix is to to define the flow rates of transaction from one stage to the next. Suppose the distributed database system has K nodes. The communication flow matrix from stage s to stage $s+1$ is denoted by $\mathbf{A}_s$ with element $a_{ij}^{(s)}$ representing the flow rates of messages from node i to j.

Between the first and the second stage of the two phase commit protocol, a PRE message generated at TM_i has to be sent to $\{DM_j,\ X \in DM_j\}$ with the flow rates given by

$$a_{ij}^{(1)} = \lambda_i \omega_{ij} \quad (i,j=1,...,K) \qquad (4.16)$$

where λ_i is the transaction arrival rate at TM_i. The communication flow matrix at stage 1 can be obtained by

$$\mathbf{A}_1 = \left[\lambda_1\ \lambda_2\ \cdots\ \lambda_K\right]\mathbf{W} = \begin{bmatrix} \lambda_1\omega_{11} & \lambda_1\omega_{12} & \cdots & \lambda_1\omega_{1K} \\ \lambda_2\omega_{21} & \lambda_2\omega_{22} & \cdots & \lambda_2\omega_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_K\omega_{K1} & \lambda_K\omega_{K2} & \cdots & \lambda_K\omega_{KK} \end{bmatrix} \qquad (4.17)$$

Between stage 2 and 3, all DMs received PRE message will send back either a CMT or ABT message back to TM_i. This is a reversion of the transmission procedure between stage 1 and 2. Its communication flow matrix $\mathbf{A}_2$ can be obtained simply by transposing the matrix $\mathbf{A}_1$. That is

$$\mathbf{A}_2 = \mathbf{A}_1^T = \begin{bmatrix} \lambda_1\omega_{11} & \lambda_2\omega_{21} & \cdots & \lambda_K\omega_{K1} \\ \lambda_1\omega_{12} & \lambda_2\omega_{22} & \cdots & \lambda_K\omega_{K2} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_1\omega_{1K} & \lambda_2\omega_{2K} & \cdots & \lambda_K\omega_{KK} \end{bmatrix} \qquad (4.19)$$

Stage 3–4 and 4–5 are respectively the same as stage 1–2 and 2–3. Their communication flow matrices are given by

$$\mathbf{A}_3 = \mathbf{A}_1; \tag{4.20}$$

and

$$\mathbf{A}_4 = \mathbf{A}_2 \tag{4.21}$$

4.2.3 Arrival Rate Matrix

The communication flow matrix introduced above can be used to calculate the arrival rate at various service centres very efficiently. The arrival rate at stage s is defined by a vector with K members

$$\Lambda_s = \left[\lambda_1^{(s)} \;\; \lambda_2^{(s)} \;\; \cdots \;\; \lambda_K^{(s)}\right] \tag{4.22}$$

where $\lambda_k^{(s)}$ is the arrival rate of the node k at stage s.

At the source stage, i.e. $s=1$, the arrival rate at node k is given by λ_k. That is

$$\lambda_k^{(1)} = \lambda_k \qquad k=1,...,K \tag{4.23}$$

$$\Lambda_1 = \left[\lambda_1^{(1)} \;\; \lambda_2^{(1)} \;\; \cdots \;\; \lambda_K^{(1)}\right] = \left[\lambda_1 \;\; \lambda_2 \;\; \cdots \;\; \lambda_K\right] \tag{4.24}$$

At the lock request stage, i.e. $s=2$, the arrival rates of lock request service centres can be obtained by

$$\Lambda_2 = \mathbf{V}^T \mathbf{A}_1 \tag{4.25}$$

where

$$\mathbf{V} = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ \cdots \\ 1 \end{bmatrix} \tag{4.26}$$

is a vector of K elements with all elements equal to 1. Thus

$$\Lambda_1 = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \lambda_1\omega_{11} & \lambda_1\omega_{12} & \cdots & \lambda_1\omega_{1K} \\ \lambda_2\omega_{21} & \lambda_2\omega_{22} & \cdots & \lambda_2\omega_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_K\omega_{K1} & \lambda_K\omega_{K2} & \cdots & \lambda_K\omega_{KK} \end{bmatrix} \tag{4.27}$$

$$= \begin{bmatrix} \lambda_1\omega_{11}+\lambda_2\omega_{21}+\cdots+\lambda_K\omega_{K1} & \lambda_1\omega_{12}+\lambda_2\omega_{22}+\cdots+\lambda_K\omega_{K2} & \cdots & \lambda_1\omega_{1K}+\lambda_2\omega_{2K}+\cdots+\lambda_K\omega_{KK} \end{bmatrix}$$

Similarly the arrival rate vector at locked stage, lock request stage and sink stage are respectively given by

$$\Lambda_3 = \mathbf{V}^T\mathbf{A}_2 \quad ; \tag{4.28}$$

$$\Lambda_4 = \mathbf{V}^T\mathbf{A}_3 \quad ; \tag{4.29}$$

and

$$\Lambda_5 = \mathbf{V}^T\mathbf{A}_4 \tag{4.30}$$

The general formula to obtain the arrival rate at various service stages is given by

$$\Lambda_s = \mathbf{V}^T\mathbf{A}_{s-1} \quad s=2,...,5 \tag{4.31}$$

It should be noted that the communication channel from node i to j has been used for four times during four stages of the two phase commit protocol. The overall arrival rates of the communication channels should be equal to the sum of the arrival rates at these four stages. If we use Λ_c to denote the communication flow rate matrix, Λ_c is equal to the sum of the flow rates matrices at four stage. That is

$$\Lambda_c = \sum_{s=1}^{4} \mathbf{A}_s \tag{4.32}$$

It is a $K \times K$ dimensional matrix with element $\lambda_{ij}^{(c)}$ being defined as the arrival rate of communication channel from node i to j.

4.2.4 Markov Chain Matrix

The Markov chain matrix is an essential factor in defining queueing network models. When a system is as complicated as a distributed database, to obtain the Markov chain matrix of the system is not a straightforward task. This is especially true when the system with several stages has to share communication channels. A systematic way to solve this problem is to use our communication flow method to obtain the Markov chain matrix.

Let us first introduce the concept of the local Markov chain matrices. Let $\mathbf{P}_I^{(k)}$ denote the matrix of the input flow to the communication channel at node k and $\mathbf{P}_O^{(k)}$ the output flow from the communication channel at node k. In $\mathbf{P}_I^{(k)}$ matrix, the element p_{si} $(s=1,...,5)$ $(i=1,...,K)$ denotes the probability of a job entering the ith communication channel in node k from stage s. We have

$$\mathbf{P}_I^{(k)} = \begin{bmatrix} \dfrac{\mathbf{I}_1\mathbf{A}_1}{|\mathbf{I}_1\mathbf{A}_1|} \\ \dfrac{\mathbf{I}_2\mathbf{A}_2}{|\mathbf{I}_2\mathbf{A}_2|} \\ \dfrac{\mathbf{I}_3\mathbf{A}_3}{|\mathbf{I}_3\mathbf{A}_3|} \\ \dfrac{\mathbf{I}_4\mathbf{A}_4}{|\mathbf{I}_4\mathbf{A}_4|} \\ \dfrac{\mathbf{I}_5\mathbf{A}_5}{|\mathbf{I}_5\mathbf{A}_5|} \end{bmatrix} \tag{4.33}$$

where $\mathbf{I}_k$ is a vector with kth element equal to one and the rest equal to zero, i.e.

$$\mathbf{I}_i = \begin{bmatrix} 0 \cdots 0\,1\,0 \cdots 0 \end{bmatrix} \tag{4.34}$$

Since there is no traffic flowing from stage 5, i.e. the sink, to the communication channels, we have

$$\mathbf{A}_5 = \mathbf{0} \tag{4.35}$$

The overall input matrix is equal to

$$\mathbf{P}_I = \begin{bmatrix} \mathbf{P}_I^{(1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_I^{(2)} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P}_I^{(K)} \end{bmatrix} \tag{4.36}$$

Let us denote $\mathbf{P}_O$ as the output Markov chain matrix defining the traffic from communication channels to the service centre at various stages. The element p_{is} of $\mathbf{P}_O$ is defined as the probability of traffic from the ith communication channel to the service centre at stage s. And we further define matrix $\mathbf{P}_O^{(k)}$ in which its element $p_{is}^{(k)}$ is the probability of traffic from from the ith communication channel at node k to the sth stage. We have

$$\mathbf{P}_O^{(k)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \dfrac{\mathbf{I}_0\mathbf{A}_0\mathbf{I}_0^T}{\mathbf{I}_0\Lambda_c\mathbf{I}_0^T} & \dfrac{\mathbf{I}_1\mathbf{A}_1\mathbf{I}_1^T}{\mathbf{I}_1\Lambda_c\mathbf{I}_1^T} & \dfrac{\mathbf{I}_2\mathbf{A}_2\mathbf{I}_2^T}{\mathbf{I}_2\Lambda_c\mathbf{I}_2^T} & \dfrac{\mathbf{I}_3\mathbf{A}_3\mathbf{I}_3^T}{\mathbf{I}_3\Lambda_c\mathbf{I}_3^T} & \dfrac{\mathbf{I}_4\mathbf{A}_4\mathbf{I}_4^T}{\mathbf{I}_4\Lambda_c\mathbf{I}_4^T} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.37}$$

Since there is no traffic flowing from the communication channels to the source stage, we have

$$\mathbf{A}_0 = \mathbf{0} \tag{4.38}$$

The overall output Markov chain matrix $\mathbf{P}_O$ is given by

$$\mathbf{P}_O = \begin{bmatrix} \mathbf{P}_O^{(1)} & \mathbf{P}_O^{(2)} & \cdots & \mathbf{P}_O^{(K)} \\ \mathbf{P}_O^{(1)} & \mathbf{P}_O^{(2)} & \cdots & \mathbf{P}_O^{(K)} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{P}_O^{(1)} & \mathbf{P}_O^{(2)} & \cdots & \mathbf{P}_O^{(K)} \end{bmatrix} \tag{4.39}$$

Combine the input and the output Markov chain matrices together, we can obtain the overall Markov chain matrix of the queueing network model.

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_I & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_O \end{bmatrix} \tag{4.40}$$

In this section, we have introduced an systematic method to define a distributed database in the form of communication flow matrix, access pattern matrix, arrival rate matrix and Markov chain matrix, which has not been addressed before. Previous researches can only pass the problem of defining a complicated distributed database to individual users, who have to manually define a model. This tedious task can be avoided by using the systematic model definition method introcuded in this section.

4.2.5 Queueing Network Model

Using the result derived in the previous sections, a distributed database can be easily defined as a network of queues with the defined Markov chain matrix, as shown in figure 4.2. It illustrates the kth node of a distributed database consisting of $K+2$ service centres. Let us denote service centres $\{k_i, i=1,...,K\}$ as the communication channels with the arrival rates defined by equation (4.32) and service centres k_{K+1}, and k_{K+2} as the lock request centre at stage 2 and the transaction execution centre at stage 4 respectively. The probability of a transaction going from one service centre to another is uniquely defined by the Markov chain matrix in equation (4.40).

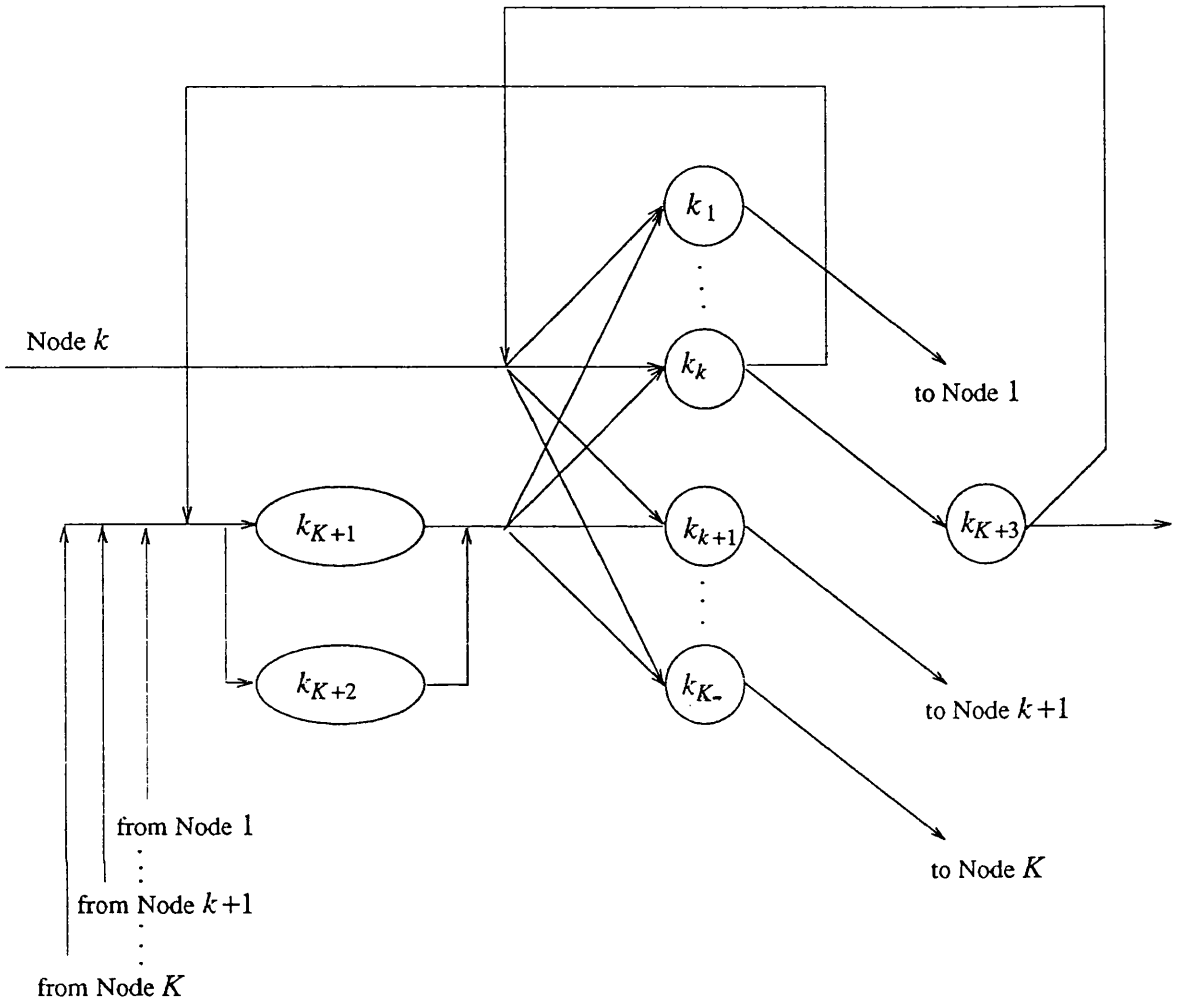


Figure 4.2. 2PL Distributed Database Model at Node k

4.3 Fixed Waiting Model

4.3.1 Locking Model with Fixed Waiting

Let us first discuss the fixed waiting scheduling algorithm. At the lock request stage, i.e. stage 2 in figure 4.1, a transaction first requests for locks at a DM with a preemptive priority. The database manager then checks the lock table for each required lock; which means that if m locks are required, the database manager will check the locks table for m times. If all the locks are granted, the transaction will leave the DM; otherwise the transaction is said to be blocked and must enter a blocked queue. The transaction will wait in the blocked queue for a fixed amount of time and then request those locks again until it grants all of them. All the locks will be released at the lock release stage, i.e. stage 4, after the transaction is executed. The fixed waiting scheduling algorithm can be best modeled by a network of queues as shown in figure 4.2.

We shall first study the locking model of one DM in isolation. Suppose each transaction issued at TM_k requires r_k locks. Since the overhead of lock operation is usually much smaller than that of the transaction execution operation and a lock operation always has a preemptive priority over a transaction execution operation, a lock request is almost always being served immediated. This is exactly the same as using an infinite-server service centre to process lock operations.

We can define a locking model with r_k stages, each of which represents the procedure of requesting one lock, as shown in figure 4.3. In stage 0, a transaction is being served to grant its first lock. Once the lock is granted, the transaction enters the first stage of locking; and so on. After it grants all the required locks, it leaves the lock request centre. When there is a conflict upon requesting locks, a transaction has to wait in the blocked queue. There is a conflict-avoidance delay $1/\mu_b$ for each blocked transaction. The main objective here is to obtain the mean number of locks being held at the lock request centre at node k, denoted by $NL_k^{(lock)}$.

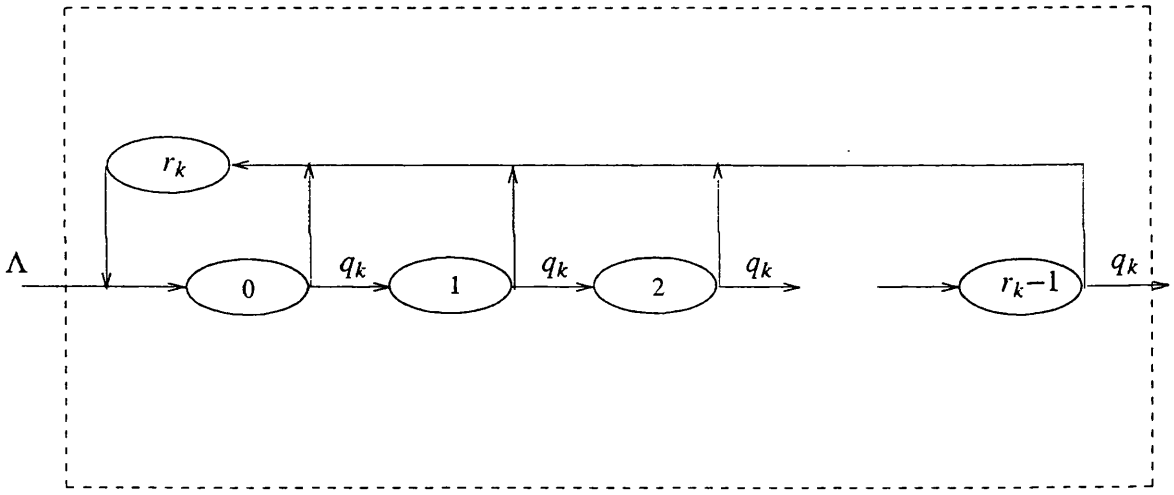


Figure 4.3. The Database Locking Model at DM_k

The interarrival rates at the stages are given by

$$\lambda_0 = \Lambda + \lambda_{r_k} \tag{4.41a}$$

$$\lambda_i = q_k \lambda_{i-1} = q_k^i \lambda_0 \qquad (i=1,...,r_k-1) \tag{4.41b}$$

where Λ is the overall arrival rate of the locking request centre at node k.

$$\lambda_{r_k} = \sum_{i=0}^{r_k-1} (1-q_k)\lambda_i$$
$$= \sum_{i=0}^{r_k-1} (1-q_k)q_k^i\lambda_0$$
$$= (1-q_k^{r_k})\lambda_0 \qquad (4.41c)$$

The above equations can be rewritten as

$$\lambda_0 = \frac{\Lambda}{1-\sum_{j=0}^{r_k-1}(1-q_k)q_k^j}$$
$$= q^{-r_k}\Lambda \qquad (4.42a)$$

$$\lambda_i = q_k^{i-r_k}\Lambda \qquad (4.42b)$$

and

$$\lambda_{r_k} = \frac{\Lambda\sum_{j=0}^{r_k-1}(1-q_k)q_k^j}{1-\sum_{j=0}^{r_k-1}(1-q_k)q_k^j} = (q_k^{-r_k}-1)\Lambda \qquad (4.42c)$$

Applying open queueing network theory[32] we can obtain a product form solution,

$$P(n_0,n_1,\cdots,n_{r_k}) = \frac{P_0(n_0)P_1(n_1)\cdots P_{r_k}(n_{r_k})}{G} \qquad (4.43)$$

where $P(n_0,n_1,\cdots,n_{r_k})$ is the steady-state probability of a network state with r_k+1 service centres, $P_j(n_j)$ $(j=0,\ldots,r_k)$ is a factor corresponding to the steady-state probability of the state of service centre j in isolation and G is the normalizing factor given by

$$G^{-1} = Prob\{n_0=0\}Prob\{n_1=0\}\cdots Prob\{n_{r_k}=0\} \qquad (4.44)$$

The steady-state probability of the state of service centre j in isolation is

$$P_j(n) = \frac{\lambda_j^n}{\prod_{i=1}^{n}\mu_j(i)}Prob\{n=0\} \qquad (4.45)$$

where $\mu_j(i)$ is the service rate at service centre j where i customers are being served. For infinite-server service centres

$$\mu_j(i) = i\mu_j \tag{4.46}$$

Thus $P_j(n)$ becomes

$$\begin{aligned} P_j(n) &= \frac{\lambda_j^n}{\prod_{i=1}^{n} i\mu_j} Prob\{n=0\} \\ &= \frac{1}{n!}\left(\frac{\lambda_j}{\mu_j}\right)^n Prob\{n=0\} \end{aligned} \tag{4.47}$$

Substituting equations (4.44) and (4.45) into equation (4.43), we have

$$\begin{aligned} P(n) &= \frac{1}{n_0!}\left(\frac{\lambda_0}{\mu_0}\right)^{n_0} \frac{1}{n_1!}\left(\frac{\lambda_1}{\mu_1}\right)^{n_1} \cdots \frac{1}{n_{r_k}!}\left(\frac{\lambda_{r_k}}{\mu_{r_k}}\right)^{n_{r_k}} \\ &= \frac{(n_0+n_1+\cdots+n_{r_k})!}{(n_0+n_1+\cdots+n_{r_k})!} \frac{\left(\frac{\lambda_0}{\mu_0}\right)^{n_0}\left(\frac{\lambda_1}{\mu_1}\right)^{n_1} \cdots \left(\frac{\lambda_{r_k}}{\mu_{r_k}}\right)^{n_{r_k}}}{n_0!n_1!\cdots n_{r_k}!} \\ &= \frac{\left(\frac{\lambda_0}{\mu_0} + \frac{\lambda_1}{\mu_1} + \cdots \frac{\lambda_{r_k}}{\mu_{r_k}}\right)^{(n_0+n_1+\cdots+n_{r_k})}}{(n_0+n_1+\cdots+n_{r_k})!} \end{aligned} \tag{4.48}$$

Let $n = n_0 + n_1 + \cdots + n_{r_k}$, we have

$$\begin{aligned} P(n) &= \\ &= \frac{\left(\sum_{i=0}^{r_k} \frac{\lambda_i}{\mu_i}\right)^n}{n!} \end{aligned} \tag{4.49}$$

In the locking model, $1/\mu_i$ $(i=0,...,r_k-1)$ is equal to the mean service time spend in lock scheduling at jth stage of locking. Since the lock request service times at all stages are equal,

$$\mu_i = \mu_s \quad (i=0, \cdots, r_k-1) \tag{4.50}$$

where μ_s is the service rate for requesting one lock. Substituting equations (4.42a-c) and (4.50) into equation (4.48), we have

$$P(n) = \frac{\left[\sum_{i=0}^{r_k-1} \frac{q_k^{i-r_k}}{\mu_s}\Lambda + \frac{q_k^{-r_k}-1}{\mu_b}\Lambda\right]^n}{n!}$$

$$= \frac{\left[\frac{\Lambda}{\mu_{ag}}\right]^n}{n!} \tag{4.51}$$

We can easily conclude, from the above result, that the locking model with r_k stages can be aggregated into one infinite-server service centre with its mean service time $1/\mu_{ag}$ equal to

$$\frac{1}{\mu_{ag}} = \frac{q_k^{-r_k}-1}{(1-q_k^{r_k})\mu_s} + \frac{q_k^{-r_k}-1}{\mu_b} \tag{4.52}$$

where $1/\mu_b$ is the service time of the blocked queue.

As given by equation (3.40), the waiting in the blocked queue with fixed waiting approximately equals to the time of two successive completions of transaction executions. That is whenever a transaction completes, the blocked transaction restarts. Since the throughput of transaction execution centre is λ_{K+2}, we have

$$\frac{1}{\mu_b} = \frac{1}{\lambda_{K+2}}$$

Now let us calculate the mean number of locks held at the aggregated locking request service centre at model k, which is defined as

$$NL_k^{(lock)} = \sum_{i=1}^{r_k} (\textit{The number of locks held at stage } i \textit{ by each customer}) \cdot$$

$$(\textit{The number of customers at stage } i) \tag{4.53}$$

According to the definition of the locking model, a transaction having granted i locks will enter stage i. Thus the number of locks held at stage i by each transaction is equal to i. And the mean number of transactions at stage i can be easily calculated by applying to the open queueing network theory for infinite-server service centres. That is

$$\bar{n}_i = \frac{\lambda_i}{\mu_s} \quad (i=0,....,r_k) \tag{4.54}$$

Thus we have

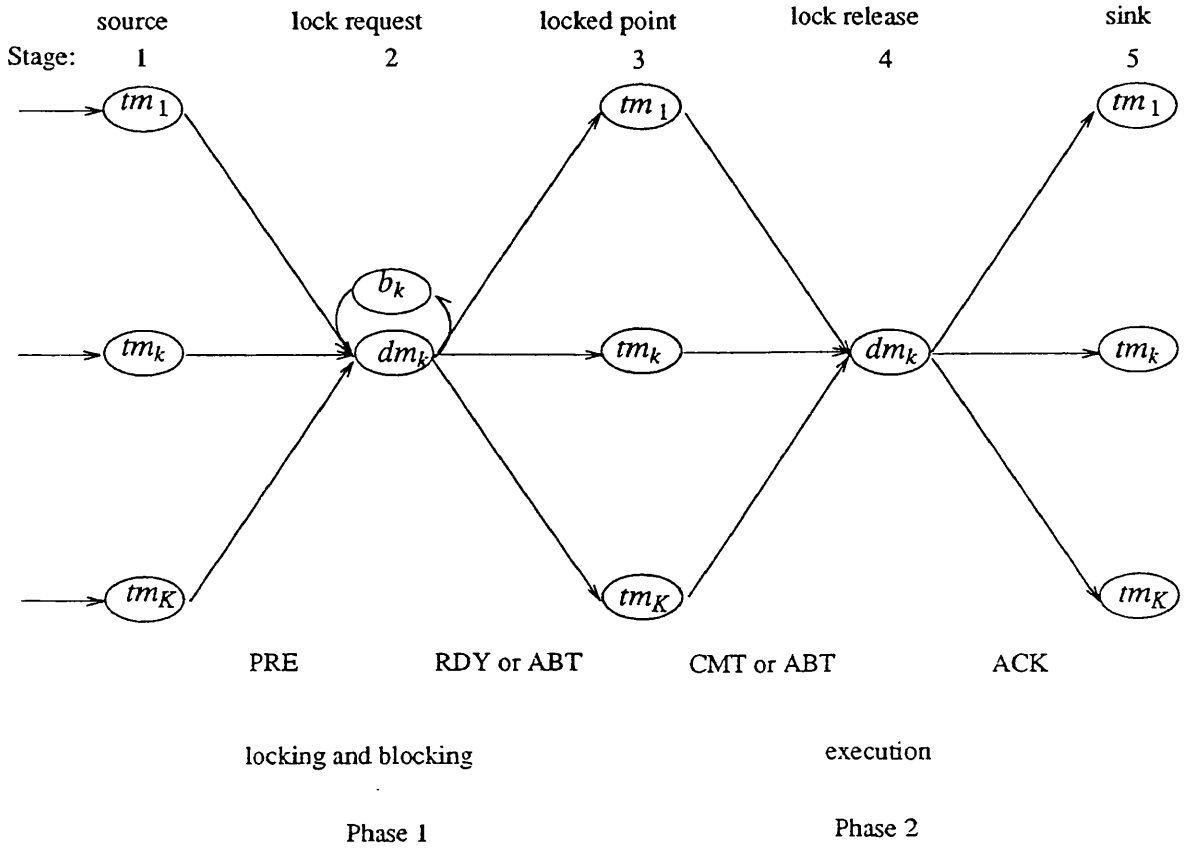


Figure 4.4. Distributed Database Locking Model

$$
\begin{aligned}
NL_k^{(lock)} &= \sum_{i=1}^{r_k} i\bar{n}_i \\
&= \sum_{i=1}^{r_k-1} \frac{i\lambda_i}{\mu_s} \\
&= \sum_{i=1}^{r_k-1} iq_k^{i-r_k}\frac{\Lambda}{\mu_s} \\
&= \sum_{i=1}^{r_k-1} iq_k^{i}\frac{\Lambda}{q_k^{r_k}\mu_s}
\end{aligned}
\tag{4.55}
$$

The method of evaluating locking by using infinite-server queueing network has not been used before. Tay, Suri and Goodman, who introduced a flow diagram method to evaluate locking in centralized database, admitted that the method is controversial and they can offer no theoretical justification for it[78,77]. By applying our method to their locking model of centralized database, we can obtain exactly the same result under the assumption of exponential service time distribution. Moreover we can easily and consistently extend this method to distributed locking.

Now we shall consider the locking model in a distributed database. After locks are granted at each lock request centre, a transaction holding the locks still has to go through two communication stages, i.e. the participants vote RDY and the coordinator issues CMT and one execute stage before releasing the locks, as shown in figure 4.4. Therefore transactions issued at TM_k still holds locks in communication channels 2-3 and 3-4 which are denoted by $NL_k^{(com\,2)}$ and $NL_k^{(com\,3)}$ respectively; and the locks held in the execution centre at node k is denoted by $NL_k^{(ex)}$.

As stated in section 2, stages 1-2, 2-3, 3-4 and 4-5 in figure 4.4 correspond to the four communication stages, PRE, RDY, CMT, and ACK respectively. Since the actual communication service centre from node j to i is the composition of the above four, its mean queueing length is equal to the sum of the four. Let $\bar{n}_{k_i}$ denote the mean queueing length of the ith communication channel in node k and $\bar{n}_{k_i}^{(s)}$ denote $\bar{n}_{k_i}$ at stage s. In a queueing network model only $\bar{n}_{k_i}$ can be directly obtained. Since $\bar{n}_{k_i}^{(s)}$ is proportional to $\bar{n}_{k_i}$, it can be obtained by using the communication flow matrix introduced in section 4.2.3. We have

$$\bar{n}_{k_i}^{(s)} = \frac{a_{ki}^{(s)}}{\lambda_{ki}^{(c)}} \bar{n}_{k_i} \quad s=1,...,4 \text{ and } k,i=1,...,K \tag{4.56}$$

Each transaction issued at TM_k still holds r_k locks when transmitting RDY and CMT messages. Therefore the transactions at communication channels 2-3 and 3-4 hold a total number of $NL_k^{(com\,2)} + NL_k^{(com\,3)}$ locks.

$$\begin{aligned} NL_k^{(com\,2)} &= \sum_{i=1}^{K} r_k \bar{n}_{i_k}^{(2)} \\ &= \sum_{i=1}^{K} r_k \frac{a_{ik}^{(2)}}{\lambda_{ik}^{(c)}} \bar{n}_{i_k} \end{aligned} \tag{4.57}$$

$$\begin{aligned} NL_k^{(com\,3)} &= \sum_{i=1}^{K} r_k \bar{n}_{k_i}^{(3)} \\ &= \sum_{i=1}^{K} r_k \frac{a_{ki}^{(3)}}{\lambda_{ki}^{(c)}} \bar{n}_{k_i} \end{aligned} \tag{4.58}$$

The transaction execution centre at TM_k corresponds to the k_{K+2} service centre in figure 4.2. The mean number of locks held in it is given by

$$NL_k^{(ex)} = r_k \bar{n}_{k_{K+2}} \tag{4.59}$$

Referring to figure 4.4, the service time of the blocked queue can be defined as the sum of the remaining lock request delay, two way communication delay and the transaction execution delay,

$$\frac{1}{\mu_b} = \frac{r_k}{2\mu_s} + \frac{2}{\mu_t} + \frac{1}{\mu_e}$$

where $1/\mu_t$ is the service time at communication service centre and $1/\mu_e$ is the mean service time at

transaction execution service centre.

Another very important parameter to describe the locking model is the probability of conflicts at DM_k, defined as

$$1-q_k = \frac{mean\ number\ of\ lock\ requests\ at\ node\ k}{number\ of\ granules\ at\ node\ k} = \frac{NL_k^{(lock)} + NL_k^{(com\,2)} + NL_k^{(com\,3)} + NL_k^{(ex)}}{L_k}$$

where $NL_k^{(lock)}$, $NL_k^{(com\,2)}$, $NL_k^{(com\,3)}$ and $NL_k^{(ex)}$ are the number of locks being held at lock request, RDY, CMT and execution stages respectively. Substituting (4.55), (4.57), (4.58) and (4.59) into the above equation, we have

$$1-q_k = \frac{\frac{\lambda_{k_{K+1}}}{q_k^{r_k}\mu_s}\sum_{i=1}^{r_k-1} iq_k^i + r_k\left[\left[\sum_{i=1}^{r_k}\frac{a_{ik}^{(2)}}{\lambda_{ik}^{(c)}}\bar{n}_{i_k} + \frac{a_{ki}^{(3)}}{\lambda_{ki}^{(c)}}\bar{n}_{k_i}\right] + \bar{n}_{j_{K+2}}\right]}{L_k} \tag{4.60}$$

This result is very important in the derivation of the whole distributed database model. q_k is dependent on almost all the parameters of the system, because in distributed locking lock conflict rate is related to the distribution and granularity of locks, number of transactions holding locks and the operatins of locks. On the other hand, other parameters also depend on q_k. Once q_k is obtained, they can also be easily calculated. q_k can be calculated by solve the equations iteratively with numerical method.

4.3.2 Queueing Network Solution

The queueing network model of the distributed database system with the basic 2PL is shown in figure 4.2. Each node is represented as one chain with connections to other chains. Any two nodes communicating with each other through a data communication channel represented as a single server service centre. The locking request is presented as a single aggregated service centre. Transaction execution service centre is served by a cpu-i/o compound server as discussed in section 2.5. The whole distributed database constructs an open queueing network with transactions arriving at the source and departuring at the sink.

We shall first present the queueing network solution of a distributed database system with exponential service times. Applying Jackson's queueing network theory, the open system has a product form solution of

$$P(n_1, n_2, \cdots, n_m) = \frac{P_1(n_1)P_2(n_2)\cdots P_m(n_m)}{G} \tag{4.61}$$

where m is the total number of service centres in the system, $P(n_1, n_2, \cdots, n_m)$ is the steady state probability of a network state, $P_j(n_j)$, $j=1,\ldots,m$) is a factor of the steady-state probability of the state of

service centre j in isolation, and G is a normalizing constant.

The Markov chain matrix of the open queueing network of the distributed database is given by equation (4.40) and its arrival rate matrix is given by equation (4.32). We shall define

$$\rho_j = \frac{\lambda_j}{\mu_j} \tag{4.62}$$

as the throughput of service centre j, where $1/\mu_j$ is the mean service time of service centre j. Applying Jackson's theory, the system utility for single server service centre is

$$U_j = \rho_j \tag{4.63a}$$

and for infinite-service centre is

$$U_j = 0 \tag{4.63b}$$

The mean queueing length for single server service centre is

$$\bar{n}_j = \frac{\rho_j}{1-\rho_j} \tag{4.64a}$$

and for infinite-server service centre is

$$\bar{n}_j = \frac{\lambda_j}{\mu_j} \tag{4.64b}$$

Applying Little's law the mean response time of service centre j is

$$\bar{R}_j = \frac{\bar{n}_j}{\lambda_j} \tag{4.65}$$

4.4 Extended Diffusion Approximation Approach

4.4.1 Diffusion Approximation Solution

In most performance evaluation models, interarrival process is assumed to be Poisson and service time be exponentially distributed[81, 29, 73]. However in a real world service time distributions do not usually fall into this category. For instance the service time distributions of computer system tend to be hyperexponential[54, 55]. Diffusion approximation[26, 39, 37] method provides a solution to the nonexponential queueing network problem.

For general queueing networks with single server service centres, a diffusion approximation model was first introduced by Gelenbe and Kobayashi[26, 39]. The model can be used in a wide range of applications. The method can be applied to the queueing network with only single server service centres. Since the locking in a distributed database is modeled by an infinite-server service centre, the restriction on single

server service centre needs to be released. We shall first present the conventional diffusion approximation method and then further extend it to the queueing networks with infinite-server service centres.

We shall consider an open queueing network as in the previous section. Let e_i be the relative throughput at service centre i,

$$e_i = p_{0i} + \sum_{j=1}^{m} e_j p_{ji} \qquad (i=1,...,m) \tag{4.67}$$

where m is the total number of service centres in the open distributed database queueing network and p_{ij} is the element of the Markov chain matrix given in equation (4.40). The arrival rate of service centre i is proportional to the relative throughput.

$$\lambda_i = \lambda_0 e_i \tag{4.68}$$

The utilization of the service centre i is

$$\rho_i = \frac{\lambda_0 e_i}{\mu_i}, \quad \text{if } \lambda_0 e_i < \mu_i \qquad (i=1,...,m) \tag{4.69}$$

Under the condition of heavy traffic, the total number of arrivals to station i in the interval $[0,t]$ will be normally distributed with mean

$$\lambda_i t \tag{4.70a}$$

and variance

$$\sum_{j=0}^{m} [(C_j - 1)p_{ji} + 1]\lambda_j p_{ji} t \qquad (i=1,...,m) \tag{4.70b}$$

where C_j is the squared coefficient of the variation of the interarrival time at service centre j. In the equation we have used the fact that the sum of independent normal random variables is normally distributed with variance being the sum of individual variances.

We can now construct the diffusion approximation to the length of an individual queue. We have

$$-\frac{\partial f_i(x_i,t)}{\partial t} - \beta_i \frac{\partial f_i(x_i,t)}{\partial x_i} + \frac{1}{2}\alpha_i \frac{\partial^2 f_i(x_i,t)}{\partial x_i^2} + \lambda_i P_{0i}(t)\delta(x_i - 1) = 0 \tag{4.71}$$

$$\frac{d}{dt}P_{0i}(t) = -\lambda_i P_{0i}(t) + \lim_{x_i \to 0^-}[-\beta_i f_i(x_i,t) + \frac{1}{2}\alpha_i \frac{\partial f_i(x_i,t)}{\partial x_i}] \tag{4.72}$$

$$f_i(0,t) = 0 \ \ (\textit{absorbing boundary}) \qquad (i=1,...,m) \tag{4.73}$$

where $f_i(x_i,t)$ is the density function approximating the length of the ith queue, $P_{0i}(t)$ is the probability of the ith queue being empty, and $\delta(x_i - 1)$ is the Dirac density function concentrated at $x_i = 1$. The other

parameters of the above equations are given by,

$$\beta_i = \lambda_i - \mu_i \tag{4.74}$$

$$\rho_i = \lambda_i/\mu_i \tag{4.75}$$

$$\alpha_i = \rho_i\mu_i K_i^2 + \sum_{j=0}^{m}[(C_j-1)p_{ji}+1]\lambda_j p_{ji} \tag{4.76}$$

$$(C_i-1) - \frac{(1-\rho_i)}{\lambda_i + \lambda_i p_{ii}^2(1-\rho_i)} \sum_{j=0, j\neq i}^{m} (C_j-1)\lambda_j p_{ji}^2 = \frac{\rho_i(K_i^2-1)}{1-(1-\rho_i)p_{ii}^2} \qquad (i=1,...,m) \tag{4.77}$$

where K_i is the squared coefficient of variation of the service time distribution at service centre i. C_i $(i=1,...,m)$ can be easily solved by numerical methods.

The stationary solution of equations (4.71), (4.72) and (4.73) is

$$f_i(x_i) = \begin{cases} \rho_i(e^{-\gamma_i}-1)e^{\gamma_i x_i}, & x_i \geq 1 \\ \rho_i(1-e^{\gamma_i x_i}), & 0 \leq x_i \leq 1 \end{cases} \tag{4.78}$$

$$P_{0i} = 1-\rho_i \tag{4.79}$$

where

$$\gamma_i = -2\beta_i/\alpha_i \tag{4.80}$$

$$\rho_i = \lambda_i/\mu_i \tag{4.81}$$

The approximate average queue length is

$$\bar{n}_i = \rho_i\left[1-\frac{\alpha_i}{2\beta_i}\right] \tag{4.82}$$

If for some i, $p_{ii}\neq 0$, we have to modify the parameters of the i-th queue so that

(i) p_{ij} $(j=1,...,m+1)$ is replaced by

$$p_{ij} = \begin{cases} 0 & \text{if } j=1 \\ \dfrac{p_{ij}}{(1-p_{ii})} & \text{if } j\neq 1 \end{cases} \tag{4.83}$$

(ii) μ_i and K_i^2 are replaced by

$$\bar{\mu}_i = \mu_i(1-p_{ii}) \tag{4.84a}$$

$$\overline{K}_i^2 = (1-p_{ii})K_i^2 + p_{ii} \tag{4.84b}$$

respectively.

Applying Little's law, the average turn-around time of the system is given by

$$T_a = \sum_{i=1}^{n} \frac{\overline{n}_i}{\lambda_0} \tag{4.85}$$

4.4.2 Diffusion Approximation Solution of Distributed Locking

The above diffusion approximation queueing network method is suitable only for single server service centres. But there are some infinite-server service centres in our model. We shall now further extend the method to infinite-server service centres, i.e. $/G/G/\infty$ queue.

Now let us first study the $G/M/\infty$ queue, without lossing the generality of the problem. Consider that customers arrive at epochs $\tau_1, \tau_2, \ldots$ and assume that the interarrival time $\tau_{k+1}-\tau_k$ $(k=0,1,\ldots;\ \tau_0=0)$ is i.i.d. with common distribution function $G(t)=P\{\tau_{k+1}-\tau_k \le t\}$ $(k=0,1,\ldots)$ and mean interarrival time $\overline{t}$. Also assume that the service time is exponentially distributed with mean $\overline{x}$. Let χ_i be the number of customers in the system prior to the interarrival of the ith customer

$$P\{\chi_{k+1}=j\} = \sum_{i=0}^{\infty} P\{\chi_{k+1}=j \mid \chi_k=i\} P\{\chi_k=i\} \qquad (j=0,1,\ldots;\ k=1,2,\ldots) \tag{4.86}$$

The transition probabilities are

$$p_{ij} = P\{\chi_{k+1}=j \mid \chi_k=i\} \tag{4.87}$$

The unique stationary distribution is

$$\Pi_j = \lim_{k\to\infty} P\{\chi_k=j\} \qquad (j=0,1,\ldots) \tag{4.88}$$

After taking the limits on both sides of equation (4.86), its stationary distribution becomes

$$\Pi_j = \sum_{i=0}^{\infty} p_{ij}\Pi_i \qquad (j=0,1,\ldots) \tag{4.89}$$

together with the normalization equation

$$\sum_{j=0}^{\infty} \Pi_j = 1 \tag{4.90}$$

where

$$p_{ij} = 0 \qquad (i+1<j<0) \tag{4.91}$$

At the $(k+1)$th arrival, there are approximately $\overline{N}$ servers being continuously busy during interarrival time

$\tau_{k+1}-\tau_k$. Since the service time of each server is assumed to be exponentially distributed with mean $\bar{x}$, it yields that

$$P\{\chi_{k+1}=j \mid \chi_k=i, \tau_{k+1}-\tau_k=t\} = \frac{(\bar{N}t/\bar{x})^{i+1-j}}{(i+1-j)!} e^{-\bar{N}t/\bar{x}} \qquad (i+1-j \geq 0) \tag{4.92}$$

and since the interarrival time $\tau_{k+1}-\tau_k$ has distribution function $G(t)$, we have

$$p_{ij} = \int_0^\infty \frac{(\bar{N}t/\bar{x})^{i+1-j}}{(i+1-j)!} e^{-\bar{N}t/\bar{x}}\, dG(t) \qquad (i+1-j \geq 0) \tag{4.93}$$

Substituting (4.93) into (4.94), we obtain

$$\Pi_j = \sum_{i=j-1}^{\infty} \Pi_i \int_0^\infty \frac{(\bar{N}t/\bar{x})^{i+1-j}}{(i+1-j)!} e^{-\bar{N}t/\bar{x}} dG(t) \tag{4.94}$$

With a change of variable, this equation becomes

$$\Pi_j = \sum_{i=0}^{\infty} \Pi_{i+j-1} \int_0^\infty \frac{(\bar{N}t/\bar{x})^i}{i!} e^{-\bar{N}t/\bar{x}}\, dG(t) \tag{4.95}$$

If Π_j is replaced by

$$\Pi_j = A\omega^j \tag{4.96}$$

equation (4.95) becomes

$$A\omega^j = \sum_{i=0}^{\infty} A\omega^{i+j-1} \int_0^\infty \frac{(\bar{N}t/\bar{x})^i}{i!} e^{-\bar{N}t/\bar{x}}\, dG(t)$$

$$\omega = \int_0^\infty e^{-(1-\omega)\bar{N}t/\bar{x}} dG(t)$$

$$= G^*((1-\omega)\bar{N}/\bar{x}) \tag{4.97}$$

where $G^*(s)$ is the Laplace transformation of $G(t)$.

Let

$$\alpha = (1-\omega)\bar{N}/\bar{x} \tag{4.98}$$

Applying this equation to (4.97), we obtain

$$1 - \frac{\alpha\bar{x}}{\bar{N}} = G^*(\alpha) \tag{4.99}$$

If we expand $G^*(\alpha)$ in a power series, it becomes

$$1 - \frac{\alpha\bar{x}}{\bar{N}} = 1 - \bar{t}\alpha + \frac{\overline{t^2}\alpha^2}{2!} + o(\alpha^2) \tag{4.100}$$

For the system in equilibrium, $\lim_{t\to\infty} G(t)$ corresponds to $\lim_{\alpha\to 0} G^*(\alpha)$. Thus the higher order terms of α may be neglected. The above equation therefore becomes

$$\alpha \approx \frac{2(\bar{t}-\bar{x}/\bar{N})}{\overline{t^2}} \tag{4.101}$$

For the $G/G/\infty$ system in equilibrium[64], the average number of busy servers tends to be the traffic intensity ρ'.

$$\rho' = \bar{x}/\bar{t} \to \bar{N} \tag{4.102}$$

Thus we have

$$(\bar{x}/\bar{t})^2 \approx \bar{N}^2 \tag{4.103}$$

From this equation, $\overline{t^2}$ becomes

$$\overline{t^2} = \sigma_a^2 + \bar{t}^2$$

$$\approx \sigma_a^2 + (\bar{x}/\bar{N})^2 \tag{4.104}$$

Because of the exponential nature of the service centre,

$$\sigma_b^2 = \bar{x}^2 \tag{4.105}$$

Thus equation (4.104) can be written as

$$\overline{t^2} \approx \sigma_a^2 + \sigma_b^2/\bar{N}^2 \tag{4.106}$$

Applying (4.106) to (4.101),

$$\alpha \approx \frac{2\bar{t}(1-\bar{x}/\bar{N}\bar{t})}{\sigma_a^2 + \sigma_b^2/\bar{N}^2} \tag{4.107}$$

Compare this equation with the $G/G/1$ result under heavy traffic approximation given by

$$\alpha = \frac{2\bar{t}(1-\bar{x}/\bar{t})}{\sigma_a^2 + \sigma_b^2} \tag{4.108}$$

we can conclude that the $G/G/\infty$ can be replaced by the $G/G/1$ with service capacity equal to $\bar{x}/\bar{N}$.

Finally we shall obtain the mean number of busy servers $\bar{N}$. The mean waiting time of the $G/G/1$ service centre with service capacity $\bar{x}/\bar{N}$ under heavy traffic assumption is equal to[35]

$$W = \frac{1}{\alpha} = \frac{\sigma_b^2 + \overline{t}^2}{2(\overline{t} - \overline{x}/\overline{N})} \tag{4.109}$$

Applying Little's law, the mean queueing length, i.e. the mean number of busy servers here, is given by

$$\overline{N} = \frac{W}{\overline{t}}$$

$$= \frac{\sigma_a^2 + \overline{t}^2}{2\overline{t}(\overline{t} - \overline{x}/\overline{N})} \tag{4.110}$$

From this equation, the number of mean busy servers can be obtained as

$$\overline{N} = \frac{1}{2}(\sigma_a^2/\overline{t} + 2\overline{x}/\overline{t} + 1)$$

Since $\overline{t} = 1/\lambda$ and $\overline{x} = 1/\mu$,

$$\overline{N} = \frac{1}{2}(\lambda^2\sigma_a^2 + 2\lambda/\mu + 1) \tag{4.111}$$

The above result extends the diffusion approximation method to infinite-server service centers. Prior to this, diffusion approximation method can only be used to model single-server queuing networks. By applying to the above extended diffusion approximation method, we can model both single-server and infinite-server queueing netowrks with non-exponential serivice times. The method can be used to model not only distributed databases but also general queueing networks of computer systems.

With the above extended diffusion approximation method, we are now able to treat the lock request service centre in the same way as other single server service centres. The lock request service centre j_{K+1} in figure 4.4 is an infinite-server service centre which can be represented by a single server service centre with service capacity equal to $1/\overline{N}_{j_{K+1}}\mu_{j_{K+1}}$ $(j=1,...,K)$. The throughput of lock request service centre j_{K+1} is defined as

$$\rho_{j_{K+1}} = \frac{\lambda_{j_{K+1}}}{\overline{N}_{j_{K+1}}\mu_{j_{K+1}}} \tag{4.112}$$

where $\overline{N}_{j_{K+1}}$ is given by equation (4.111) and can be rewritten as

$$\overline{N}_{j_{K+1}} = \frac{1}{2}(\lambda_{j_{K+1}}^2\sigma_{j_{K+1}}^2 + \frac{2\lambda_{j_{K+1}}}{\mu_{j_{K+1}}} + 1)$$

$$= \frac{1}{2}(C_{j_{K+1}} + \frac{2\lambda_{j_{K+1}}}{\mu_{j_{K+1}}} + 1) \tag{4.113}$$

Here $C_{j_{K+1}}$ is the squared coefficient of the variation of the interarrival time at service centre j_{K+1}, which is equal to the sum of the independent variance of the interdeparture time of all the other service centres to centre j_{K+1}.

$$C_{j_{K+1}} = \frac{1}{\lambda_{j_{K+1}}} \sum_{i=0}^{m} [\,(C_i - 1)p_{ij_{K+1}} + 1\,]\lambda_i p_{ij_{K+1}} \tag{4.114}$$

The probability of the locking conflict q_k can now be solved by

$$1-q_k = \frac{\dfrac{\lambda_{k_{K+1}}}{q_k^{r_k}\mu_s}\sum_{i=1}^{r_k-1} i q_k^i + r_k\left[\left[\sum_{i=1}^{r_k}\dfrac{a_{ik}^{(2)}}{\lambda_{ik}^{(c)}}\bar{n}_{i_k} + \dfrac{a_{ki}^{(3)}}{\lambda_{ki}^{(c)}}\bar{n}_{k_i}\right] + \bar{n}_{j_{K+2}}\right]}{L_k} \qquad k=1,\ldots,K$$

The mean throughput of the lock request centre $\rho_{k_{K+1}}$, the squared coefficient C_i, and the mean queueing length $\bar{n}_{i_k}$ $(i=1,\ldots,m\,;k=1,\ldots,K)$ can be respectively obtained from equation (4.112), (4.77) and (4.82).

In this section, we have introduced an extended diffusion approximation method to model distributed database systems with non-exponential service times. Previous researches can only model distributed databases with exponential service time. By applying the extended diffusion approximation method, we can model non-Poisson interarrival process and non-exponential service time distribution. Moreover locking and transaction blocking are also modeled by using the phase method. Various characteristics of a distributed database, such as data replication, data locality, read-write ratio, lock granularity, etc. are factored in. Using our method, a distributed database can therefore be modeled in a systematic way, which is not available before. The model can achieve a high degree of accuracy because of the introduction of the extended diffusion approximation method and enclosure of all the important characteristics of a distributed database.

4.5 Validation by Simulation

4.5.1 Simulation Model

In order the verify the analytic model introduced in the previous sections, a simulation model of basic 2PL with fixed waiting collision resolution algorithm is built as shown in figure 4.5. The model simulates an open distributed database system with transactions entering and leaving the system at a steady rate. A transaction goes through several stages (i.e. PRE, lock, RDY, CMT, execution and ACK) in the system. The coordinator starts with dispatching a transaction into several sub-transactions (sending PRE), which go to the communication queues to reach a remote destination. The dispatched sub-transactions then enter the lock queue to request locks. Upon conflict, they wait a fixed period and restart again; otherwise they send RDY to the coordinator through the correspondent communication channel. When the coordinator collects all the RDY answers from the participants, it reaches a commit stage and a CMT message is dispatched and sent to the participants through the communication channels. After receiving the CMT message a

participant enters the transaction execution queue to send ACK to the coordinator. The transaction is completed after the coordinator collets all the ACKs.

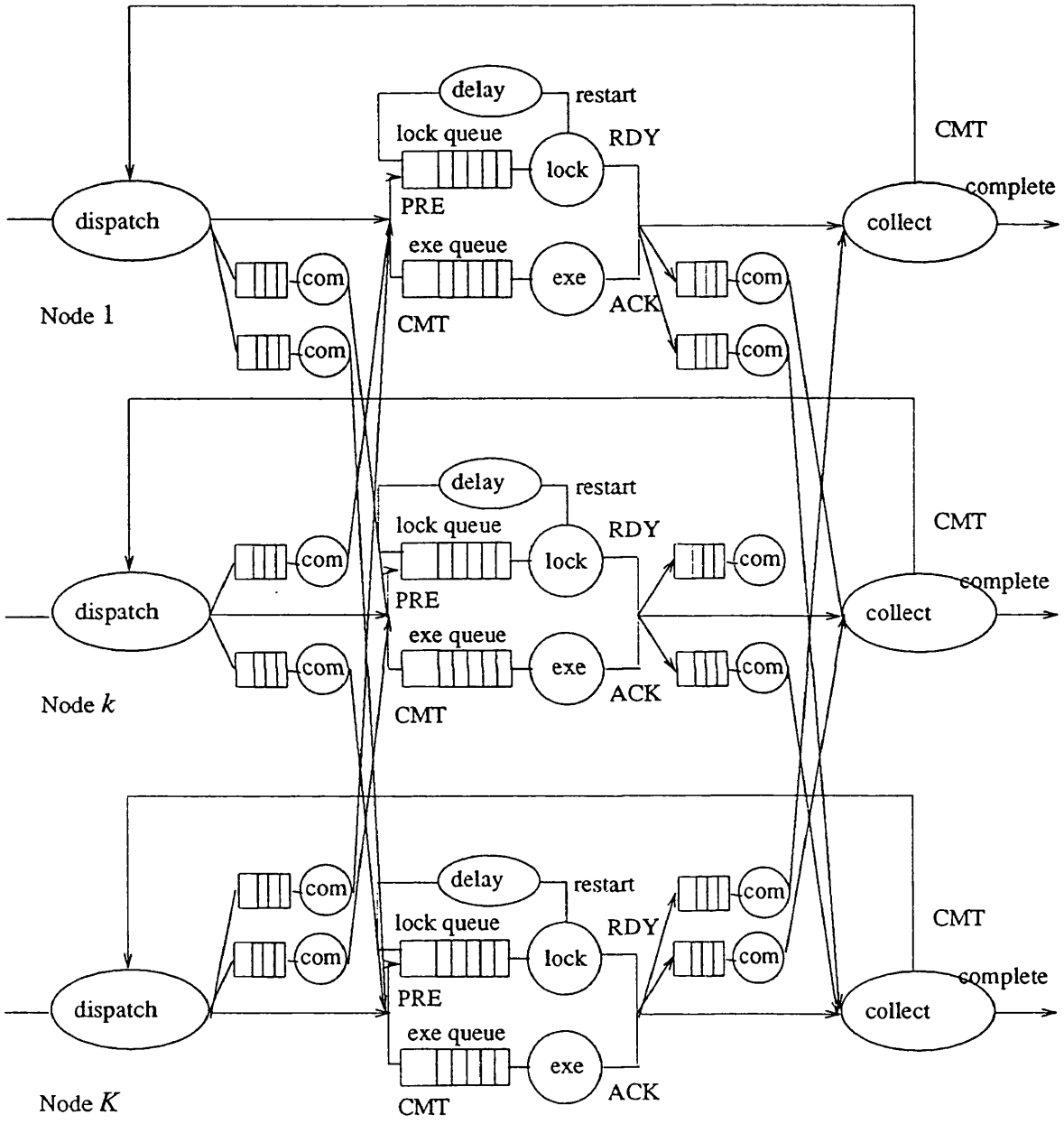


Figure 4.5. Simulation Model of a Distributed Database

Locking is performed, in this simulation model, by establishing a lock script at each site and locking and unlocking the correspondent item in the lock script. Here we suppose that data are uniformly distributed over the database. A uniform random generator is used to decide which item is to be locked.

The number of write transactions against that of read transactions is determined by the read-write ratio γ_{rw}. The read and write transactions are governed by the read and write rules defined in definition 4.1 and

4.2 respectively.

In the simulation model there are four types of queues, the communication queue, the transaction execution queue, the lock queue and the delay queue. The service time demands of the above queueing centres have general distributions. The actual service time distribution functions used in modeling the above queues can be determined by measurements from a real system. The measurement method to obtain the service time distribution functions can be found in appendix C. Since the main purpose of this section is to validate the analytic model by simulation, it is not important as to which distribution function is chosen. For simplicity, a constant service time for communication is used since a packet switching network tends to have a deterministic packet transmission time[37]. An exponentially distributed service demand with single server for transaction execution is used in this example and an exponential service time with infinite-server for locking and blocking delay are used.

4.5.2 Comparison of Results

As an application of the model, the test system consists of five database nodes. Without losing generality, the network of the test system is assumed to be fully connected. The parameters of the model are set as follows,

- mean service rates and squared coefficients of variance

Table 4.1. The service time parameters of figure 4.6 to 4.11

Service centre	Mean service rate $(1/s)$	Squared coefficient of variance K^2
Data transfer channel	18.182	0
Lock request scheduling	1160.09	1
Transaction execution	8.0	1

The interarrival service time is supposed to be exponentially distributed with

$$K_0^2=1 \tag{4.115}$$

- Other parameters

Table 4.2. The parameters of figure 4.6 to 4.11	
Specification	Value
Number of database nodes k	5
Interarrival rate $\lambda_0^{'}$ $(1/s)$	0-10
Update ratio γ_{rw}	0.25
Number of duplicated copies N_f	5
Number of locks requeried r	15
Number of granulars L	1500

In order to verify our analytic model and compare it with other models, three groups of results are obtained based on the same distributed database model. The first one is from simulation model; the second from our analytic model based on diffusion approximation; and the third from Jackson's queueing network model based on the assumption of exponential distribution.

Various performance results are obtained from each of the three models, i.e. simulation, diffusion, and Jackson model. In order to illustrate the effect of different service time distributions, the mean waiting times of service centres with different distributions are provided for comparison. As defined previously, the communication channel has a deterministic service time distribution and the transaction execution service centre has an exponential distribution. The mean waiting times of the two different types of service centres are illustrated in figure 4.6 and 4.7 respectively. Notice the interesting effect of the deterministic service time distribution of communication service centre in figure 4.6. Our diffusion approximation results are very close to the simulation results, while Jackson's are not. This is because our diffusion model factors in the deterministic distribution character of the communication channel, while Jackson's model uses exponential assumption. It shows that our model is much more accurate than Jackson's model. Figure 4.7 illustrates the mean waiting time of the transaction execution centre. Results from our model and Jackson's are quite close to the simulation results, because the service time distributions are assumed to be exponential in both cases.

Figure 4.8 and 4.9 provide the results of the lock request service centre. In figure 4.8, the mean waiting times of the aggregated lock request centre from the three models are very close since the service time is exponentially distributed. In figure 4.9 the mean conflict rates of lock request obtained from the analytic models are also very close to the simulation model. In both cases we can see a very accurate agreement between analytic results and simulation results for the lock request model, which verifies that our derivation of the extented diffusion approximation method with infinite-server service centre is correct.

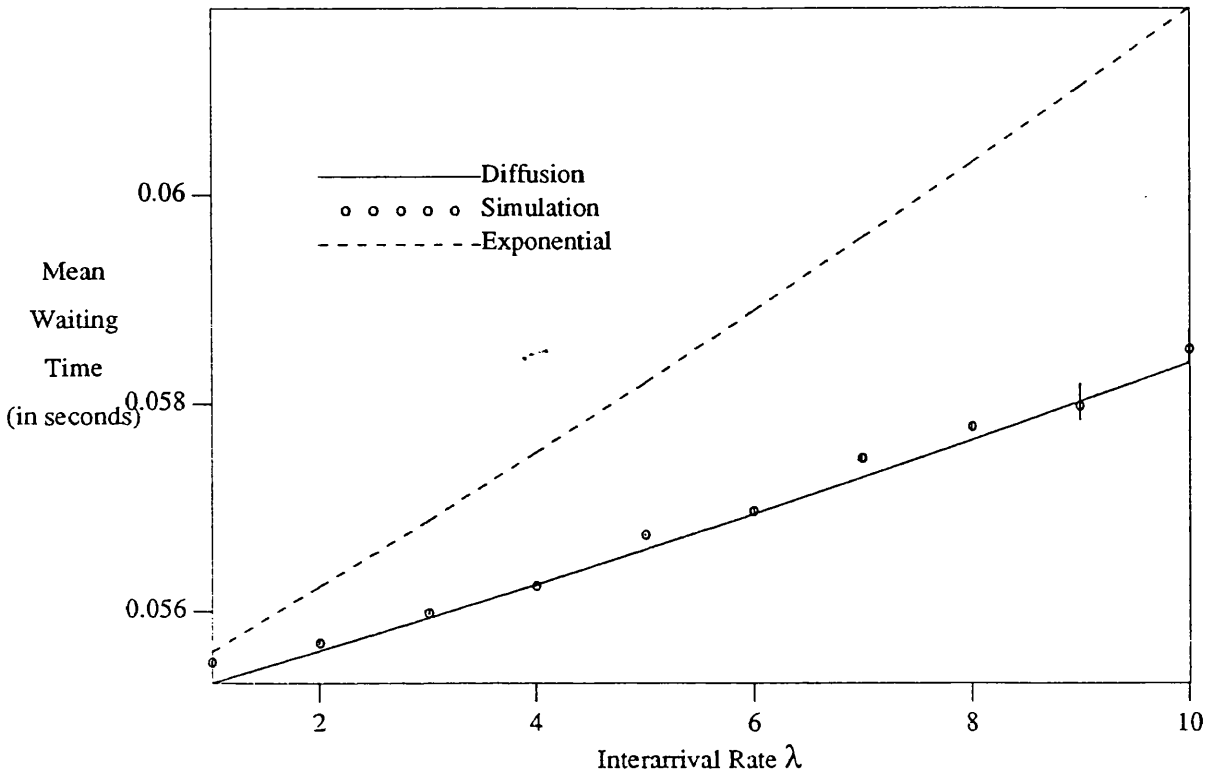


Figure 4.6. Communication Channel with Deterministic Distribution

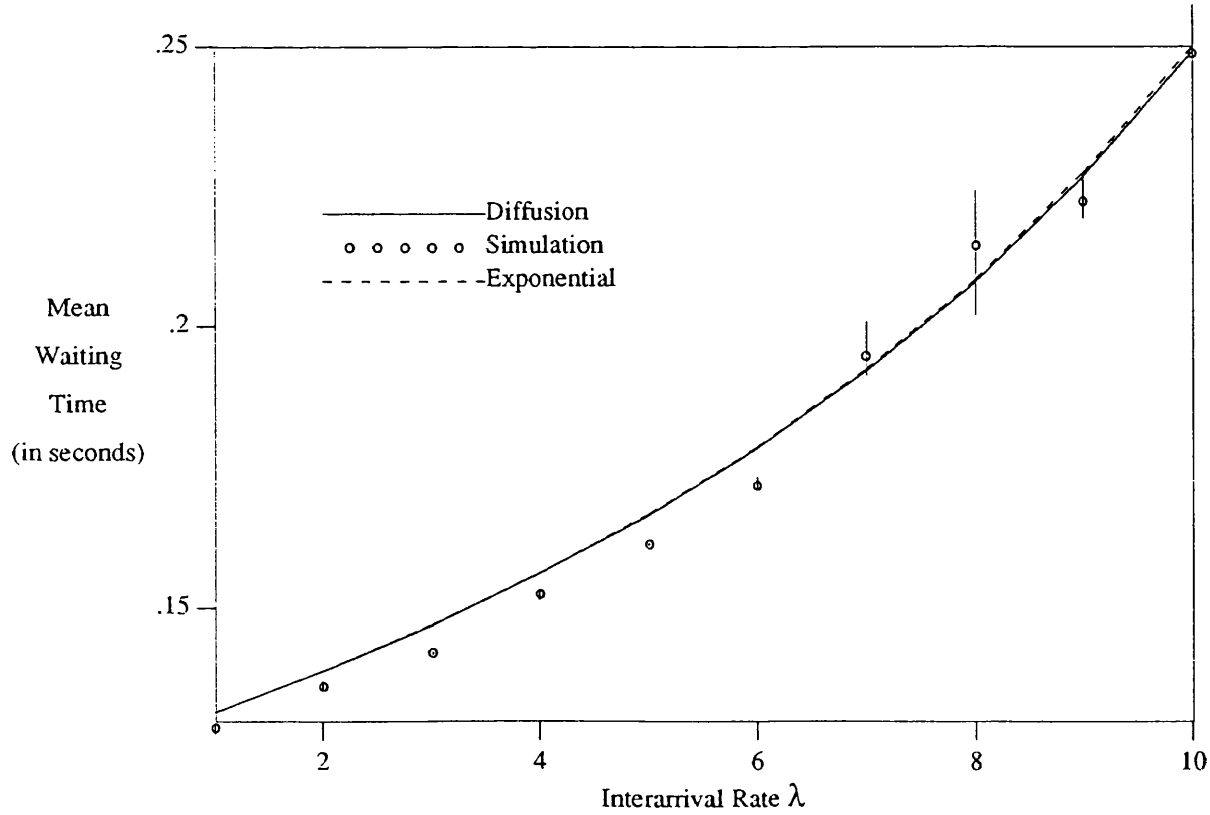


Figure 4.7. Transaction Execution Center with Exponential Distribution

Figure 4.10 illustrates the useful computer time at each node. Simulation results fit well into the diffusion approximation and Jackson's results since the service demand of the computer in the execution centre is exponentially distributed.

Figure 4.11 gives the overall turnaround time of the system. We can see that the simulation results are again very close to the diffusion approximation results. Therefore we can say that our analytic model is well verified.

From the comparison of results, we can see that the analytic results agree with the simulatin results with a high degree of accuracy. In the case of non-exponential service time, the diffusion approximation result is much more accurate than the conventional queueing network results.

4.6 Case Studies

Before starting evaluation, the service time distributions of the components of a distributed database, such as the communication service centre, the locking service centre and the transaction execution service centre, have to be obtained by measurements[54].

The analytic performance evaluation of computer systems, such as the distributed databases and computer communication, etc requires considerably accurate specifications of the service time distributions of the system. In appendix C, the evaluation method to obtain the discrete service time distributions of various computer components such as communication, and database processing on the basis of the statistical data observed from the real system are introduced. Applying the method, the service time parameters of the test system are given in table 4.3.

In order to study the implications and behaviours of the basic 2PL algorithm under various circumstances, a number of situations defined by sets of modeling parameters are studied by running the analytic model. Firstly the implication of read-write ratio on the performance of the algorithm is studied. Secondly we have investigated the effects of data replication on performance. In the third case, we have studied the performance under various lock granularities. The fourth tests intends to reveal the impacts of read-write ratio vs. data replication. And finally the rates of lock conflicts with different data replications are compared.

Table 4.3. The service time parameters of figure 4.13 to 4.17

Service centre	Mean service rate $(1/s)$	Squared coefficient of variance K^2
Data transfer channel	56.9	0.0032
Lock request scheduling	779.7	1.0
Transaction execution	0.525	0.469

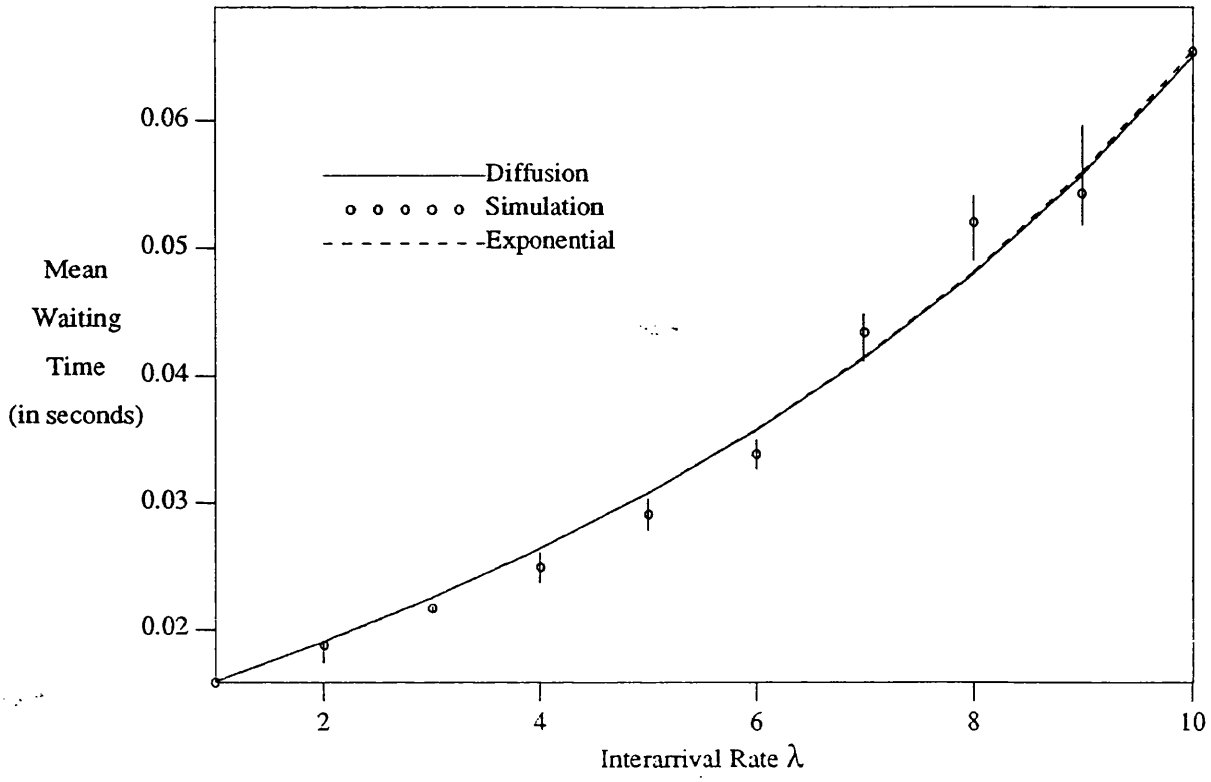


Figure 4.8. Lock Request Center with Exponential Distribution

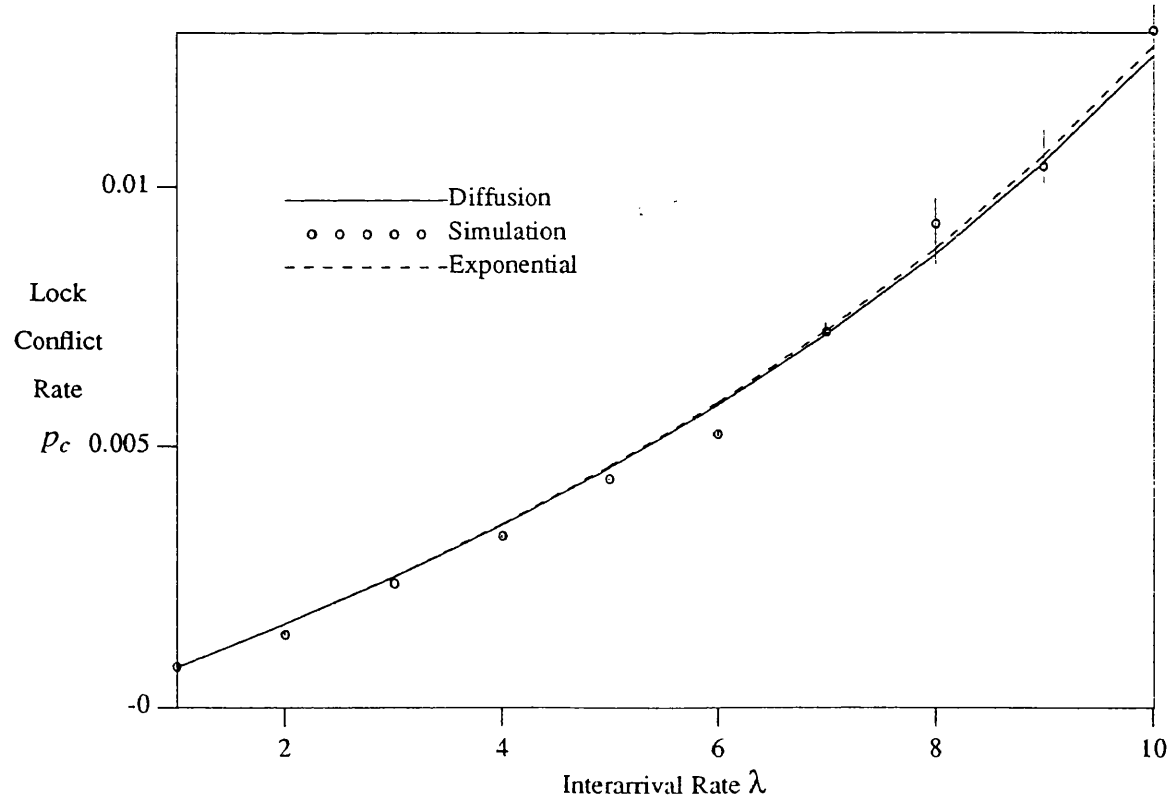


Figure 4.9. Conflict Rate p_c of Lock Request Center

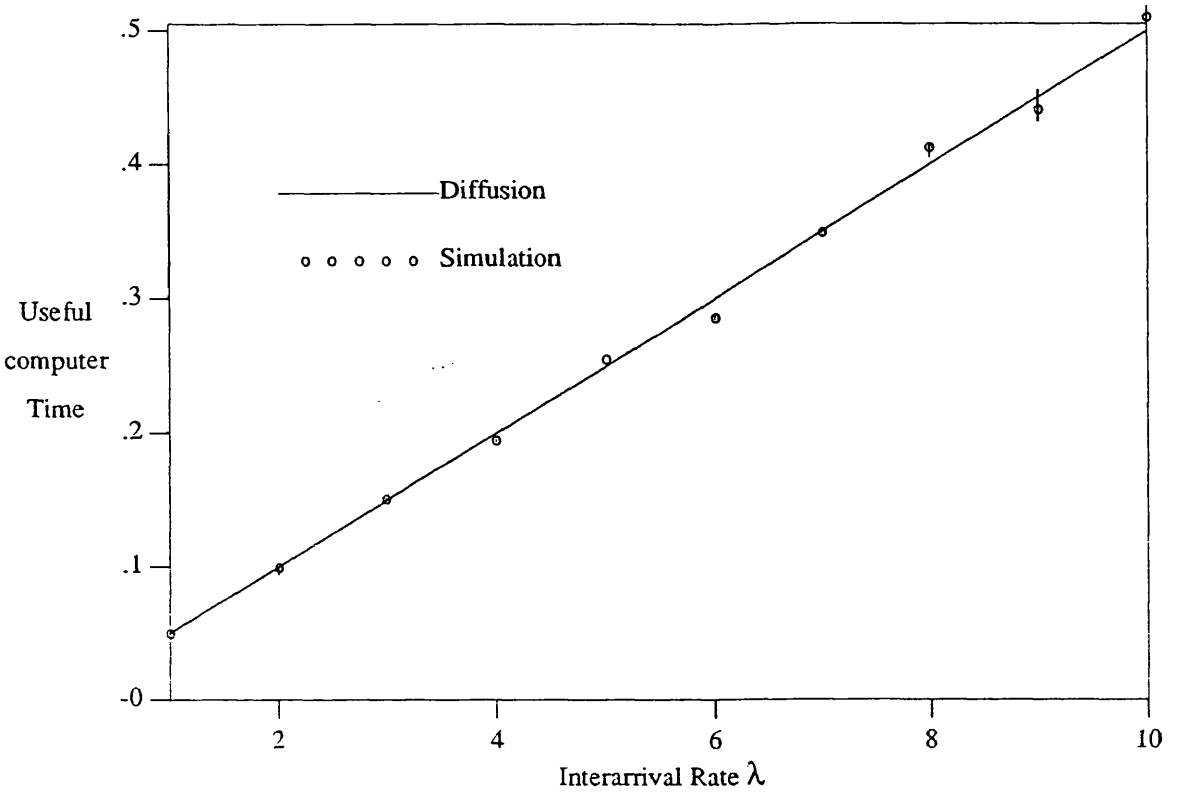


Figure 4.10. Useful Computer Time

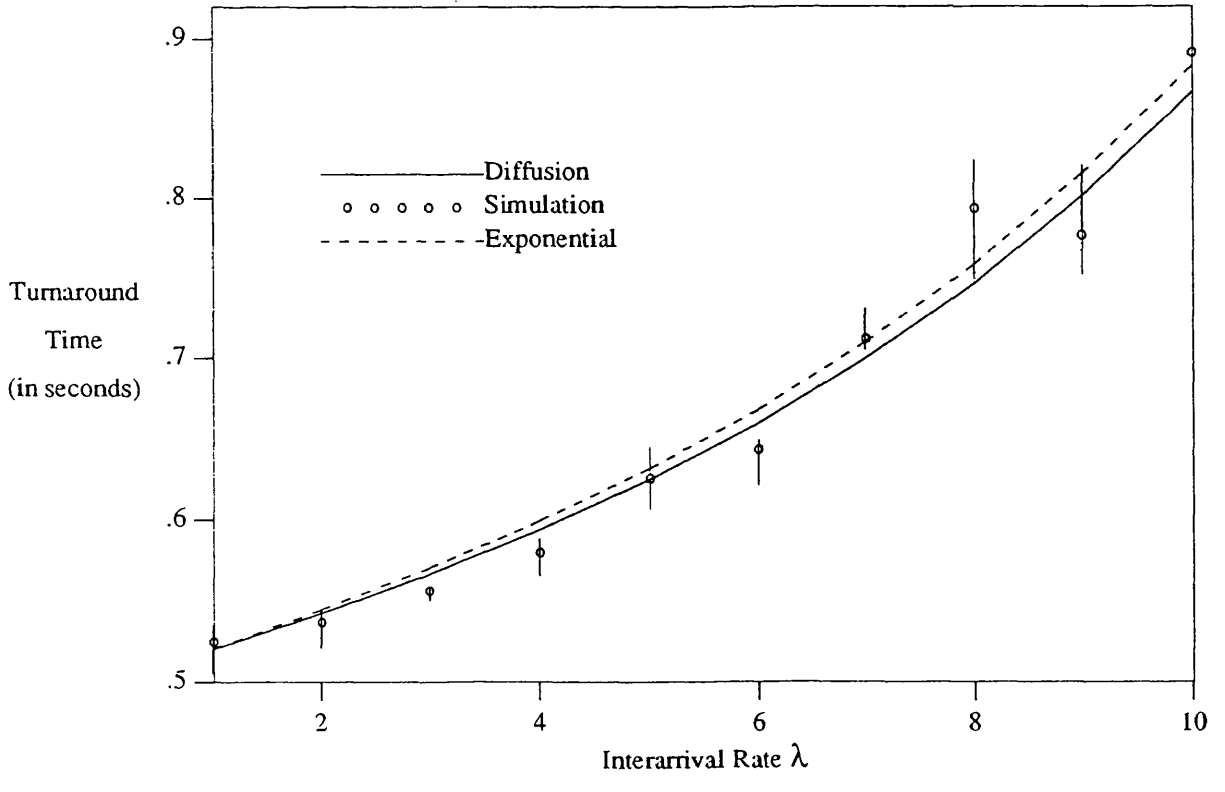


Figure 4.11. Mean Turnaround Time

The interarrival service time is assumed to be exponentially distributed with

$$K_0^2=1$$

The overall system performance is given in terms of mean turnaround time of the system, i.e. the mean response time of the system. The effect of different parameters towards the mean response time is studied. Results are given in figures 4.13 to 4.17.

Read-Write Ratio

Figure 4.13 illustrates the system behavior affected by the read-write ratio γ_{rw} with its parameters given in table 4.4. It can be clearly seen that as the proportion of writes increases the response time also increases. Because the write type transaction generates more messages and demanding more database accesses than read type transaction, the actual work load for high γ_{rw} is greater. When the interarrival rate is below $0.5(1/s)$, the response times of different γ_{rw} are very close to one another. This means that the system can perform almost equally well in lightly loaded condition and it will not be affected very much by the update ratio γ_{rw}, confirming the results reported by Agrawal, Carey and Stonebraker[2,10,11] for single site databases.

Table 4.4. The parameters of figure 4.13

Specification	Value
Number of database nodes K	5
Interarrival rate λ_0' $(1/s)$	0-2
Update ratio γ_{rw}	0.2, 0.4, 0.6, 0.8, 1.0
Number of duplicated copies N_f	2.5
Number of locks requeried r	15
Number of granulars L	1500

Data Replication

The response times under different mean number of duplicated copies are given in figure 4.14. The parameters are given in table 4.5. The response time of the system increases as N_f increases. Comparing figure 4.14 with figure 4.13, we can see that the effect of N_f on the response time is greater than the effect of γ_{rw}. We can also see that the system with no duplicated copies (i.e. $N_f=1$) performs much better than others under the condition of $\gamma_{rw}=0.5$ and $r=15$.

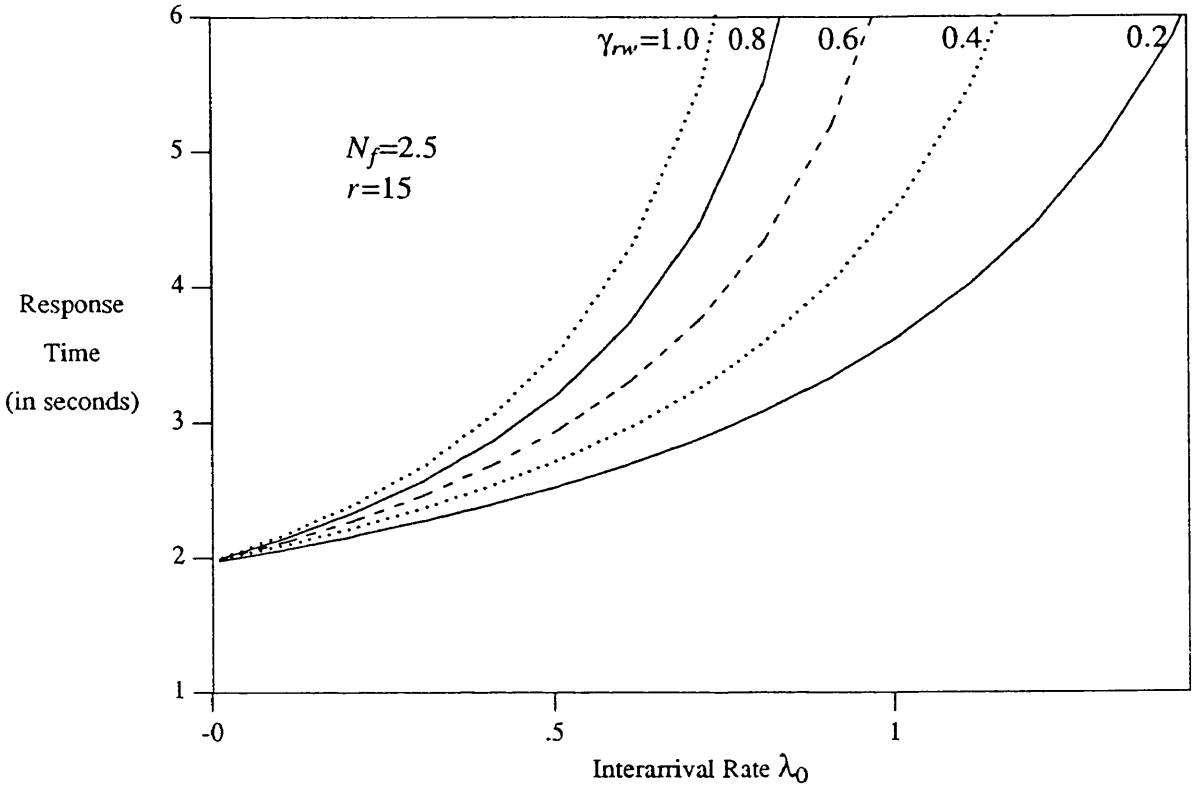


Figure 4.13. Response Times with Different Read-Write Ratio γ_{rw}

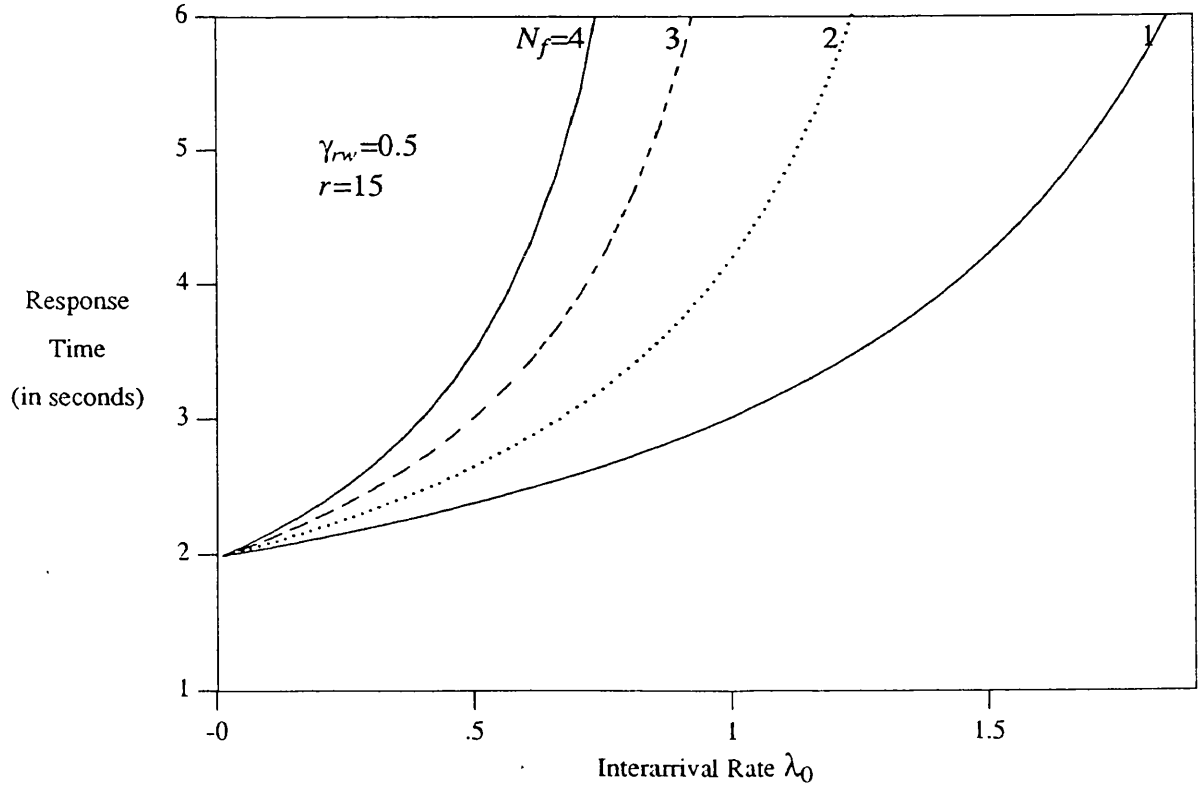


Figure 4.14. Response Time with Different Replicated Copies N_f

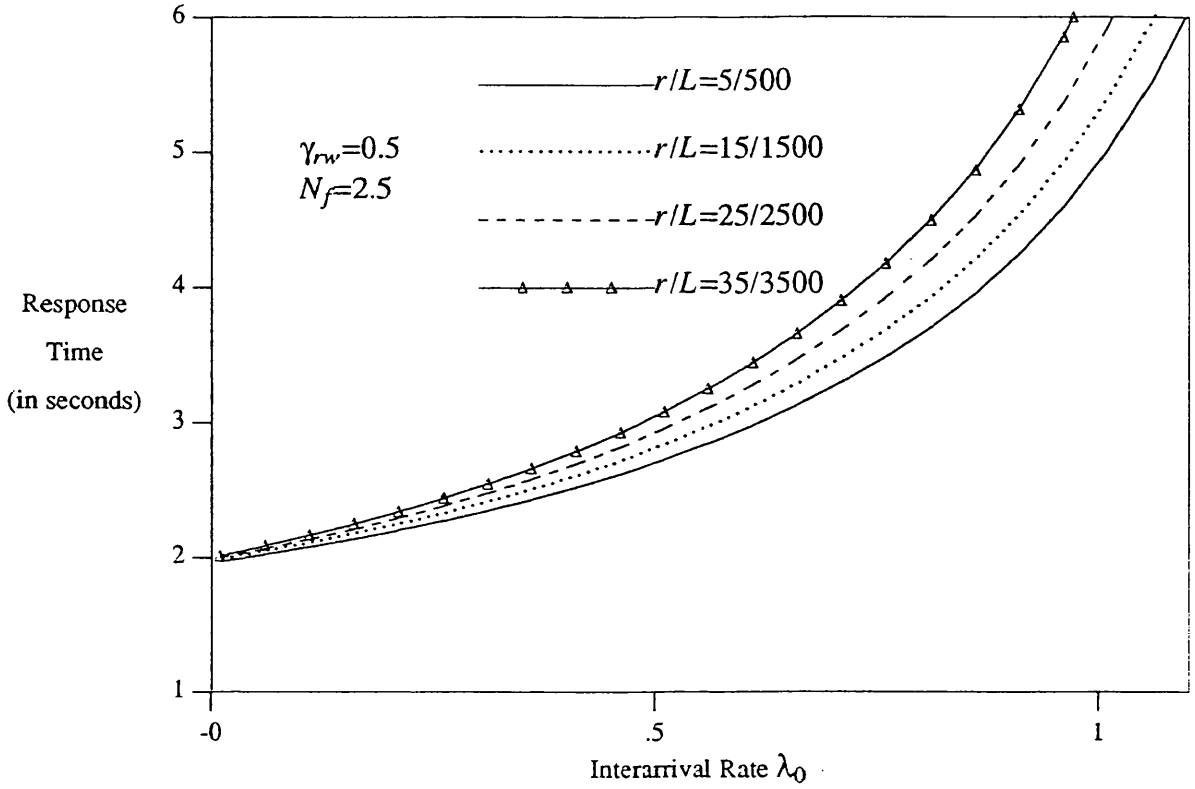


Figure 4.15. Response Times with Different Granularity r/L

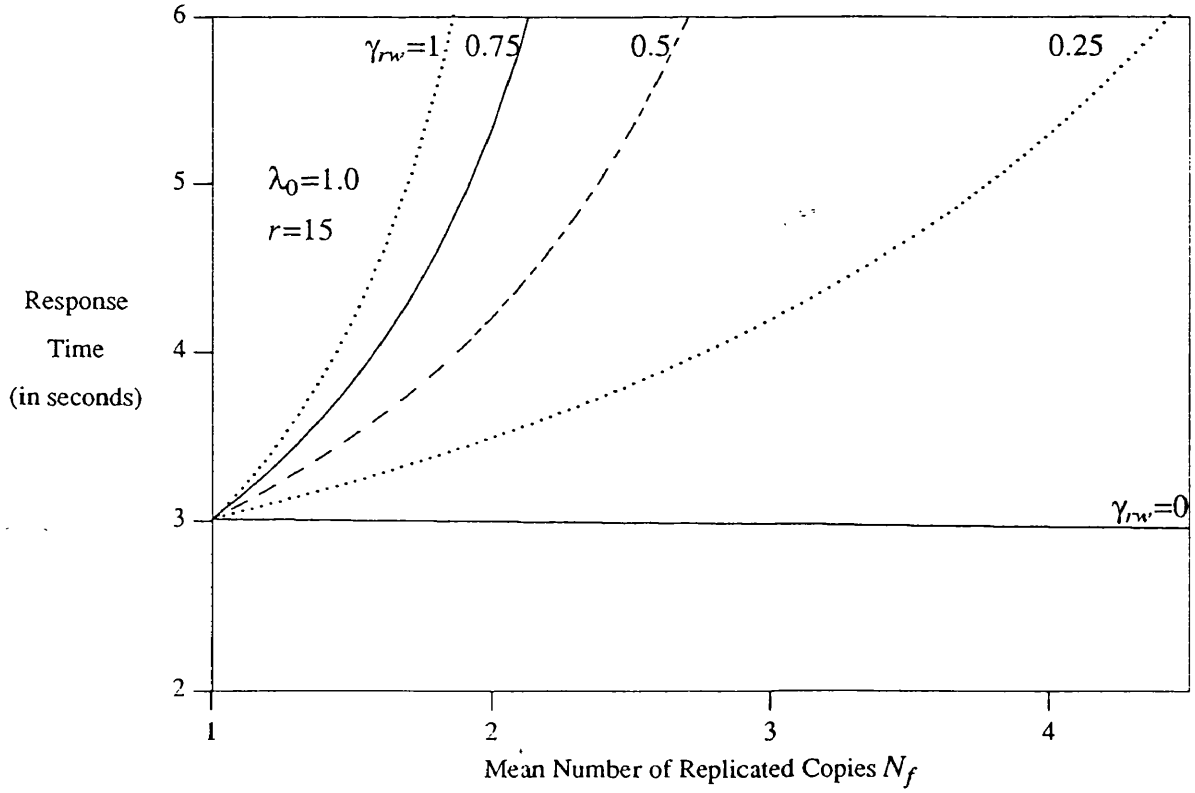


Figure 4.16. Response Times vs N_f with Different Read-Write Ratio γ_{rw}

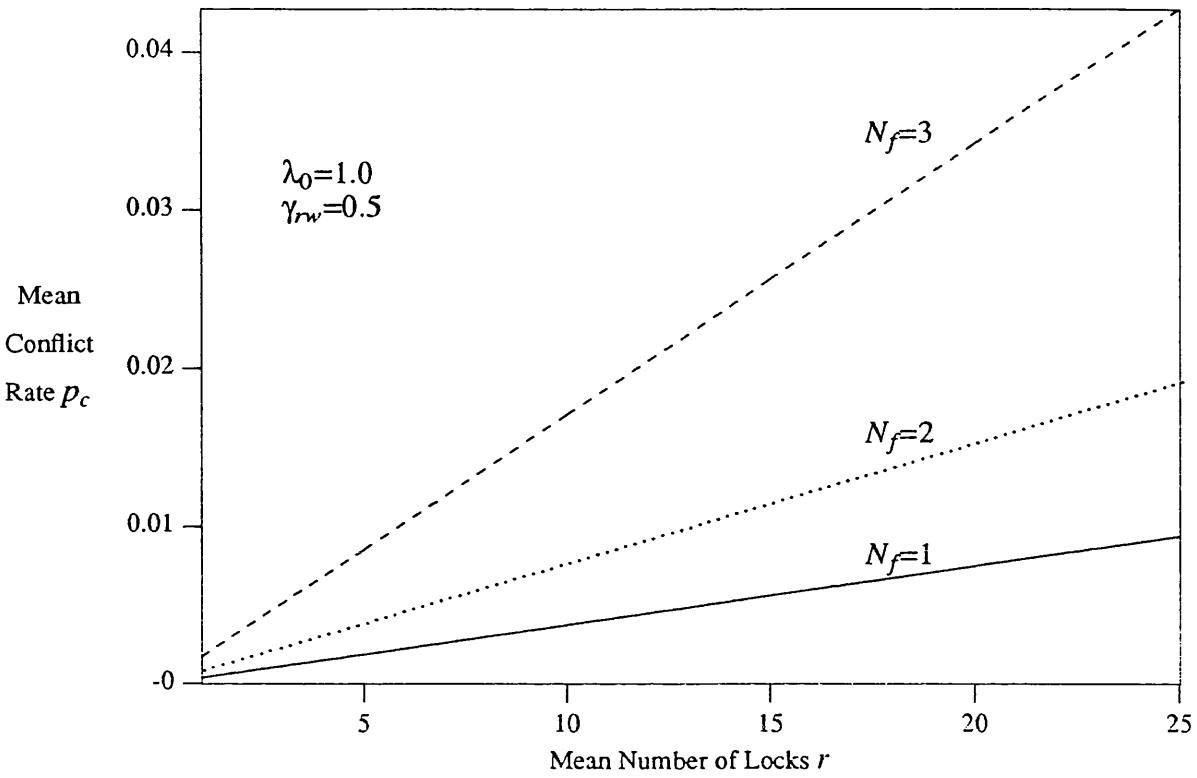


Figure 4.17. Mean Conflict Rates with Different Replicated Copies N_f

Table 4.5. The parameters of figure 4.14

Specification	Value
Number of database nodes k	5
Interarrival rate λ_0' $(1/s)$	0-2.5
Update ratio γ_{rw}	0.5
Number of duplicated copies N_f	1, 2, 3, 4
Number of locks requeried r	15
Number of granulars L	1500

Lock Granularity

Figure 4.15 shows the effect of database granularity with the parameters are given in table 4.6. The mean number of locks for each transaction vs. the total number of granulars is set at 5/500, 15/1500, 25/2500, 35/3500 respectively, which simulates transactions with the same data-size but different lock granularity. The results suggest that coarse granularity performs slightly better than fine granularity under the condition of γ_{rw}=0.5 and N_f=2.5. The results suggest that the effect of the database granularity is

relatively small, because of the small locking overhead.

Table 4.6. The parameters of figure 4.15	
Specification	Value
Number of database nodes k	5
Interarrival rate $\lambda_0^{'}$ $(1/s)$	0-1.4
Update ratio γ_{rw}	0.5
Number of duplicated copies N_f	2.5
Number of locks requeried r	5, 15, 25, 35
Number of granulars L	500, 1500, 2500, 3500

Read-Write Ratio vs. Data Replication

Figure 4.16 shows the change of response time with N_f under different γ_{rw}. It is not surprising that when all the transactions are of read only type, i.e. γ_{rw}=0, the response time decrease as the number of duplicated copies increases. The response time then increases slowly as the number of duplicated copies increases when the update type transaction only occupies a small portion; γ_{rw}=0.25. When the read-write ratio γ_{rw} grows bigger the response time increases dramatically with N_f. This quantitative result can be used as a guidance for DDBMS system design. When the system is highly update oriented, the optimal mean duplicated copies should be set to one, which means no duplication at all. The parameters are given in table 4.7.

Table 4.7. The parameters of figure 4.16	
Specification	Value
Number of database nodes k	5
Interarrival rate $\lambda_0^{'}$ $(1/s)$	1
Update ratio γ_{rw}	0, 0.25, 0.5, 0.75, 1
Number of duplicated copies N_f	1.0 -4.5
Number of locks requeried r	15
Number of granulars L	1500

Conflict Rate

Figure 4.17 indicates how the number of duplicated copies N_f affects the conflict rate of lock request. The higher the N_f, the greater the conflict rate p_c. So N_f should be carefully chosen to obtain a good system performance. The parameters are given in table 4.8.

Table 4.8. The parameters of figure 4.17

Specification	Value
Number of database nodes k	5
Interarrival rate $\lambda_0^{'}$ $(1/s)$	1
Update ratio γ_{rw}	0.5
Number of duplicated copies N_f	1, 2, 3, 4
Number of locks requeried r	0-30
Number of granulars L	1500

The case studies illustrate a distributed database from various angles. It helps us to understand the effects of various factors to the overall performance of the system. The scope of obtaining results from the analytic method is unlimited.

4.7 Summary

In this chapter, we have introduced a novel method to define a distributed database systematically. This method very much simplifies the task of defining a queueing network of a distributed database. An extended diffusion approximation method is used to model a distributed database with non-Poisson interarrival process and non-exponential service time. The analytic model is further validated by simulation model. Very good agreements are achieved.

Chapter 5. Major Locking Protocols in DDB

In the previous chapters we have introduced the method to model the performance of basic two phase locking protocol. However there are quite a few concurrency control protocols based on two phase locking, among which primary copy 2PL, majority consensus 2PL and centralized 2PL are the most well known locking protocols used in distributed database. It is the purpose of this chapter to evaluate and compare these different locking algorithms by using the consistent modeling method introduced in the previous two chapters and show the integrity of the modeling method by applying it to these popular two phase locking algorithms.

Section 5.1 evaluates the primary copy 2PL protocol. Section 5.2 estimates the performance of the majority consensus 2PL protocol. In section 5.3 the centralized 2PL protocol is modeled. Section 5.4 uses the analytic results to compare these 2PL protocols.

5.1 Primary Copy 2PL

5.1.1 System Specification

The primary copy 2PL is specially designed for distributed databases with replicated data copies[76]. One of the physical copies of each logical data item is assigned as the primary copy. Locking can only be applied to the primary copy. The primary copy 2PL approach simplifies the locking procedure and prevents dead locks at the cost of extra communication. For example, suppose a logical data item X has N_f copies $x_1, x_2, \cdots, x_i, \cdots, x_{N_f}$, where x_1 is defined as the primary copy. A read transaction $Read(X)$ trying to read a copy x_i other than the primary copy x_1 has to access the primary copy x_1 first to own a read lock at the first phase of two phase commit protocol. It then accesses copy x_i and releases locks on the primary copy x_1 at second phase of two phase commit. The $Read(X)$ procedure is shown in figure 5.1

The $Write(X)$ transaction of the primary 2PL is similar to that of basic 2PL. The only difference is that locking is performed on the primary copy rather than on all the copies. At the first phase of 2PL a $Prepare(X)$ is performed on all the copies while only the $Prepare(X)$ on the primary copy implies lock request for data item X. At the second phase of the 2PL a $dm-write(X)$, which means a $Write(X)$ applied on a physical database by a DM, is propagated to all the copies $x_i (i=1,...,N_f)$, while only at the primary copy x_1 a $dm-write(X)$ implicitly releases locks as shown in figure 5.2. The primary copy 2PL protocol is formally represented as follows:

Primary Copy 2PL Protocol

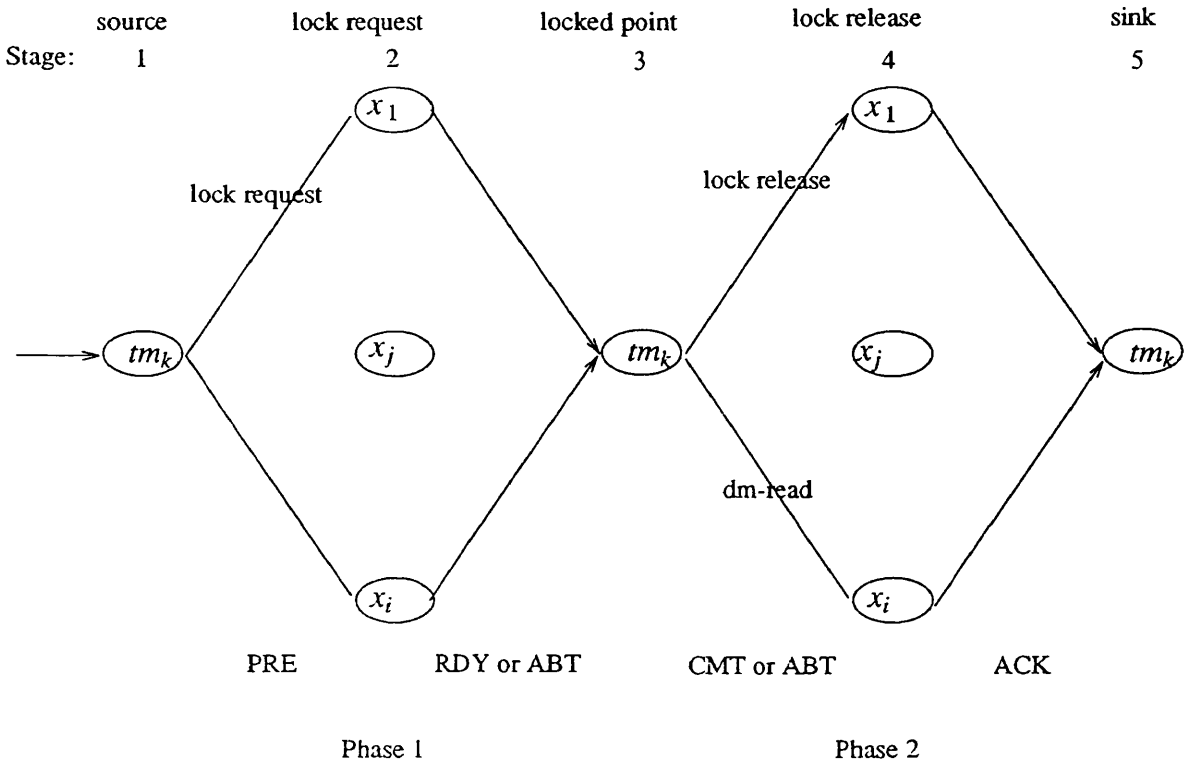


Figure 5.1. Concurrency Control Structure of $Read(X)$ of Primary Copy 2PL

Phase 1:

1.1 The coordinator TM_k sends $Prepare(X)$ and lock request (PRE) message to the participants $\{DM_i, X \in DM_i\}$.

1.2 Each DM_i receives the message. If the primary copy x_1 is stored at DM_i, it checks for the requested locks. If all locks are granted and it is willing to commit, then it writes the transaction's record in the log and sends a ready (RDY) answer message to TM_k. Otherwise it sends abort (ABT) message to TM_k. If the primary copy is not stored at DM_i, it simply sends a RDY to TM_k.

Phase 2:

2.1 Upon receiving the answer message RDY or ABT from $\{DM_i, X \in DM_i\}$, TM_k sends commit (CMT) message to all $\{DM_i, X \in DM_i\}$ if all of them have voted RDY, otherwise it writes an abort record and sends ABT to all $\{DM_i, X \in DM_i\}$.

2.2 After receiving the command message from TM_k, each DM_i writes either an abort or commit record in log; then executes the transaction and releases the locks at $\{DM_i, X \in DM_i\}$; and sends the acknowledge (ACK) message to TM_k.

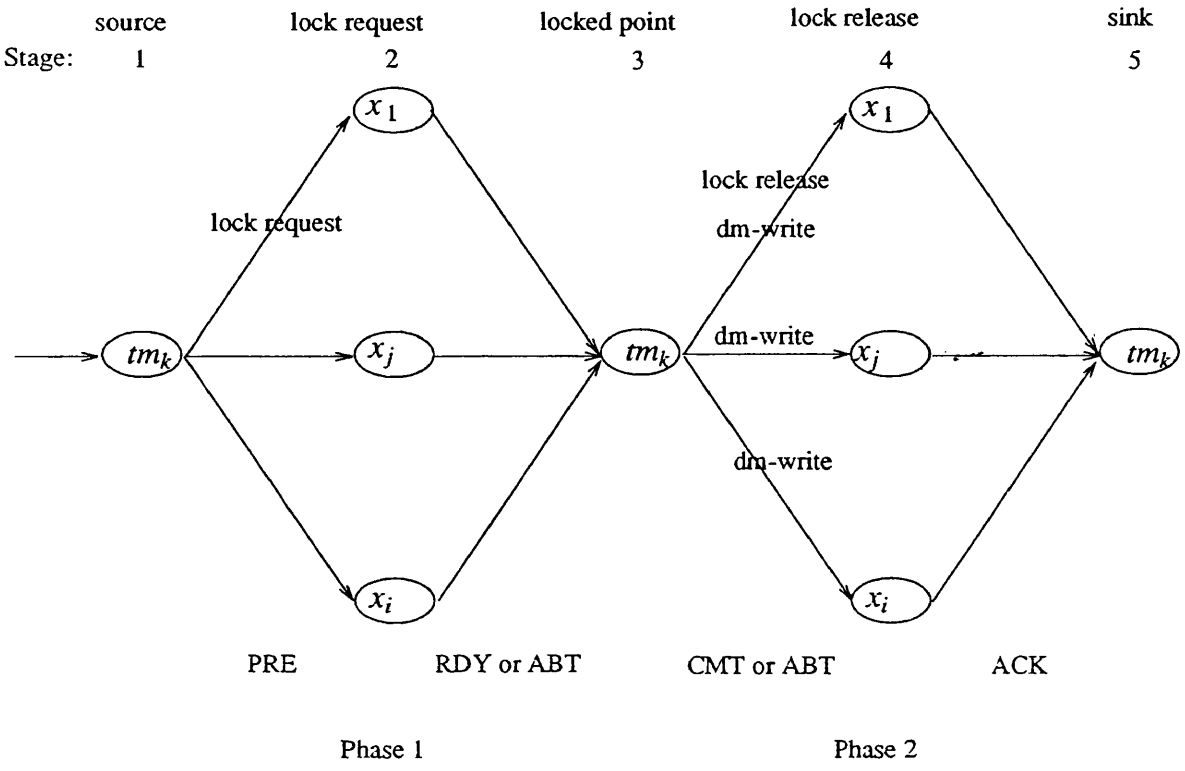


Figure 5.2. Concurrency Control Structure of $Write\,(X)$ of Primary Copy 2PL

2.3 TM_k waits for ACK messages from all $\{DM_i, X \in DM_i\}$; then writes a completion record in log.

5.1.2 Model Definition

The modeling method for the primary copy 2PL is similar to that of basic 2PL introduced in chapters 3 and 4. We shall use the same method to develop models for the primary copy 2PL.

Access Rules

We shall first formally define the read and write rules of the primary copy 2PL.

Definition 5.1: The rules of $Read\,(X)$ issued from TM_k are as follows:

- If $x_1 \in DM_k$, $lock\,(x_1)$ and $dm\text{-}read\,(x_1)$ at DM_k;
- If $\overline{x_1 \in DM_k}$ and $X \in DM_k$, $lock\,(x_1)$ at $DM_i (x_1 \in DM_i)$ and $dm\text{-}read\,(X)$ at DM_k;
- If $\overline{x_1 \in DM_k}$ and $\overline{X \in DM_k}$, $lock\,(x_1)$ and $dm\text{-}read\,(x_1)$ at $DM_i (x_1 \in DM_i)$.

Definition 5.2: The rules of $Write\,(X)$ issued from TM_k are as follows:

- If $x_1 \in DM_j$, $lock(x_1)$ and $write(x_1)$ at DM_j;
- If $\overline{x_1 \in DM_j}$ and $X \in DM_j$, $lock(x_1)$ at $DM_i (x_1 \in DM_i)$ and $dm\text{-}write(X)$ at DM_j.

Access Pattern

Suppose the primary copy x_1 is stored in one of the DMs with equal probability. The probability of the primary copy stored at DM_k is therefore given by

$$Prob\{x_1 \in DM_k\} = \frac{1}{K} \quad (k=1, \cdots, K) \tag{5.1a}$$

and the probability of a non-primary copy stored at DM_k is given by

$$Prob\{X \in DM_k \mid \overline{x_1 \in DM_k}\} = \frac{N_f - 1}{K-1} \tag{5.1b}$$

The rules of the primary copy 2PL imply that a read transaction can generate one or two DM accesses, while a write transaction can produce N_f DM accesses. The mean number of DM accesses by one transaction issued from TM_k is given by

$$\begin{aligned} N_{acc} = {} & 1 \cdot Prob\{read \mid x_1 \in DM_k\} \\ & + 2 \cdot Prob\{read \mid \overline{x_1 \in DM_k} \cap X \in DM_k\} \\ & + 1 \cdot Prob\{read \mid \overline{x_1 \in DM_k} \cap \overline{X \in DM_k}\} \\ & + N_f \cdot Prob\{write\} \end{aligned} \tag{5.2}$$

where $Prob\{read\}$ and $prob\{write\}$ are defined as the transaction being a read or write type respectively. Since $Prob\{read\}$ and $Prob\{write\}$ are independent of the locations of the data copies, the equation becomes

$$\begin{aligned} N_{acc} = {} & 1 \cdot Prob\{read\} \cdot Prob\{x_1 \in DM_k\} \\ & + 2 \cdot Prob\{read\} \cdot Prob\{\overline{x_1 \in DM_k} \cap X \in DM_k\} \\ & + 1 \cdot Prob\{read\} \cdot Prob\{\overline{x_1 \in DM_k} \cap \overline{X \in DM_k}\} \\ & + N_f \cdot Prob\{write\} \end{aligned}$$

The probabilities of $Prob\{read\}$ and $Prob\{write\}$ is given by

$$Prob\{read\} = 1 - \gamma_{rw_k}$$

and

$$Prob\{write\} = \gamma_{rw_k}$$

The rest of probabilities in equation (5.2) can be derived by

$$Prob\{\overline{x_1 \in DM_k} \cap X \in DM_k\}$$
$$= Prob\{\overline{x_1 \in DM_k}\} Prob\{X \in DM_k \mid \overline{x_1 \in DM_k}\}$$
$$= \left[1 - \frac{1}{K}\right]\left[\frac{N_f - 1}{K-1}\right]$$

$$Prob\{\overline{x_1 \in DM_k} \cap \overline{X \in DM_k}\}$$
$$= Prob\{\overline{x_1 \in DM_k}\} Prob\{\overline{X \in DM_k} \mid \overline{x_1 \in DM_k}\}$$
$$= \left[1 - \frac{1}{K}\right]\left[1 - \frac{N_f - 1}{K-1}\right]$$

Substituting the above probabilities in the equation of (5.2), the number of DM accesses generated by one transaction issued at TM_k can be written as

$$N_{acc_k} = (1-\gamma_{rw_k})\frac{1}{K} + 2(1-\gamma_{rw_k})\left[1-\frac{1}{K}\right]\left[\frac{N_f-1}{K-1}\right] + (1-\gamma_{rw_k})\left[1-\frac{1}{K}\right]\left[1-\frac{N_f-1}{K-1}\right] + \gamma_{rw_k}N_f$$
$$= (1-\gamma_{rw_k})\left[1+\frac{N_f-1}{K}\right] + \gamma_{rw_k}N_f \qquad (5.3)$$

N_{acc_k} also determines the ratio between the total number of DM accesses generated and the total number of transaction issued. Suppose λ'_k is the arrival rate of transaction issued at TM_k. The rate of DM accesses generated at TM_k is therefore

$$\lambda_k = N_{acc_k}\lambda'_k$$
$$= \left[(1-\gamma_{rw_k})\left[1+\frac{N_f-1}{K}\right] + \gamma_{rw_k}N_f\right]\lambda'_k \qquad (5.4)$$

With the above results, now we are ready to derive the access pattern matrix $\mathbf{W}$.

Theorem 5.3: The access pattern matrix $\mathbf{W}$ of the primary copy 2PL is given by

$$\omega_{kj} = \begin{cases} \dfrac{\frac{N_f}{K}}{(1-\gamma_{rw_k})(1+\frac{N_f-1}{K})+\gamma_{rw_k}N_f} & j=k \text{ and } k=1,\ldots,K \\ \dfrac{(1-\gamma_{rw_k})\frac{1}{K}+\gamma_{rw_k}\frac{N_f}{K}}{(1-\gamma_{rw_k})(1+\frac{N_f-1}{K})+\gamma_{rw_k}N_f} & j\neq k \text{ and } k=1,\ldots,K \end{cases} \qquad (5.5a)$$

Proof: From definition 5.1 and 5.2, we immediately have

$$\begin{aligned}\omega_{kk} = & Prob\{Access_{kk} \mid x_1 \in DM_k\} Prob\{x_1 \in DM_k\} \\ & + Prob\{Access_{kk} \mid \overline{x_1 \in DM_k} \cap X \in DM_k\} Prob\{\overline{x_1 \in DM_k} \cap X \in DM_k\} \\ & + Prob\{Access_{kk} \mid \overline{x_1 \in DM_k} \cap \overline{X \in DM_k}\} Prob\{\overline{x_1 \in DM_k} \cap \overline{X \in DM_k}\}\end{aligned} \tag{5.5b}$$

Under the condition of $\{x_1 \in DM_k\}$ both $dm-read(X)$ and $dm-write(X)$ need to access node DM_k. The probability of access is

$$Prob\{Access_{kk} \mid x_1 \in DM_k\} Prob\{x_1 \in DM_k\} = \frac{1 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}}$$

If there is a nonprimary copy in DM_k, i.e. $\{\overline{x_1 \in DM_k} \cap X \in DM_k\}$, the $dm-read(X)$ accesses DM_k to read and the $dm-write(X)$ accesses DM_k to write. The probability of access is given by

$$Prob\{Access_{kk} \mid \overline{x_1 \in DM_k} \cap X \in DM_k\} = \frac{1 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}}$$

If no copy is stored at DM_k, there is no need to access it. Therefore

$$Prob\{Access_{kk} \mid \overline{x_1 \in DM_k} \cap \overline{X \in DM_k}\} = 0$$

Substituting the above probabilities into the equation of (5.5b), we can immediately obtain

$$\begin{aligned}\omega_{kk} = & \frac{1 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}} \cdot \frac{1}{K} \\ & + \frac{1 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}} \cdot \left(1 - \frac{1}{K}\right)\left(\frac{N_f - 1}{K - 1}\right) \\ & + \frac{0 \cdot Prob\{read\} + 0 \cdot Prob\{write\}}{N_{acc_k}} \left(1 - \frac{1}{K}\right)\left(1 - \frac{N_f - 1}{K - 1}\right) \\ = & \frac{1}{N_{acc_k}}\left[\frac{1}{K} + \left(1 - \frac{1}{K}\right)\left(\frac{N_f - 1}{K - 1}\right)\right] \\ = & \frac{1}{N_{acc_k}} \cdot \frac{N_f}{K} \\ = & \frac{\frac{N_f}{K}}{(1 - \gamma_{rw_k})(1 + \frac{N_f - 1}{K}) + \gamma_{rw_k} N_f}\end{aligned}$$

Similarly

$$\begin{aligned}\omega_{\substack{kj \\ (j \neq k)}} = & Prob\{Access_{kj} \mid x_1 \in DM_j\} \cdot Prob\{x_1 \in DM_j\} \\ & + Prob\{Access_{kj} \mid \overline{x_1 \in DM_j} \cap X \in DM_j\} \cdot Prob\{\overline{x_1 \in DM_j} \cap X \in DM_j\} \\ & + Prob\{Access_{kj} \mid \overline{x_1 \in DM_j} \cap \overline{X \in DM_j}\} \cdot Prob\{\overline{x_1 \in DM_j} \cap \overline{X \in DM_j}\}\end{aligned} \tag{5.5c}$$

Under the condition of $\{x_1 \in DM_j\}$ read and write transaction will both access the primary copy at DM_j, that is

$$\begin{aligned} &Prob\{Access_{kj} \mid x_1 \in DM_j\} \\ &\quad = \frac{1 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc}} \\ &\quad = \frac{1}{N_{acc}} \end{aligned}$$

The case of $\{\overline{x_1 \in DM_j} \cap X \in DM_j\}$ deserves some comments. According to the read rule of the primary copy 2PL, a read transaction will access the primary copy and the local copy. In this case, $Read(X)$ will not access DM_j, since the primary copy is not stored at DM_j and DM_j is not a local node $(j \neq k)$. Therefore we have

$$\begin{aligned} Prob &= \{Access_{kj} \mid \overline{x_1 \in DM_j} \cap X \in DM_j\} \\ &= \frac{0 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc}} \\ &= \frac{\gamma_{rw_k}}{N_{acc}} \end{aligned}$$

under the condition of $\{\overline{x_1 \in DM_j} \cap \overline{X \in DM_j}\}$ there is no need to access DM_j. That is

$$Prob\{Access_{kj} \mid \overline{x_1 \in DM_j} \cap \overline{X \in DM_j}\} = 0$$

Substituting the above probabilities to equation (5.5c), we have

$$\omega_{kj} = \frac{1}{N_{acc}} \cdot \frac{1}{K} + \frac{\gamma_{rw_k}}{N_{acc}} \left[1 - \frac{1}{K}\right] \left[\frac{N_f - 1}{K - 1}\right] + 0$$

the primary copy 2PL can be constructed in the same way as that of the basic 2PL system. The overall structure of the queueing network model is shown in figure 4.2.

With the newly derived access pattern matrix $\mathbf{W}$, the arrival rate matrix and the Markov chain matrix of the primary copy 2PL queueing network model can be obtained by using the communication flow method introduced in section 4.2.2. The arrival rates are given by

$$\begin{aligned} \Lambda_0 &= \left[\lambda_1 \;\; \lambda_2 \;\; \cdots \;\; \lambda_K\right] \\ &= \left[N_{acc_1}\lambda_1' \;\; N_{acc_2}\lambda_2' \;\; \cdots \;\; N_{acc_K}\lambda_K'\right] \end{aligned} \tag{5.6a}$$

$$\begin{aligned} \Lambda_1 &= \mathbf{V}^T \Lambda_1 \\ &= \left[\lambda_1\omega_{11} + \lambda_2\omega_{21} + \ldots + \lambda_K\omega_{K1} \;\; \lambda_1\omega_{12} + \lambda_2\omega_{22} + \ldots + \lambda_K\omega_{K2} \;\; \cdots \;\; \lambda_1\omega_{1K} + \lambda_2\omega_{2K} + \ldots + \lambda_K\omega_{KK}\right] \end{aligned} \tag{5.6b}$$

$$\Lambda_2 = \mathbf{V}^T \mathbf{A}_2 \tag{5.6c}$$
$$= \left[\lambda_1\omega_{11}+\lambda_2\omega_{12}+...+\lambda_K\omega_{1K} \;\; \lambda_1\omega_{21}+\lambda_2\omega_{22}+...+\lambda_K\omega_{2K} \;\; \cdots \lambda_1\omega_{K1}+\lambda_2\omega_{K2}+...+\lambda_K\omega_{KK}\right]$$

$$\Lambda_3 = \mathbf{V}^T \mathbf{A}_3 \tag{5.6d}$$
$$= \left[\lambda_1\omega_{11}+\lambda_2\omega_{21}+...+\lambda_K\omega_{K1} \;\; \lambda_1\omega_{12}+\lambda_2\omega_{22}+...+\lambda_K\omega_{K2} \;\; \cdots \lambda_1\omega_{1K}+\lambda_2\omega_{2K}+...+\lambda_K\omega_{KK}\right]$$

$$\Lambda_4 = \mathbf{V}^T \mathbf{A}_4 \tag{5.6e}$$
$$= \left[\lambda_1\omega_{11}+\lambda_2\omega_{12}+...+\lambda_K\omega_{1K} \;\; \lambda_1\omega_{21}+\lambda_2\omega_{22}+...+\lambda_K\omega_{2K} \;\; \cdots \lambda_1\omega_{K1}+\lambda_2\omega_{K2}+...+\lambda_K\omega_{KK}\right]$$

and the communication flow matrix is given by

$$\Lambda_c = \sum_{s=1}^{4} \mathbf{A}_s \tag{5.6f}$$
$$= 2\begin{bmatrix} 2\lambda_1\omega_{11} & \lambda_2\omega_{21}+\lambda_1\omega_{12} & \cdots & \lambda_K\omega_{K1}+\lambda_1\omega_{1K} \\ \lambda_1\omega_{12}+\lambda_2\omega_{21} & 2\lambda_2\omega_{22} & \cdots & \lambda_K\omega_{K2}+\lambda_2\omega_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_1\omega_{1K}+\lambda_K\omega_{K1} & \lambda_2\omega_{2K}+\lambda_K\omega_{K2} & \cdots & 2\lambda_K\omega_{KK} \end{bmatrix}$$

and the Markov chain matrices are given by

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_I & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_O \end{bmatrix} \tag{5.7}$$

Locking

The primary copy 2PL locks only the primary copy of the logical data item. It generates less locking activities in a distributed database system than the basic 2PL algorithm.

Suppose the mean number of locks required by a transaction at TM_k is denoted by r_k. A transaction generates an average of N_{acc} DM accesses, where N_{acc} is defined as the mean number of DM accesses by on transaction.

$$N_{acc} = \sum_{k=1}^{K} N_{acc_k}$$

Only one of them requires locks on the primary copy. If we observe the system from the view point of each $DM_k(k=1,...K)$, one out of N_{acc} accesses is to request locks, which means that only one out of N_{acc} arrivals to a DM node needs locking service. Therefore the actual arrival rate of the lock request centre is equal to

$$\hat{\lambda}_{k_{K+1}} = \frac{\lambda_{k_{K+1}}}{N_{acc}} \tag{5.8}$$

Substituting the above equation into equation (4.55), the mean number of locks held at the lock request centre can be modified as

$$NL_k^{(lock)} = \frac{\lambda_{k_{K+1}}}{q_k^{r_k} N_{acc} \mu_s} \sum_{i=1}^{r_k-1} i q_k^i \tag{5.9}$$

The mean number of transaction execution (i.e. $dm-read(X)$ and $dm-write(X)$) generated by one transaction is given by

$$\begin{aligned} N_{ex} &= 1 \cdot Prob\{read\} + N_f \cdot Prob\{write\} \\ &= 1-\gamma_{rw} + N_f \gamma_{rw} \end{aligned} \tag{5.10}$$

where

$$\gamma_{rw} = \sum_{k=1}^{K} \frac{\gamma_{rw_k}}{K} \tag{5.11}$$

It should be noted that only N_{ex} out of N_{acc} DM accesses involve with transaction executions; while the rest only involve with releasing locks, which means that only N_{ex}/N_{acc} percent of the transactions enter the transaction execution centre. The actual arrival rate of the transaction execution centre is therefore

$$\hat{\lambda}_{k_{K+2}} = \frac{N_{ex}}{N_{acc}} \lambda_{k_{K+2}}$$

At the execution stage, only one out of N_{acc} transactions holds locks. The actual number of locks held at the execution stage is

$$NL_k^{(ex)} = \frac{r_k}{N_{acc}} \bar{n}_{k_{K+2}} \tag{5.12}$$

Similarly at the communication stage, only one out of N_{acc} transactions holds locks. The modified number of locks held at the communication stage is therefore

$$NL_k^{(com\,2)} = \frac{r_k}{N_{acc}} \left[\sum_{i=1}^{K} \frac{a_{ki}^{(2)}}{\lambda_{ki}^{(c)}} \bar{n}_{k_i} \right] \tag{5.13a}$$

$$NL_k^{(com\,3)} = \frac{r_k}{N_{acc}} \left[\sum_{i=1}^{K} \frac{a_{ik}^{(3)}}{\lambda_{ik}^{(c)}} \bar{n}_{i_k} \right] \tag{5.13b}$$

Now we have got all the necessary equations to solve the probability of the locking conflict q_k.

$$1-q_k = \frac{NL_k^{(lock)} + NL_k^{(com\,2)} + NL_k^{(com\,3)} + NL_k^{(ex)}}{L_k}$$

Substituting equation (5.9), (5.12), (5.13a) and (5.13b) into the above equation, we immediately have

$$1-q_k = \frac{\dfrac{\lambda_{k_{K+1}}}{q_k^{r_k} N_{acc} \mu_s} \sum_{i=1}^{r_k-1} i q_k^i + \dfrac{r_k}{N_{acc}} \left[\left[\sum_{i=1}^{K} \dfrac{a_{ki}^{(2)}}{\lambda_{ki}^{(c)}} \bar{n}_{k_i} + \dfrac{a_{ik}^{(3)}}{\lambda_{ik}^{(c)}} \bar{n}_{i_k} \right] + \bar{n}_{k_{K+2}} \right]}{L_k} \qquad k=1,\ldots,K$$

Diffusion Approximation Queueing Network Solution

Applying the diffusion approximation method introduced in section 4.4 to the primary copy 2PL medel, we can obtain the following results. The relative throughput of the queueing network can be derived from its Markov chain matrix given by equations (5.7). Let e_i be the relative throughput at service centre i,

$$e_i = p_{0i} + \sum_{j=1}^{m} e_j p_{ji} \qquad (i=1,\ldots,m) \tag{5.14}$$

where

$$m = K(K+2)$$

is the total number of service centres in the open distributed database queueing network. The arrival rate of service centre i is proportional to the relative throughput.

$$\lambda_i = \lambda_0 e_i \quad (i=1, \cdots, m) \tag{5.15}$$

The utilization of the service centre i is

$$\rho_i = \frac{\lambda_0 e_i}{\mu_i}, \quad \text{if } \lambda_0 e_i < \mu_i \qquad (i=1,\ldots,m) \tag{5.16}$$

Under the condition of heavy traffic, the total number of arrivals to station i in the interval $[0,t]$ will be normally distributed with mean $\lambda_i t$ and variance

$$\sum_{j=0}^{m} [(C_j-1)p_{ji}+1]\lambda_j p_{ji} t \qquad (i=1,\ldots,m) \tag{5.17}$$

where C_j is the squared coefficient of the variation of the interarrival time at service centre j.

By constructing the diffusion approximation to the length of individual queue, where $f_i(x_i,t)$ is the density function approximating the length of the i-th queue, the stationary solution of the diffusion approximation equation is given by

$$f_i(x_i) = \begin{cases} \rho_i(e^{-\gamma_i}-1)e^{\gamma_i x_i}, & x_i \geq 1 \\ \rho_i(1-e^{\gamma_i x_i}), & 0 \leq x_i \leq 1 \end{cases} \tag{5.18}$$

where

$$\gamma_i = -2\beta_i/\alpha_i \tag{5.19}$$

The parameters of the above equations are given as,

$$\beta_i = \lambda_i - \mu_i \tag{5.20}$$

$$\alpha_i = \rho_i \mu_i K_i^2 + \sum_{j=0}^{m} [(C_j - 1)p_{ji} + 1]\lambda_j p_{ji} \tag{5.21}$$

The squared coefficient of variation of the interarrival time C_i, $(i=1, \cdots, m)$ can be obtained numerically by solving a system of linear equations

$$(C_i - 1) - \frac{(1-\rho_i)}{\lambda_i + \lambda_i p_{ii}^2 (1-\rho_i)} \sum_{j=0, j\neq i}^{m} (C_j - 1)\lambda_j p_{ji}^2 = \frac{\rho_i (K_i^2 - 1)}{1-(1-\rho_i)p_{ii}^2} \qquad (i=1,...,m) \tag{5.22}$$

where K_i is the squared coefficient of variation of the service time distribution at service centre i.

The lock request service centre j_{K+1} in figure 4.2 is a IS service centre which can be represented by a single server service centre with service capacity equal to $1/\bar{N}_{j_{K+1}}\mu_{j_{K+1}}$ $(j=1,...,K)$. From equation (4.112) the throughput of lock request service centre j_{K+1} can be given by

$$\rho_{j_{K+1}} = \hat{\lambda}_{j_{K+1}} / \bar{N}_{j_{K+1}} \mu_{j_{K+1}} \tag{5.23}$$

where $\bar{N}_{j_{K+1}}$ is given by equation (4.113) and rewritten as

$$\bar{N}_{j_{K+1}} = \frac{1}{2}\left(C_{j_{K-1}} + \frac{2\hat{\lambda}_{j_{K+1}}}{\mu_{j_{K+1}}} + 1\right) \tag{5.24}$$

$C_{j_{K+1}}$ is the squared coefficient of the variation of the interarrival time at service centre j_{K+1}, which equals to the sum of the independent variance of the interdeparture time of all the other service centres to centre j_{K+1}.

$$C_{j_{K+1}} = \frac{1}{\hat{\lambda}_{j_{K+1}}} \sum_{i=0}^{m} [\,(C_i - 1)p_{ij_{K+1}} + 1\,]\hat{\lambda}_i p_{ij_{K+1}} \tag{5.25}$$

Substituting equation (5.8), (5.24) and (5.25) into equation (5.23), we can obtain $\rho_{j_{K+1}}$. The approximate average queue length can be therefore given by

$$\bar{n}_i = \rho_i \left(1 - \frac{\alpha_i}{2\beta_i}\right)$$

Applying Little's law, the average turn-around time of the system is

$$T_a = \sum_{i=1}^{m} \frac{\bar{n}_i}{\lambda_0} \tag{5.26}$$

5.1.3 Performance Results

The parameters of the evaluation of the primary copy 2PL system are the same as basic 2PL. They are given by table 4.3 to 4.8.

The overall system performance is given in terms of mean turnaround time of the system. The effect of various parameters towards the mean response time is studied and illustrated in figure 5.3 to 5.8.

Read-Write Ratio

Figure 5.3 illustrates the system behavior affected by the read-write ratio γ_{rw}. Its parameters are given in table 4.4. It shows clearly that the system performs better under smaller γ_{rw}, i.e. more read type and less write type transactions. When the range of interarrival rate is below $0.5\ (1/s)$, the response times of different γ_{rw} are very close to one another, which means the system can perform almost equally well in lightly loaded condition without being affected by the update ratio γ_{rw}.

Data Replication

The response times under different mean number of duplicated copies are given in figure 5.4. The parameters are given in table 4.5. Its pattern is very similar to that of basic 2PL in figure 4.14. But there is an intersting phenomenon at the range $0 \leq \lambda' < 0.25$. The choice of no replication is the least favorable among others. $\lambda' \rightarrow 0.25$ is a turning point. The reason behind this is that when work load is small there is no queueing at the transaction execution and data communication centres. High replication can increase the chance of local access but not necessarily the processing time. Therefore high data replication can reduce the response time. As the work load builds up, queues will form more rapidly for higher replicated system. Thus the performance tends to favor lower replication.

Lock Granularity

Figure 5.5 shows the effect of database granularity with parameters given in table 4.6. The mean number of locks for each transaction vs. the total number of granulars is set at 5/500, 15/1500, 25/2500, 35/3500 respectively. The results show that the difference between coarse and fine granularity is even smaller than that of basic 2PL in figure 4.15, because locking is only performed on primary copies. The effect of granularity on performance is related only to the lock processing times at the sites of primary copies.

Read-Write Ratio vs. Data Replication

Figure 5.6 shows the change of response time with N_f under different γ_{rw}. The decrease in response time towards high replication is more obvious for the primary copy 2PL than that for the basic 2PL. When

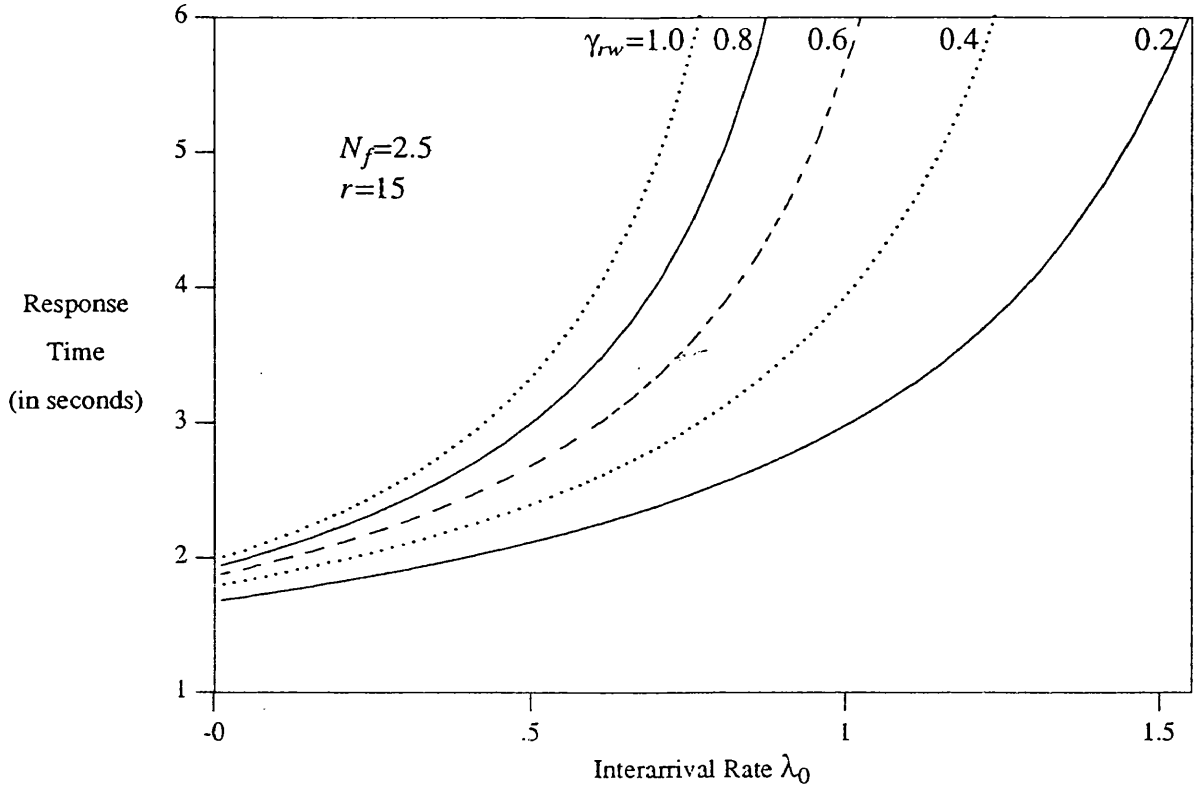


Figure 5.3. Response Times with Different Read-Write Ratio γ_{rw}

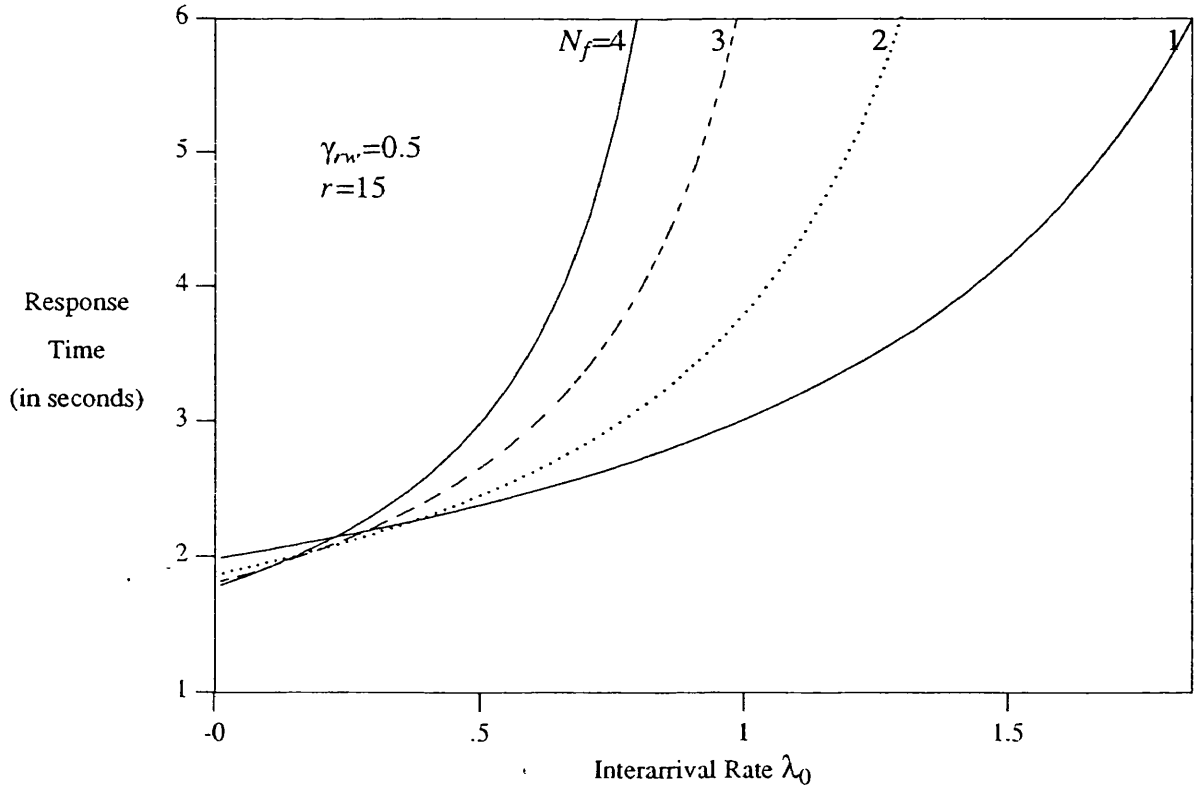


Figure 5.4. Response Time with Different Replicated Copies N_f

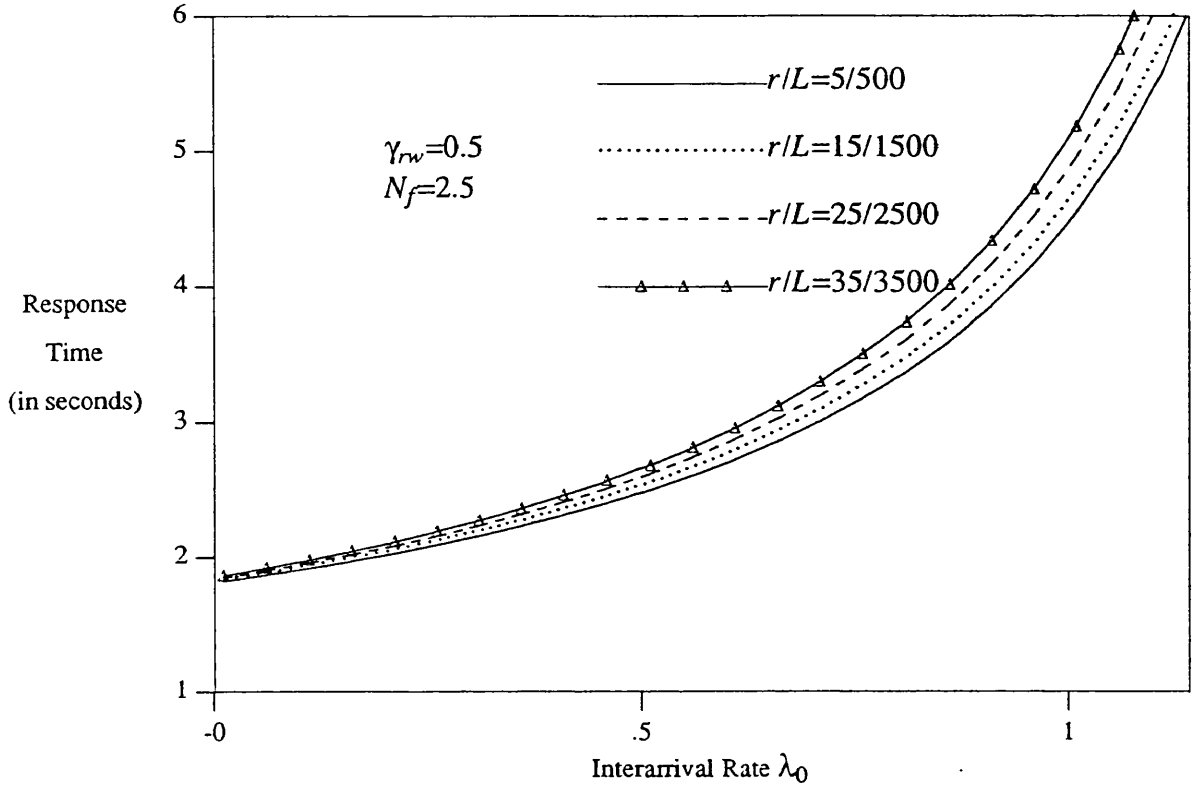


Figure 5.5. Response Times with Different Granularity r/L

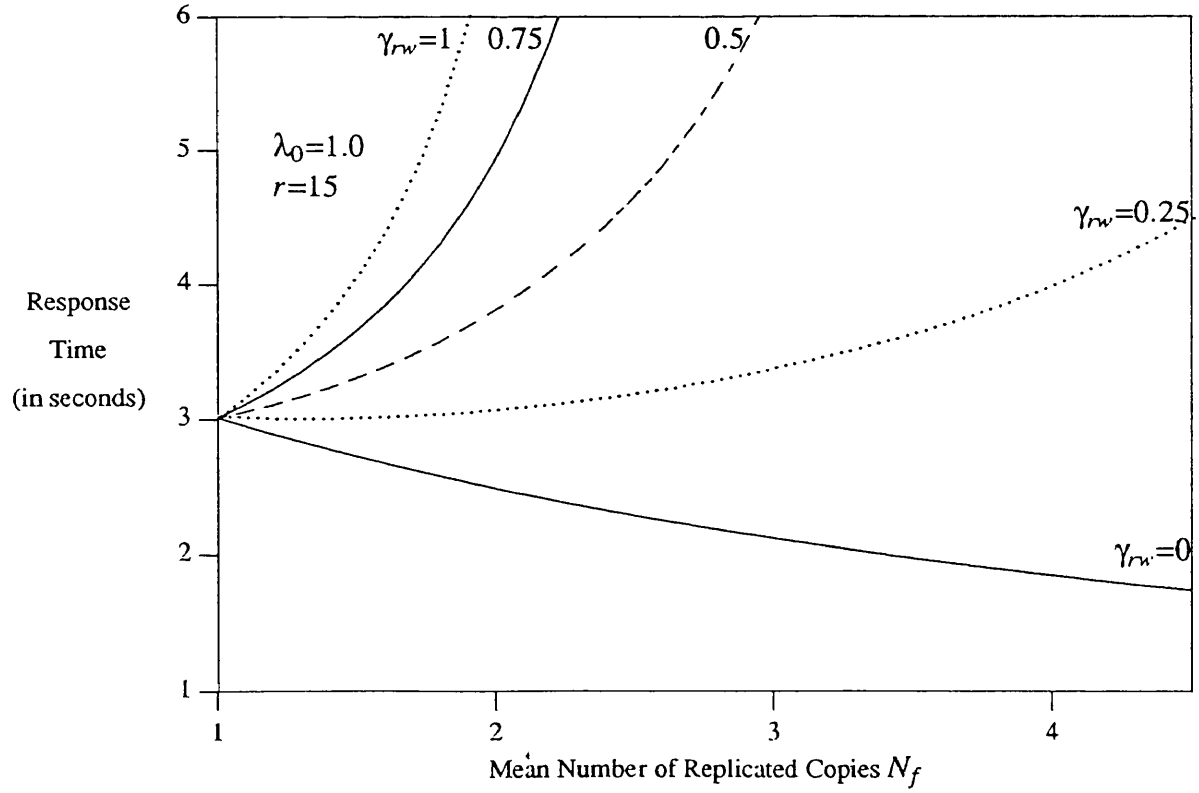


Figure 5.6. Response Times vs N_f with Different Read-Write Ratio γ_{rw}

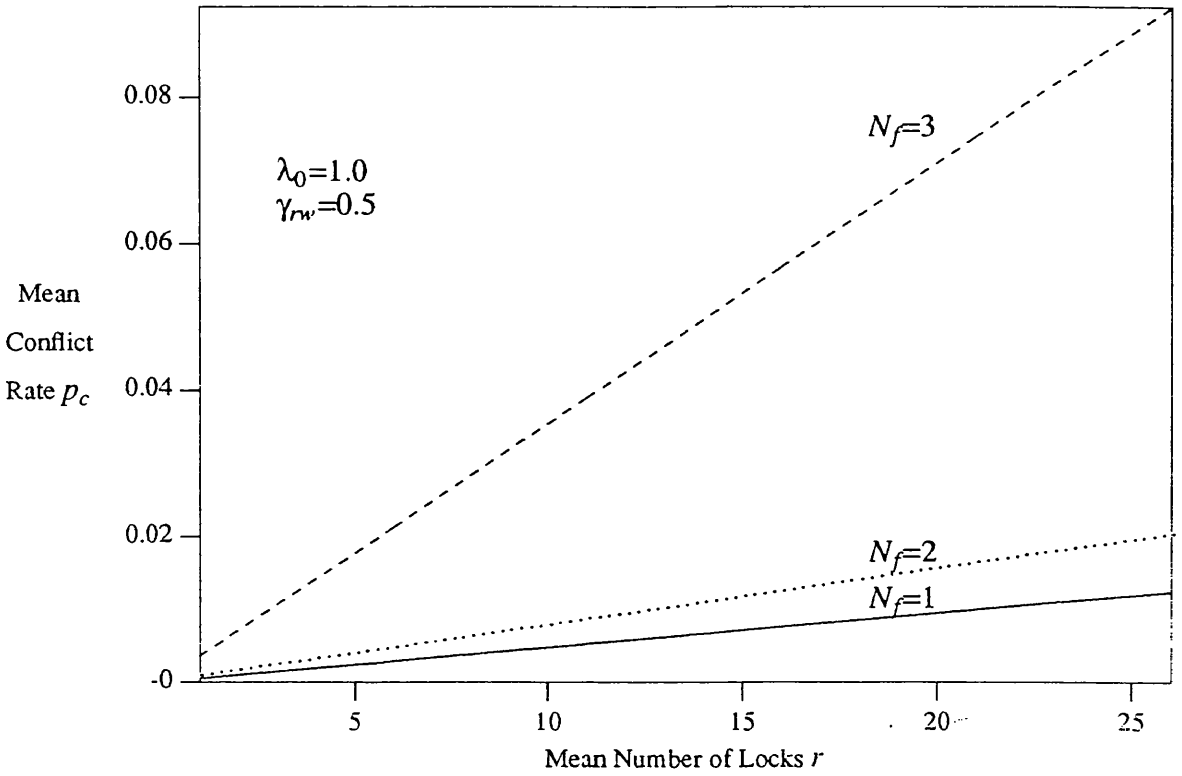


Figure 5.7. Mean Conflict Rates with Different Replicated Copies N_f

the read-write ratio γ_{rw} grows bigger the response time increases dramatically with N_f. This quantitative result can be used as a guidance for DDBMS system design. When the system is highly update oriented, the optimal mean duplicated copies should be set to one, which means no duplication at all. The parameters are given in table 4.7.

Conflict Rate

Figure 5.7 indicates how the number of duplicated copies N_f effects the conflict rate of lock request. The higher the N_f, the greater the conflict rate p_c. So N_f should be carefully chosen to obtain a good system performance. The parameters are given in table 4.8.

5.2 Majority Consensus 2PL

5.2.1 System Specification

The majority consensus approach is first introduced by Thomas[80]. The algorithm uses a majority voting rule to control the concurrent executions of transactions. The original majority consensus algorithm is based on time stamps. The algorithm is further developed into a two phase locking majority consensus algorithm. The basic idea of the majority consensus 2PL is to lock the majority of data copies before transaction execution. Since only one transaction can lock a majority of data copies at a time, no more than one transactions can succeed at the commit stage at any time. The algorithms of the majority consensus 2PL can be presented as follows:

Majority Consensus 2PL Protocol:

Phase 1:

1.1 The coordinator TM_k sends $Prepare\,(X)$ and lock request (PRE) message to all the participants $\{DM_i, X \in DM_i\}$.

1.2 Each DM_i receives the message; and checks for the requested locks. If all locks are granted and it is willing to commit, it then writes the transaction's record in the log and votes OK to TM_k; otherwise it votes ABT to TM_k.

Phase 2:

2.1 Upon receiving one reject (REJ) vote from a participant $\{DM_i\}$, TM_k sends an ABT message to all $\{DM_i, X \in DM_i\}$; Upon receiving a majority of OK votes from $\{DM_i, X \in DM_i\}$, TM_k sends commit (CMT) message to all $\{DM_i, X \in DM_i\}$; and upon receiving less than a majority of OK votes from $\{DM_i, X \in DM_i\}$ at timeout, TM_k sends ABT message to all $\{DM_i, X \in DM_i\}$.

2.2 After receiving the command message from TM_k, each DM_i writes either an abort or commit record in log; then executes the transaction and releases the locks; and sends the acknowledge (ACK) message to TM_k. If the command is ABT, it simply release all the locks held by this transaction.

2.3 TM_k waits for ACK messages from all $\{DM_i, X \in DM_i\}$; then writes a complete record in log.

The majority consensus 2PL increases the availability of a distributed database system when dealing with write oriented transaction. But in order to read a logical data item it has to lock a majority of the data copies rather than lock only one copy. This constrain is usually stronger than required for consistency. It requires extra communications and locking activities.

The $Read\,(X)$ of the majority consensus 2PL is shown in figure 5.8. A coordinator TM_k sends lock request to all the $DM_j (j=1,...,K)$. Each DM_j tries to grant the requested locks and sends either OK or REJ to DM_k. If there exists a majority of OK votes, the coordinator TM_k chooses one of the DMs to read the data item X and sends lock release to all the $DM_j (j=1,...,K)$. If a majority of OK votes can not be formed or a REJ vote is received, the coordinator TM_k sends ABT to all the DM_j $(j=1,...,K)$ and restart with probability q_{b_k} after waiting.

The concurrency control structure of $Write\,(X)$, as shown in figure 5.9, is similar to the $Read\,(X)$ except that $dm-write\,(X)$ is performed on all the data copies. The main difference between majority consensus 2PL and basic 2PL is that the majority consensus 2PL does not wait for blocked locks in local DMs but rather wait in the coordinator TM_k site. The local $DM_j (j=1,...,K)$ send their votes immediately without waiting for unavailable locks. It is up to the coordinator TM_k to decide whether to

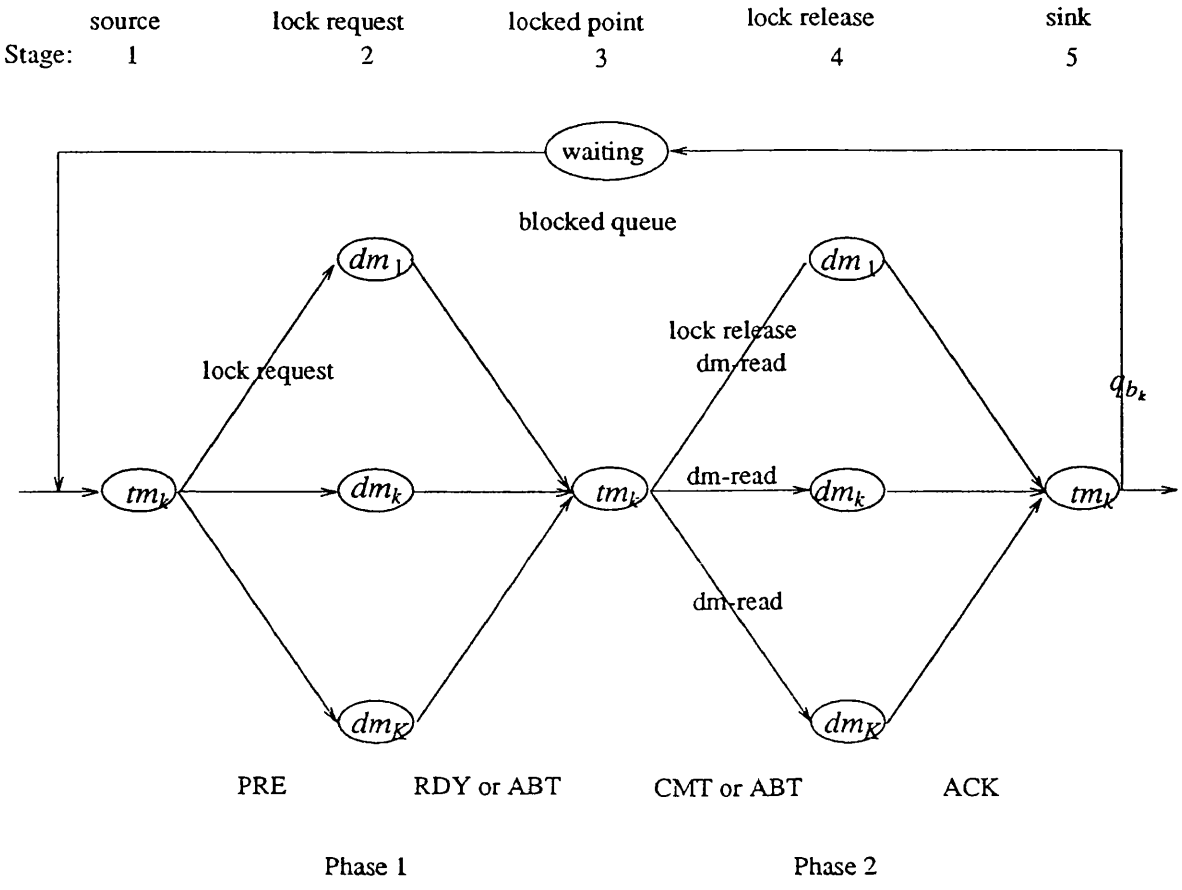


Figure 5.8. Structure of $Read\ (X)$ of Majority Consensus 2PL

commit or abort according to the result of the voting. If a majority of OK votes can not be obtained, the coordinator TM_k releases all the locks and restarts the transaction after waiting.

5.2.2 Model Definition

Access Rules

We shall give a formal definition of the read and write rules of the majority consensus 2PL as follows:

Definition 5.4: The rule of $Read\ (X)$ issued at TM_k is as follows:

- if $X \in DM_j$, $lock(X)$ and $dm-read(X)$ at DM_j.

The rule of $Write\ (X)$ is as follows

- if $X \in DM_j$, $lock(X)$ and $dm-write(X)$ at DM_j.

Access Pattern Suppose the data copies are uniformly stored in distributed database. That is

$$Prob\{X \in DM_k\} = \frac{N_f}{K}$$

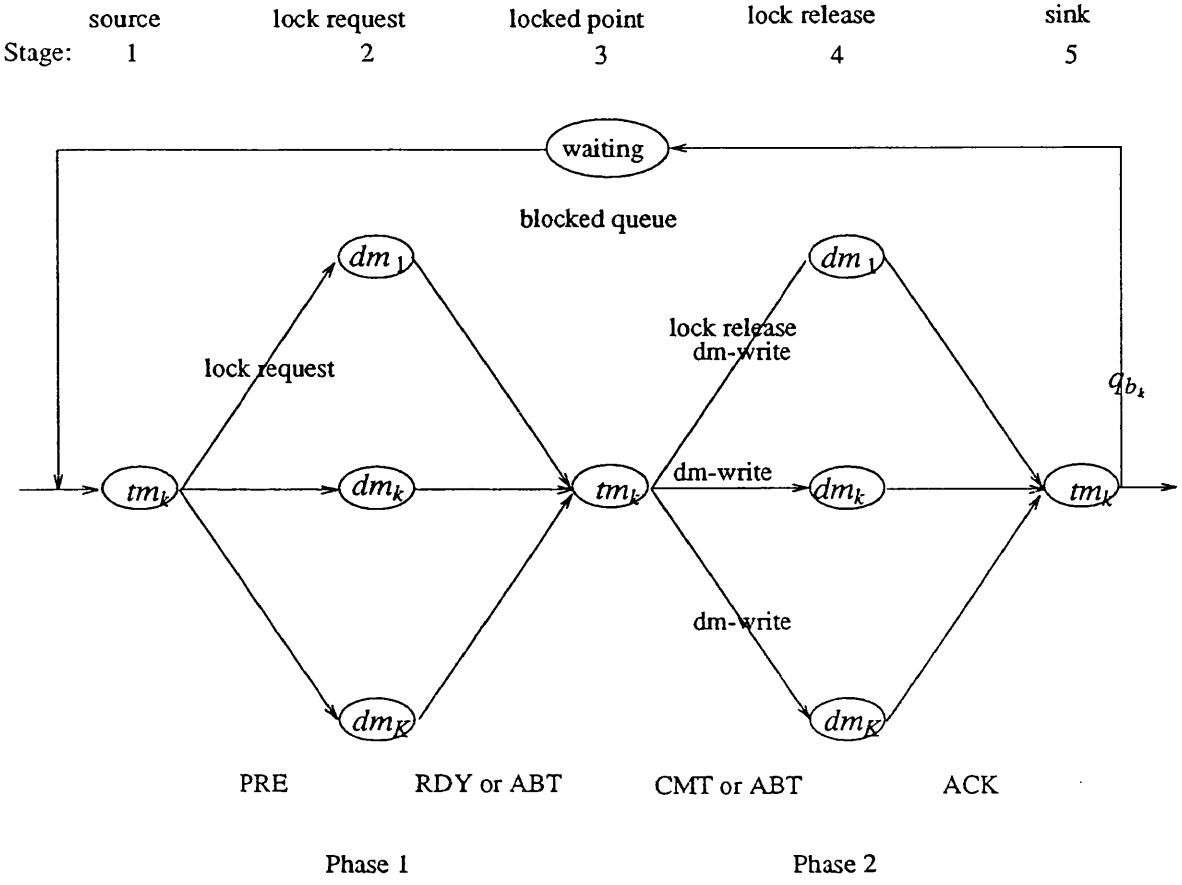


Figure 5.9. Structure of $Write\ (X)$ of Majority Consensus 2PL

The probability of a transaction at TM_k being read or write oriented are respectively

$$Prob\{read\} = 1 - \gamma_{rw_k}$$

and

$$Prob\{write\} = \gamma_{rw_k}$$

Since read and write transactions both require N_f DM accesses, the total number of DMs accessed by one transaction is simply given by

$$\begin{aligned} N_{acc} &= N_f \cdot Prob\{read\} + N_f \cdot Prob\{write\} \\ &= N_f \end{aligned} \tag{5.27}$$

Be applying the same method to the majority consensus 2PL, the access pattern matrix can also be easily obtained by the following theorem.

Theorem 5.5: The access pattern matrix $\mathbf{W}$ of majority consensus 2PL is given by

$$\omega_{kj} = \frac{1}{K} \quad k,j=1,...,K \tag{5.28}$$

Proof: ω_{kj} is defined by

$$\begin{aligned}\omega_{kj} &= Prob\{Access_{kj} \mid X \in DM_j\} Prob\{X \in DM_j\} \\ &\quad + Prob\{Access_{kj} \mid \overline{X \in DM_j}\} Prob\{\overline{X \in DM_j}\} \\ &= \frac{1}{N_{acc}} \cdot Prob\{X \in DM_j\} \\ &= \frac{1}{N_{acc}} \cdot \frac{N_f}{K} \\ &= \frac{1}{K}\end{aligned}$$

Hence [ ].

Arrival Rate Matrix

As we can see from figure 5.8 and 5.9 the blocked transactions will enter the blocked restart after a waiting delay. The probability of a transaction entering the blocked queue at TM_k is denoted by q_{b_k}. The virtual arrival rate of DM_k is therefore equal to the sum of the original arrival rate $N_{acc}\lambda'_k$ and the restart rate $q_{b_k}N_{acc}\lambda'_k$, that is

$$\begin{aligned}\lambda_k &= N_{acc}\lambda'_k + (1-q_{b_k})N_{acc}\lambda'_k \\ &= (2-q_{b_k})N_f\lambda'_k\end{aligned} \tag{5.29}$$

The arrival rate array at the source point is therefore

$$\Lambda_0 = \begin{bmatrix}\lambda_1 & \lambda_2 & \cdots & \lambda_K\end{bmatrix} \tag{5.30a}$$

The arrival rate to the blocked queue centre is given by

$$\Lambda_b = \Lambda_0 \begin{bmatrix} 1-q_{b_1} & 0 & \cdots & 0 \\ 0 & 1-q_{b_2} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1-q_{b_K} \end{bmatrix} \tag{5.30b}$$

The arrival rates at the other queueing centres are

$$\begin{aligned}\Lambda_1 &= \mathbf{V}^T\mathbf{A}_1 \\ &= \begin{bmatrix}\lambda_1\omega_{11}+\lambda_2\omega_{21}+...+\lambda_K\omega_{K1} & \lambda_1\omega_{12}+\lambda_2\omega_{22}+...+\lambda_K\omega_{K2} & \cdots & \lambda_1\omega_{1K}+\lambda_2\omega_{2K}+...+\lambda_K\omega_{KK}\end{bmatrix}\end{aligned} \tag{5.30c}$$

$$\Lambda_2 = \mathbf{V}^T \mathbf{A}_2 \tag{5.30d}$$
$$= \begin{bmatrix} \lambda_1\omega_{11}+\lambda_2\omega_{12}+...+\lambda_K\omega_{1K} & \lambda_1\omega_{21}+\lambda_2\omega_{22}+...+\lambda_K\omega_{2K} & \cdots & \lambda_1\omega_{K1}+\lambda_2\omega_{K2}+...+\lambda_K\omega_{KK} \end{bmatrix}$$

$$\Lambda_3 = \mathbf{V}^T \mathbf{A}_3 \tag{5.30e}$$
$$= \begin{bmatrix} \lambda_1\omega_{11}+\lambda_2\omega_{21}+...+\lambda_K\omega_{K1} & \lambda_1\omega_{12}+\lambda_2\omega_{22}+...+\lambda_K\omega_{K2} & \cdots & \lambda_1\omega_{1K}+\lambda_2\omega_{2K}+...+\lambda_K\omega_{KK} \end{bmatrix}$$

$$\Lambda_4 = \mathbf{V}^T \mathbf{A}_4 \tag{5.30f}$$
$$= \begin{bmatrix} \lambda_1\omega_{11}+\lambda_2\omega_{12}+...+\lambda_K\omega_{1K} & \lambda_1\omega_{21}+\lambda_2\omega_{22}+...+\lambda_K\omega_{2K} & \cdots & \lambda_1\omega_{K1}+\lambda_2\omega_{K2}+...+\lambda_K\omega_{KK} \end{bmatrix}$$

and the communication flow matrix is given by

$$\Lambda_c = \sum_{s=1}^{4} \mathbf{A}_s \tag{5.30g}$$
$$= 2\begin{bmatrix} 2\lambda_1\omega_{11} & \lambda_2\omega_{21}+\lambda_1\omega_{12} & \cdots & \lambda_K\omega_{K1}+\lambda_1\omega_{1K} \\ \lambda_1\omega_{12}+\lambda_2\omega_{21} & 2\lambda_2\omega_{22} & \cdots & \lambda_K\omega_{K2}+\lambda_2\omega_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_1\omega_{1K}+\lambda_K\omega_{K1} & \lambda_2\omega_{2K}+\lambda_K\omega_{K2} & \cdots & 2\lambda_K\omega_{KK} \end{bmatrix}$$

Markov Chain Matrix

For the majority consensus 2PL there is an additional path from the sink point to the blocked queueing centre and from the blocked queueing centre to the source point with probability $\mathbf{P}^{(sb)}$ and $\mathbf{P}^{(bs)}$ respectively. Suppose m is the total number of service centres and stage points defined in the Markov chain matrix. $\mathbf{P}^{(sb)}$ is a $(m-K)\times K$ matrix defined by

$$p_{ij}^{(sb)} = \begin{cases} q_{b_i} & \text{if } i \in (\textit{sink at } TM_k) \text{ and } j \in (\textit{blocked queue at } TM_k) \\ 0 & \text{if } \textit{otherwise} \end{cases} \tag{5.31a}$$

where $p_{ij}^{(sb)}$ can be interpreted as the probability of a transaction going from the sink point at TM_k to the blocked queue at TM_k, and $\mathbf{P}^{(bs)}$ is a $K\times(m-K)$ matrix defined by

$$p_{ij}^{(bs)} = \begin{cases} 1 & \text{if } i \in (\textit{blocked queue at } TM_k) \text{ and } j \in (\textit{source at } TM_k) \\ 0 & \text{if } \textit{otherwise} \end{cases} \tag{5.31b}$$

where $p_{ij}^{(bs)}$ is the probability of a transaction moving from the blocked queue at TM_k to the source point at TM_k. Combining $\mathbf{P}^{(sb)}$ and $\mathbf{P}^{(bs)}$ with the input and output matrices $\mathbf{P}_I$ and $\mathbf{P}_O$, the overall Markov chain matrix can be obtained by

$$\mathbf{P} = \begin{bmatrix} \begin{bmatrix} \mathbf{P}_I & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_O \end{bmatrix} & \mathbf{P}^{(sb)} \\ \mathbf{P}^{(bs)} & \mathbf{0} \end{bmatrix} \tag{5.31c}$$

Locking

The locking model of the majority consensus 2PL, as shown in figure 5.10 and 5.11, has a different communication pattern in comparison with that of basic 2PL.

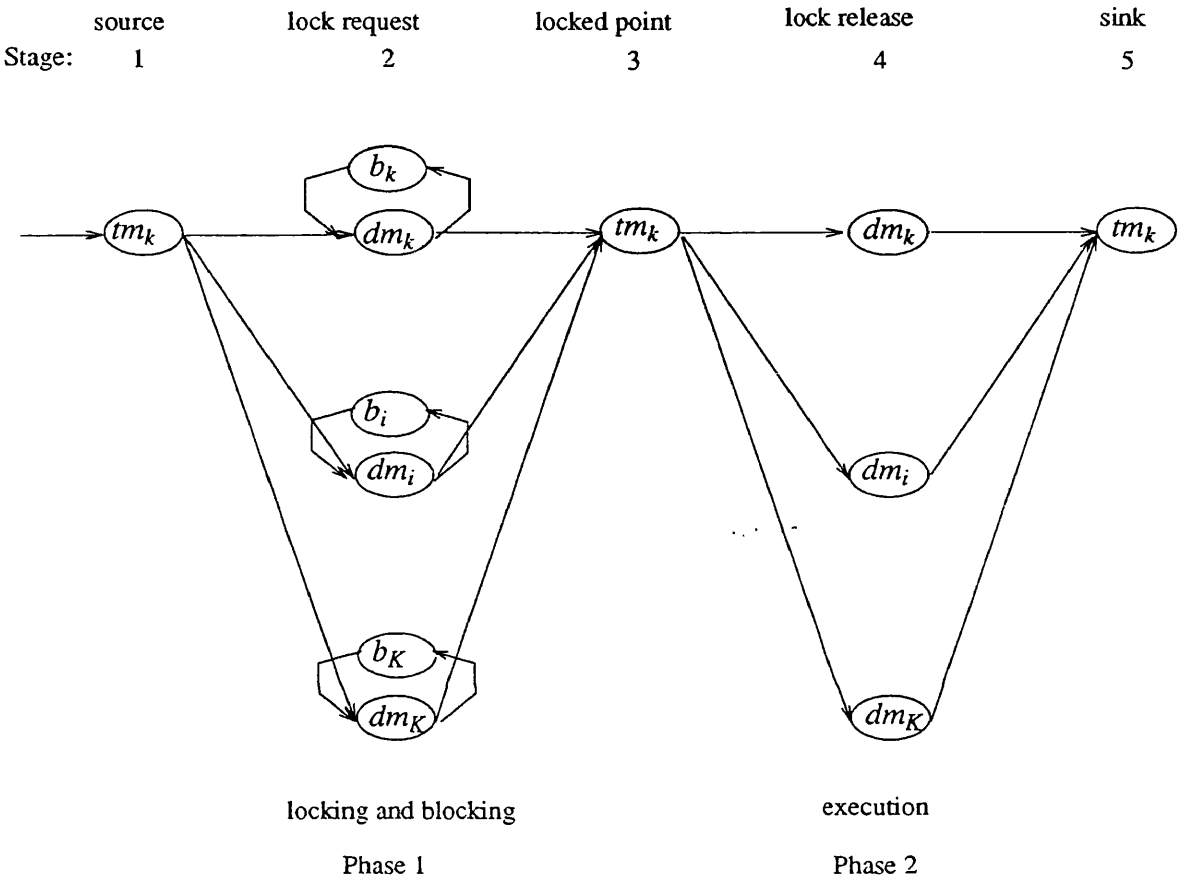


Figure 5.10. Locking Model of Basic 2PL

The blocking service centre b_k is moved from participanting DMs to the coordinator which controls the final locking decision.

We shall first study the locking behaviour of the fixed waiting model in each individual DM. As pointed out in chapter 3, we can assume that the locking overhead is much smaller than transaction execution and locking operation is always performed immediately. Under this assumption the locking behaviour at each individual DM can be modeled by using the following theorem.

Theorem 5.6: The Laplace transform of the service time of lock request and lock release at TM_k is given by

$$f^*_{lock_k}(s) = \frac{1-q_k}{q_k} \cdot \frac{1-\left[q_k\sigma^*(s)\right]^{r_k}}{1-q_k\sigma^*(s)} + \left[q_k\sigma^*(s)\right]^{r_k} \tag{5.32}$$

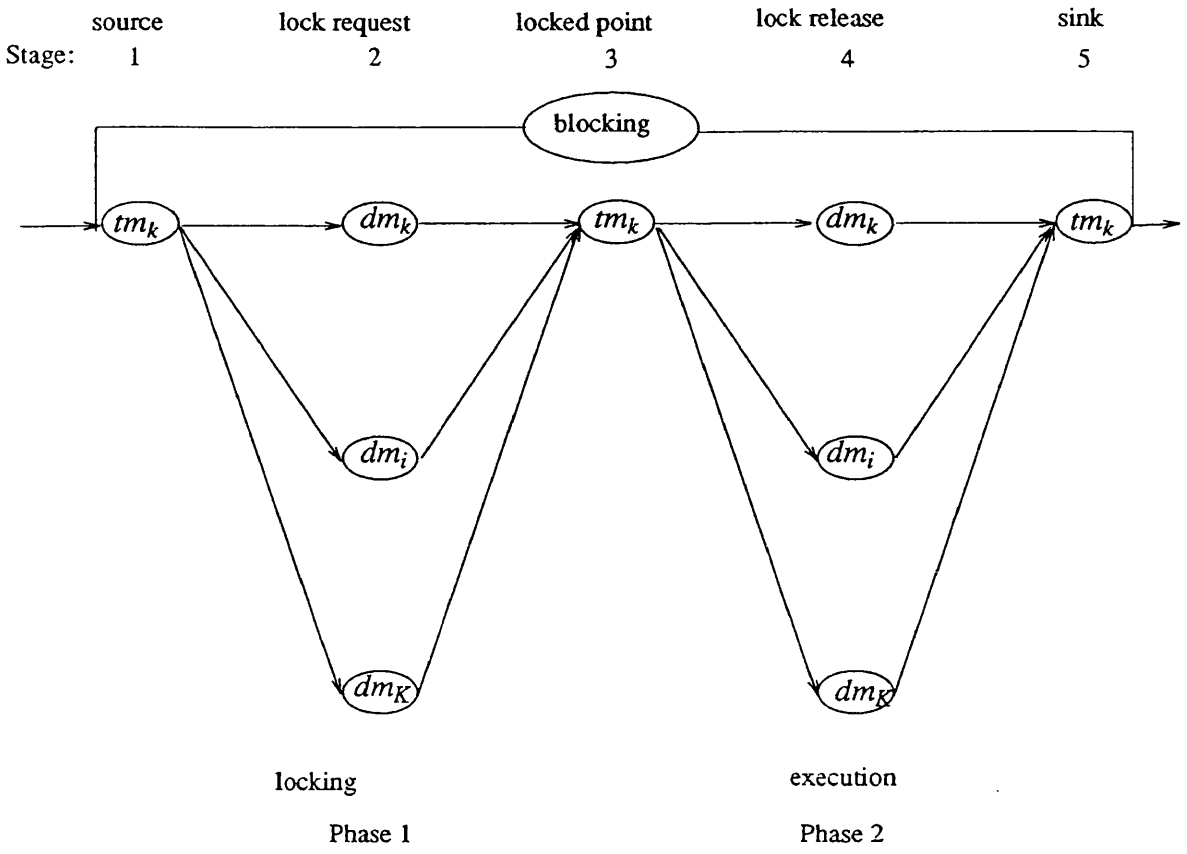


Figure 5.11. Locking Model of Majority Consensus 2PL

and the mean is given by

$$\frac{1}{\mu_{lock}} = \frac{1-q_k^{r_k}}{1-q_k} \cdot \frac{1}{\mu_s} \tag{5.33}$$

where q_k is the probability of successfully granting one lock and $\sigma^*(s)$ is the Laplace-Stieltjes transform of the service time distribution of one locking operation.

Proof: To produce a OK vote each DM must obtain r_k locks. The request of each lock can be modeled by an infinite-server service centre as introduced in chapter 3. A transaction at DM_k will successfully grant a lock with probability q_k, as illustrated in figure 5.12. It is due to Cox[17] that a system with r_k stages has the following Laplace transformation

$$f^*_{lock_k}(s) = \sum_{i=0}^{r_k-1} q_0 \cdots q_{i-1} p_i \prod_{l=1}^{i} \sigma^*(s)$$

where $p_i = 1-q_i$. Since the locking system has at lease one and at most r_k lock request stages,

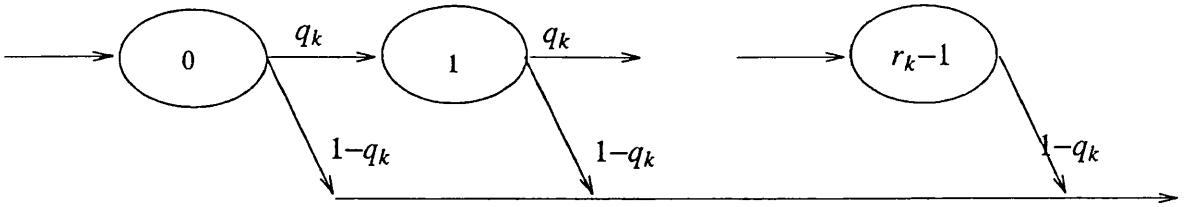


Figure 5.12. Blocking Model

$$q_0 = 1 \;;\quad p_0 = 0;$$
$$q_{r_k-1} = 0;\;;\quad p_{r_k-1} = 1$$

Furthermore the success probability q_i (i=1, $\cdots$,r_k-1) at the different stages of lock request are approximately identical under the condition of $r_k \ll L_k$ (k=1, $\cdots$,K). The success and conflict probability q_i and p_i (i=0, $\cdots$,r_k-1) are

$$q_i = q \quad i=0,...,r_k-1$$
$$p_i = 1-q \quad i=0,...,r_k-1$$

Let us denote $q_k=q$ as the success probability at DM_k. Substituting q_i and p_i into the original equation, we can further derive

$$f^*_{lock_k}(s) = \sum_{i=0}^{r_k-1} q_k^{i-1}(1-q_k)(\sigma^*(s))^i + \left[q^{r_k}\sigma^*(s)\right]^{r_k}$$
$$= \frac{1-q_k}{q_k} \cdot \frac{1-\left[q_k\sigma^*(s)\right]^{r_k}}{1-q_k\sigma^*(s)} + \left[q_k\sigma^*(s)\right]^{r_k}$$

Hence [ ].

The mean of $f^*_{lock}(s)$ is given by

$$\frac{1}{\mu_{lock}} = -\left.\frac{df^*_{lock}(s)}{ds}\right]_{s=0}$$
$$= -\frac{1-q_k}{q_k}\left[\frac{-r_k\left[q_k\sigma^*(s)\right]^{r_k-1} q_k\sigma^{*\prime}(s)\left[1-q_k\sigma^*(s)\right] - \left[-q_k\sigma^{*\prime}(s)\right]\left[1-(q_k\sigma^*(s))^{r_k}\right]}{(1-q_k\sigma^*(s))^2}\right]_{s=0}$$
$$= -r_k\left[q_k\sigma^*(s)\right]^{r_k-1} q_k\sigma^{*\prime}(s)\Big]_{s=0}$$

Since

$$\lim_{s\to 0}\sigma^*(s) = \lim_{s\to 0}\int_{-\infty}^{+\infty} e^{-st}\sigma(t)dt$$
$$= \int_{-\infty}^{+\infty}\sigma(t)dt = 1$$

and

$$-\sigma^{*'}(0) = \frac{1}{\mu_s}$$

We immediately have

$$\frac{1}{\mu_{lock}} = \frac{1-q_k^{r_k}}{1-q_k}\cdot\frac{1}{\mu_s}$$

Theorem 5.7: The probability of successfully granting one lock at DM_k is given by

$$q_k = 1 - \frac{\dfrac{\lambda_{k_{K+1}}}{q_k^{r_k}N_{acc}\mu_s}\sum_{i=1}^{r_k-1} iq_k^i + q_{b_k}r_k\left[\sum_{i=1}^{K}[\bar{n}_{k_i}^{(2)} + \bar{n}_{i_k}^{(3)}] + \bar{n}_{k_{K+2}}\right]}{L_k} \quad k=1,\ldots,K \qquad (5.34)$$

Proof: q_k is defined by

$$q_k = 1 - \frac{NL_k^{(lock)}+NL_k^{(com\,2)}+NL_k^{(com\,3)}+NL_k^{(ex)}}{L_k}$$

From equation (4.55), we have

$$NL_k^{(lock)} = \frac{\lambda_{k_{K+1}}}{q_k^{r_k}\mu_s}\sum_{i=1}^{r_k-1} iq_k^i$$

At TM_k the probability of a majority DM node voting OK is denoted by q_{b_k}. It can also be interpreted as the percentage of DM nodes which have voted OK. Since only the OK node has granted all the required locks, the number of locks held by the transaction in the communication channels is therefore given by

$$NL_k^{(com\,2)} = q_{b_k}r_k\left[\sum_{i=1}^{K}\frac{a_{ki}^{(2)}}{\lambda_{ki}^{(c)}}\bar{n}_{k_i}\right]$$

$$NL_k^{(com\,3)} = q_{b_k}r_k\left[\sum_{i=1}^{K}\frac{a_{ik}^{(3)}}{\lambda_{ik}^{(c)}}\bar{n}_{i_k}\right]$$

The number of locks held at the transaction execution centre of DM_k is similarly given by

$$NL_k^{(ex)} = q_{b_k}r_k\bar{n}_{k_{K+2}}$$

The total mean number of DM_k held is therefore equal to

$$NL_k^{(lock)} + NL_k^{(com\,2)} + NL_k^{(com\,3)} + NL_k^{(ex)} = \frac{\lambda_{k_{K+1}}}{q_k^{r_k}\mu_s} \sum_{i=1}^{r_k-1} i q_k^i + q_{b_k} r_k \left[\sum_{i=1}^{K} [\bar{n}_{k_i}^{(2)} + \bar{n}_{i_k}^{(3)}] + \bar{n}_{k_{K+2}} \right]$$

and the total number of granules at DM_k is equal to L_k. Hence

$$q_k = 1 - \frac{\frac{\lambda_{k_{K+1}}}{q_k^{r_k}\mu_s} \sum_{i=1}^{r_k-1} i q_k^i + q_{b_k} r_k \left[\sum_{i=1}^{K} \frac{a_{ki}^{(2)}}{\lambda_{ki}^{(c)}} \bar{n}_{k_i} + \frac{a_{ik}^{(3)}}{\lambda_{ik}^{(c)}} \bar{n}_{i_k} \right] + q_{b_k} r_k \bar{n}_{k_{K+2}}}{L_k}$$

[].

Next we shall derive the locking behaviour at the coordinator site TM. As mentioned previously the locking decision is made at the coordinator site TM according to the OK or REJ votes received from the participants DM_k (k=1, $\cdots$,K). If any REJ votes is received or a majority of OK can not be formed, the transaction enters the blocked queue at TM. The probability of the blocking and the duration of the blocking are derived in the following two theorems.

Theorem 5.8: The probability of non-blocking at TM_k is given by

$$q_{b_k} = 1 - \sum_{j=1}^{[\frac{N_f}{2}]} \prod_{DM_k \in OK(j)} q_k^{r_k} \prod_{\overline{DM_k \in OK(j)}} 1 - q_k^{r_k} \quad k=1,...,K \tag{5.35}$$

where $OK(j)$ denotes the set of j DMs which vote OK.

Proof: In order to obtain a majority of OK votes from $DM_i(i=1,...,,K)$, there must exist more than $N_f/2$ DM nodes which can grant all required locks. The probability of successfully granting one lock at DM_k is q_k. Thus the probability of successfully granting r_k locks at DM_k is $q_k^{r_k}$. The probability of getting j OK votes is

$$Prob\{OK\ votes=j\} = \prod_{DM_k \in OK(j)} q_k^{r_k} \prod_{\overline{DM_k \in OK(j)}} 1 - q_k^{r_k}$$

And the probability of not getting a majority of OK votes becomes

$$Prob\{OK\ votes \le \frac{N_f}{2}\} = \sum_{j=1}^{[\frac{N_f}{2}]} Prob\{OK\ votes=j\}$$

$$= \sum_{j=1}^{[\frac{N_f}{2}]} \prod_{DM_k \in OK(j)} q_k^{r_k} \prod_{\overline{DM_k \in OK(j)}} 1 - q_k^{r_k}$$

Hence

$$q_{b_k} = Prob\{OK\ votes > \frac{N_f}{2}\} = 1 - \sum_{j=1}^{[\frac{N_f}{2}]} \prod_{DM_k \in OK(j)} q_k^{r_k} \frac{\prod}{DM_k \in OK(j)} 1 - q_k^{r_k}$$

[].

Theorem 5.9: The conflict avoidance delay of the blocked transactions at TM_k can be modeled by an IS service centre with service time given by

$$\frac{1}{\mu_{b_k}} = \frac{r_k}{2\mu_s} + \frac{2}{\mu_t} + \frac{1}{\mu_e} \tag{5.36}$$

where $1/\mu_s$ is defined as the service time of requesting one lock in the lock request stage, $1/\mu_t$ is the communication service time and $1/\mu_e$ is the transaction execution service time.

Proof: When using fixed waiting conflict resolution algorithm, the conflict avoidance delay at TM_k can be defined as the sum of the remaining lock request delay, the two way communication delay and the transaction execution delay. Locks are requested in parallel at different DMs. The overall locking delay is at least r_k/μ_s. Thus the remaining lock request delay is at least $r_k/2\mu_s$ sec. If we omit queuing delay at the communication channels, the two way communication delay is $2/\mu_t$. The transaction is executed parallelly at different $DM_k(k=1,...,K)$. The transaction execution delay is therefore at least $1/\mu_e$ sec. Therefore the service time of the blocking service centre is given by

$$\frac{1}{\mu_{b_k}} = \frac{r_k}{2\mu_s} + \frac{2}{\mu_t} + \frac{1}{\mu_e}$$

Transaction Execution

Majority consensus 2PL has a unique transaction execution pattern. The success or failure of the voting determines whether or not a transaction should be executed. For each DM at execution stage, the mean number of total accesses is given by

$$\begin{aligned} N_{ex}^{(tot)} &= 1 \cdot Prob\{read\} + N_f \cdot Prob\{write\} \\ &= (1-\gamma_{rw}) + N_f\gamma_{rw} \\ &= 1+\gamma_{rw}(N_f-1) \end{aligned}$$

However only those with OK votes will enter transaction execution centre. Therefore the mean number of transaction executions generated by one transaction is

$$N_{ex} = q_b N_{ex}^{(tot)} = q_b + q_b\gamma_{rw}(N_f-1) \tag{5.37}$$

where

$$q_b = \frac{1}{K}\sum_{k=1}^{K} q_{b_k}$$

This means that only N_{ex} out of N_{acc} DM accesses go to the transaction execution centre. The actual arrival rate of the transaction execution centre is

$$\hat{\lambda}_{k_{K+2}} = \frac{N_{ex}}{N_{acc}}\lambda_{k_{K+2}} \tag{5.38}$$

Diffusion Approximation Queueing Network Solution

Applying the diffusion approximation queueing network method introduced in section 4.4 with the Markov chain matrix, the arrival rate matrix, the locking and blocking model and the probability of lock conflict derived in this section, we are able to obtain all mean measurement results of the majority consensus 2PL model.

The relative throughput at service centre i is given by

$$e_i = p_{0i} + \sum_{j=1}^{m} e_j p_{ji} \qquad (i=1,...,m) \tag{5.39}$$

where

$$m = K(K+3)$$

is the total number of service centres in the open distributed database queueing network. The arrival rate of service centre i is given by

$$\lambda_i = \lambda_0 e_i \tag{5.40}$$

The utilization of the service centre i is

$$\rho_i = \frac{\lambda_0 e_i}{\mu_i}, \quad \text{if } \lambda_0 e_i < \mu_i \qquad (i=1,...,m) \tag{5.41}$$

Under the condition of heavy traffic, the total number of arrivals to station i in the interval $[0,t]$ will be normally distributed with mean $\lambda_i t$ and variance

$$\sum_{j=0}^{m} [(C_j - 1)p_{ji} + 1]\lambda_j p_{ji} t \qquad (i=1,...,m) \tag{5.42}$$

By constructing the diffusion approximation to the length of individual queue, where $f_i(x_i,t)$ is the density function approximating the length of the i-th queue, the stationary solution of the diffusion approximation equation is given by

$$f_i(x_i) = \begin{cases} \rho_i(e^{-\gamma_i}-1)e^{\gamma_i x_i}, & x_i \geq 1 \\ \rho_i(1-e^{\gamma_i x_i}), & 0 \leq x_i \leq 1 \end{cases} \tag{5.43}$$

where

$$\gamma_i = -2\beta_i/\alpha_i \tag{5.44}$$

The parameters of the above equations are given as,

$$\beta_i = \lambda_i - \mu_i \tag{5.45}$$

$$\alpha_i = \rho_i \mu_i K_i^2 + \sum_{j=0}^{m} [(C_j-1)p_{ji}+1]\lambda_j p_{ji} \tag{5.46}$$

The squared coefficient of variation of the interarrival time C_i, $(i=1, \cdots, m)$ can be obtained numerically by solving a system of linear equations

$$(C_i-1) - \frac{(1-\rho_i)}{\lambda_i + \lambda_i p_{ii}^2(1-\rho_i)} \sum_{j=0, j\neq i}^{m} (C_j-1)\lambda_j p_{ji}^2 = \frac{\rho_i(K_i^2-1)}{1-(1-\rho_i)p_{ii}^2} \qquad (i=1,...,m) \tag{5.47}$$

where K_i is the squared coefficient of variation of the service time distribution at service centre i.

The approximate average queue length is

$$\bar{n}_i = \rho_i \left[1 - \frac{\alpha_i}{2\beta_i}\right] \tag{5.48}$$

The lock request service centre j_{K+1} in figure 4.5 is a IS service centre which can be represented by a single server service centre with service capacity equal to $1/\bar{N}_{j_{K+1}}\mu_{j_{K+1}}$ $(j=1,...,K)$. From equation (4.112), the throughput of lock request service centre j_{K+1} can be given by

$$\rho_{j_{K+1}} = \lambda_{j_{K+1}}/\bar{N}_{j_{K+1}}\mu_{j_{K+1}} \tag{5.49}$$

where $\bar{N}_{j_{K+1}}$ is given by equation (4.113) and rewritten as

$$\bar{N}_{j_{K+1}} = \frac{1}{2}\left(C_{j_{K+1}} + \frac{2\lambda_{j_{K+1}}}{\mu_{j_{K+1}}} + 1\right) \tag{5.50}$$

The mean service time of the lock request centre is given by equation (5.33).

$$\frac{1}{\mu_{j_{K+1}}} = \frac{1-q_j^{r_j}}{q_j} \cdot \frac{1}{\mu_s} \tag{5.51}$$

$C_{j_{K+1}}$ is the squared coefficient of the variation of the interarrival time at service centre j_{K+1}, which

equals to the sum of the independent variance of the interdeparture time of all the other service centres to centre j_{K+1}.

$$C_{j_{K+1}} = \frac{1}{\lambda_{j_{K+1}}} \sum_{i=0}^{m} [\,(C_i - 1)p_{ij_{K+1}} + 1\,]\lambda_i p_{ij_{K+1}} \tag{5.52}$$

The blocking centre at each TM_j is also an IS service centre which can be replaced by a single server service centre with service capacity equal to

$$\frac{1}{\bar{N}_{b_j}\mu_{b_j}}$$

where

$$\bar{N}_{b_j} = \frac{1}{2}\left[C_{b_j} + \frac{2\lambda_{b_j}}{\mu_{b_j}} + 1\right] \tag{5.53}$$

and $1/\mu_{b_j}$ given by

$$\frac{1}{\mu_{b_j}} = \frac{r_j}{2\mu_s} + \frac{2}{\mu_t} + \frac{1}{\mu_e} \tag{5.54}$$

and C_{b_j} is the squared coefficient of the variation of the interarrival time at the blocking service centre b_j at node j.

$$C_{b_j} = \frac{1}{\lambda_{b_j}} \sum_{i=0}^{m} [(C_i - 1)p_{ib_j} + 1]\lambda_i p_{ib_j} \tag{5.55}$$

The mean throughput of the lock request centre $\rho_{k_{K+1}}$, the squared coefficient C_i, and the mean queueing length $\bar{n}_{i_k}$ $(i=1,...,m;k=1,...,K)$ can be respectively obtained from equation (5.49), (5.47), (5.52), (5.55) and (5.48).

Applying Little's law, the average turn-around time of the system is

$$T_a = \sum_{i=1}^{m} \frac{\bar{n}_i}{\lambda_0} \tag{5.56}$$

5.2.3 Performance Results

The parameters of the majority consensus 2PL system are the same as those of the basic 2PL. They are given by table 4.3 to 4.8.

Various mean measurements such as mean response time, mean conflict rate and mean blocking rate have benn obtained under different system parameters. The effects of different parameters towards the mean response time are studied. The overall system performance is given in terms of mean turnaround time

of the system. Results of majority consensus 2PL are given in figure 5.13 to 5.18.

Read-Write Ratio

Figure 5.13 illustrates the system behavior affected by the read-write ratio γ_{rw}. Its parameters are set in table 4.4. It can be clearly seen that the response time increases as γ_{rw} increases. It is interesting to notice that for majority consensus 2PL the difference between the response times for various γ_{rw} starts to show at the very beginning. But as the work load builds up the response time increases more slowly than in other cases.

Data Replication

The response times under different mean number of duplicated copies are given in figure 5.14. The parameters are set in table 4.5. The comparison of different data replication shows that low replication is no longer the most favorable choice for majority consensus 2PL. When the system is lightly loaded, i.e. $\lambda' < 0.9$, the $N_f=1$ case has the greatest response time. The reason is that in the case of no replication, the majority is always one, while for other cases ,i.e. $N_f>1$ only a majority is necessary to avoid blocking.

Lock Granularity

Figure 5.15 shows the effect of database granularity with parameters being given in table 4.6. The mean number of locks for each transaction vs. the total number of granulars is set at 5/500, 15/1500, 25/2500, 35/3500 respectively. The results suggest that there is almost no difference between coarse and fine granularity. The effect of the database granularity is comparatively very small.

Read-Write Ratio vs. Data Replication

Figure 5.16 shows the change of response time with N_f under different γ_{rw}. The decrease of response time along with the increase of data replication under $\gamma_{rw}=0$ is more obvious than that in other cases. The phenomenon illustrated in the case of $\gamma_{rw}=0.25$ is quite interesting. It shows an optimal replication range between $N_f=2$ and 3. This result can be used as a good guidance for designing a distributed database system. The parameters are given in table 4.7.

Conflict Rate

Figure 5.17 indicates how the number of duplicated copies N_f effects the conflict rate of lock request. Figure 5.18 illustrates the mean blocking rate at the majority consensus point TM. It shows that the locking rate is more than linearly proportional to the mean number of locks required, because the effect of majority voting is factored in.

5.3 Centralized 2PL

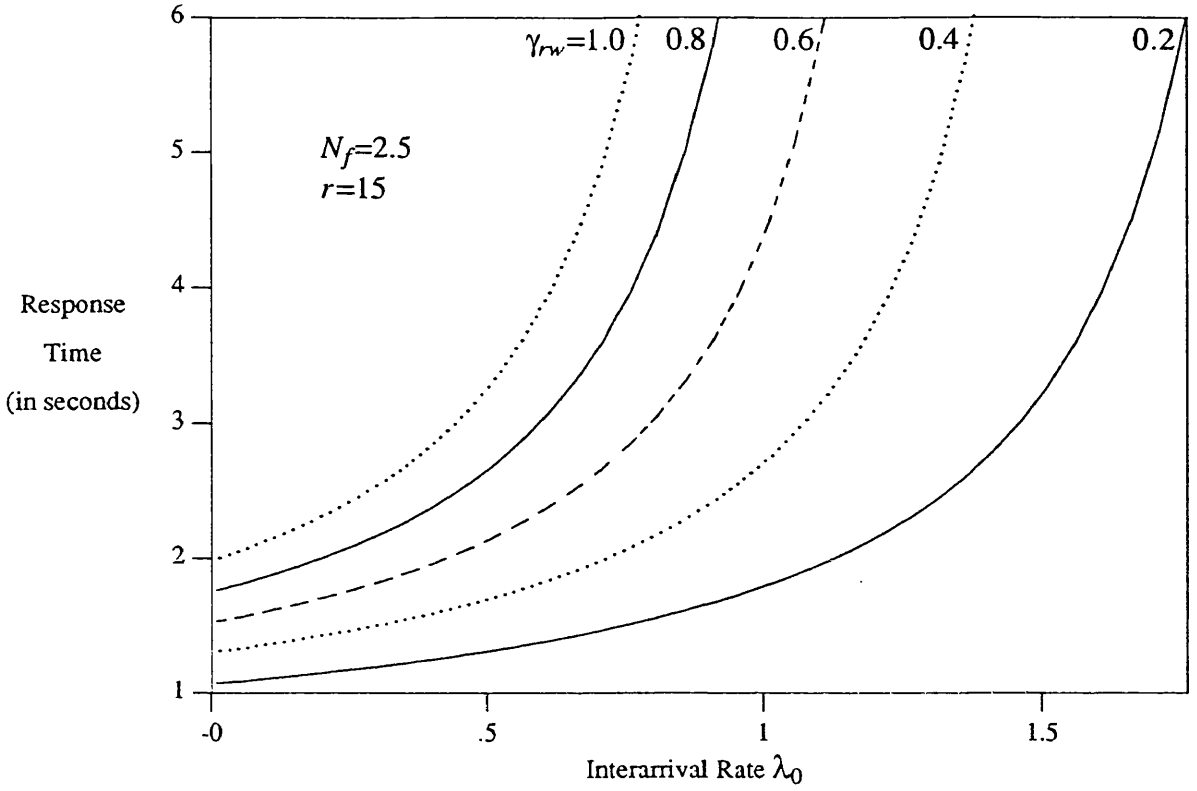


Figure 5.13. Response Times with Different Read-Write Ratio γ_{rw}

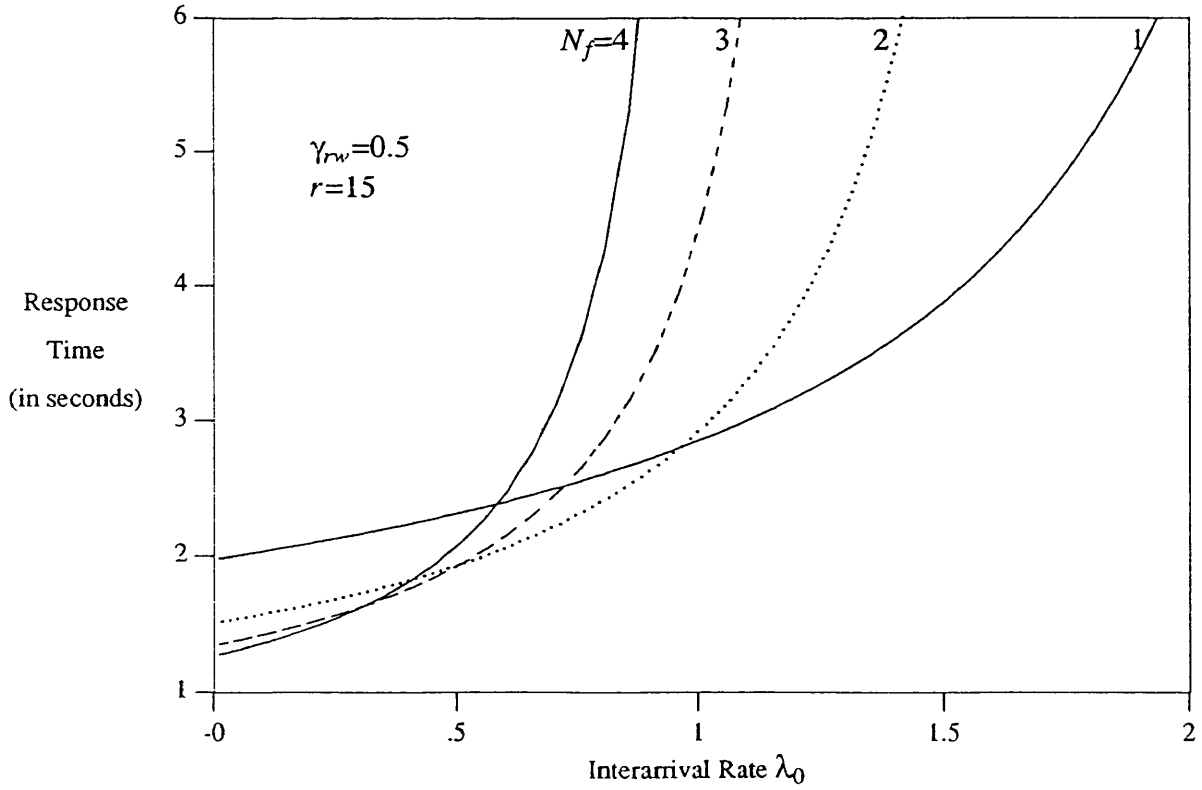


Figure 5.14. Response Time with Different Replicated Copies N_f

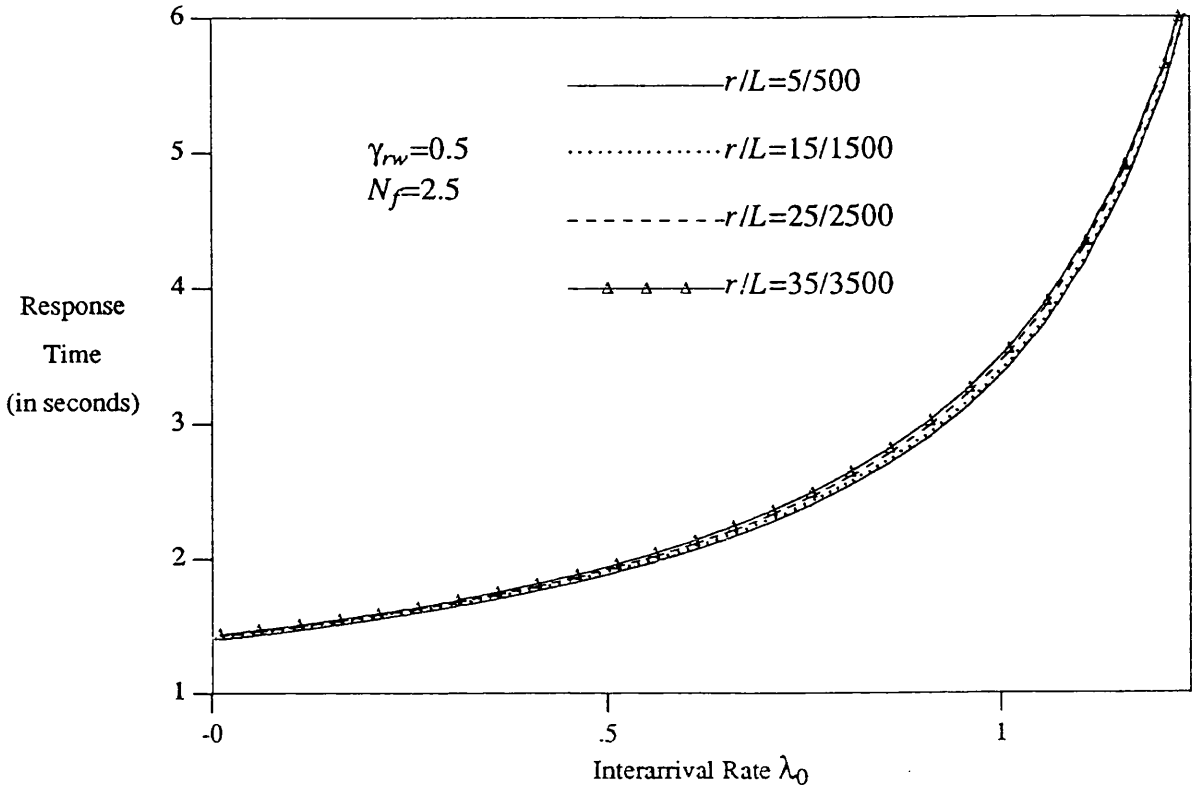


Figure 5.15. Response Times with Different Granularity r/L

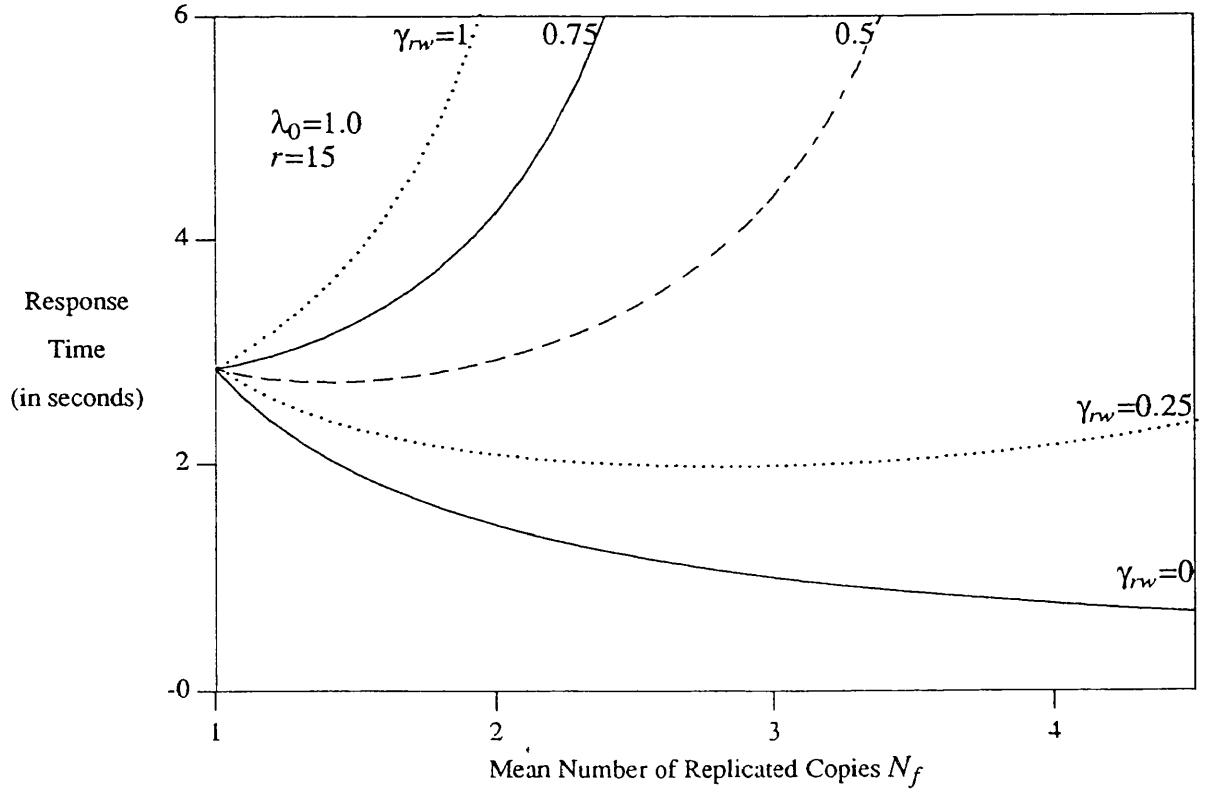


Figure 5.16. Response Times vs N_f with Different Read-Write Ratio γ_{rw}

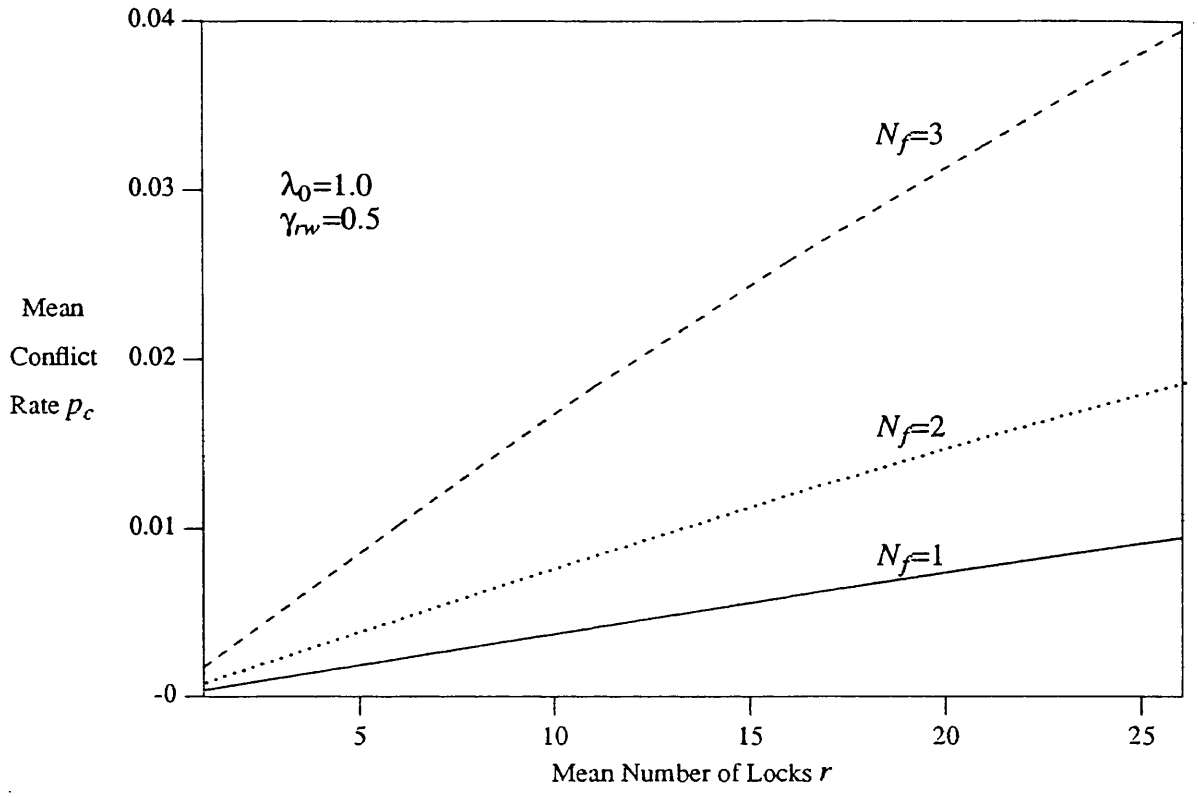


Figure 5.17. Mean Conflict Rates with Different Replicated Copies N_f

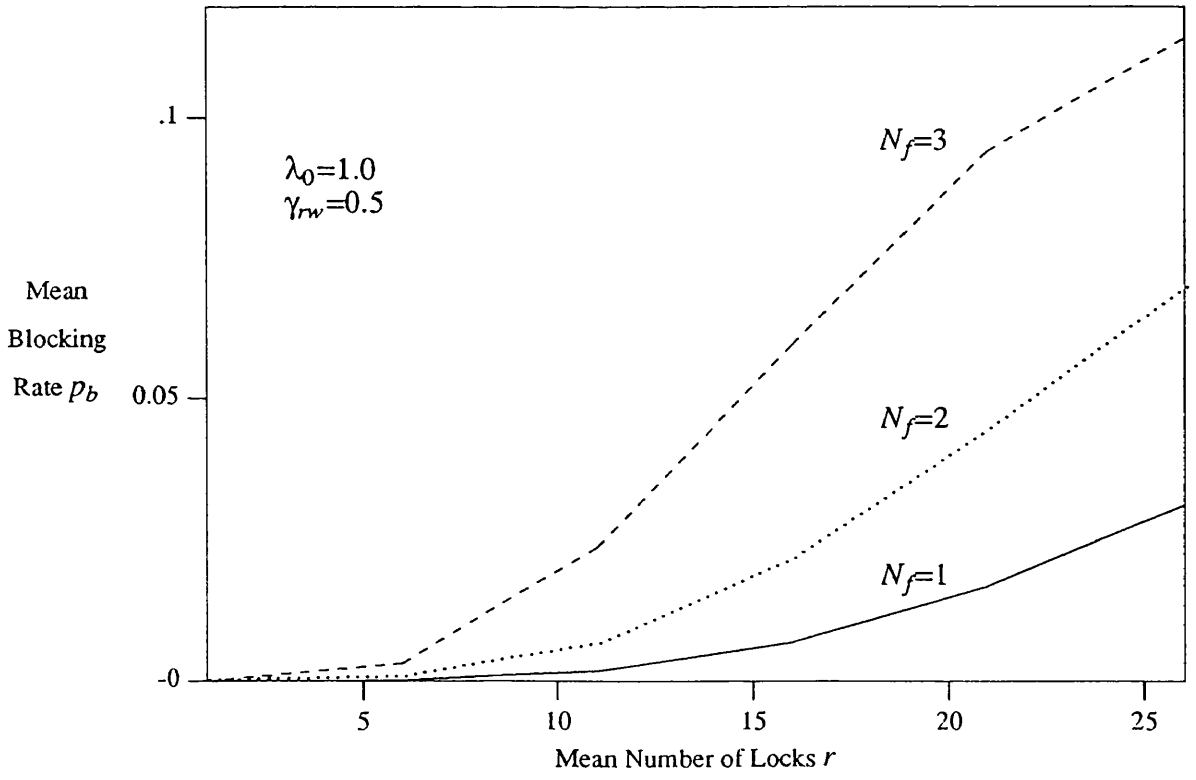


Figure 5.18. Mean Blocking Rates with Different Replicated Copies N_f

5.3.1 System Specification

In the centralized 2PL the lock scheduler is placed at a single site. A transaction must lock all the locks from this central site before accessing the actual data copies. For $Read(X)$, where X is not stored at the central site, the TM must first request locks from the central DM, wait for the central DM to grant all the locks, and then send $dm-read(x)$ to the DM which stores the data. After reading the data, the central DM must release the locks. The read-oriented control structure is illustrated in figure 5.19, where DM_1 is defined as the central site. Usually the centralized 2PL needs more communication than the basic 2PL since the lock request and lock release can not be performed explicitly by $dm-read(X)$. However if X is always stored in the centralized node the communication required by a query is the same as the basic 2PL. A $Write(X)$ requires two phases of communications, i.e., $prepare(X)$ (lock request) and $dm-write(X)$ (lock release). The transaction control structure is shown in figure 5.20. The $Prepare(X)$ and $dm-write(X)$ are similar as that of basic 2PL except that only the $prepare(X)$ at DM_1 implies lock request.

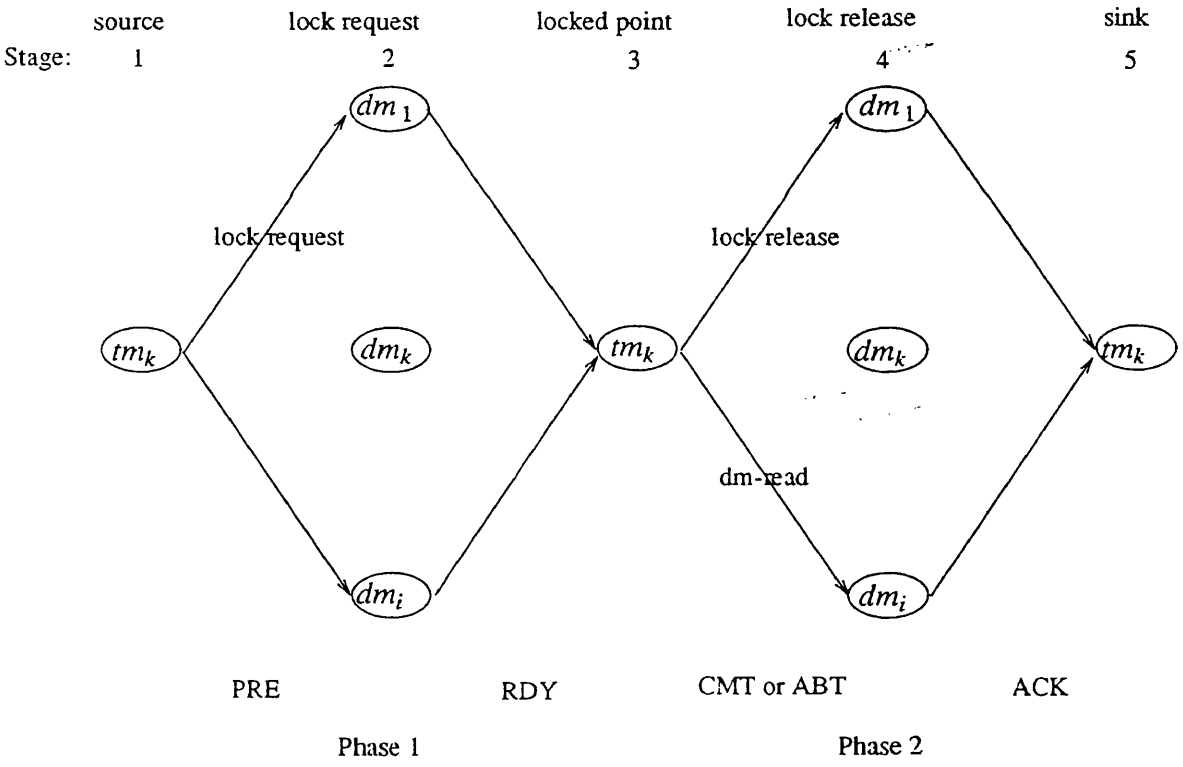


Figure 5.19. Concurrency Control Structure of $Read(X)$ of Centralized 2PL

Combining the query control with transaction control structures we can form an integrated control structure model with two phases as shown in figure 5.21, where the probability of access is defined by access pattern matrix **W**. The centralized 2PL protocol is as follows:

Centralized 2PL Protocol:

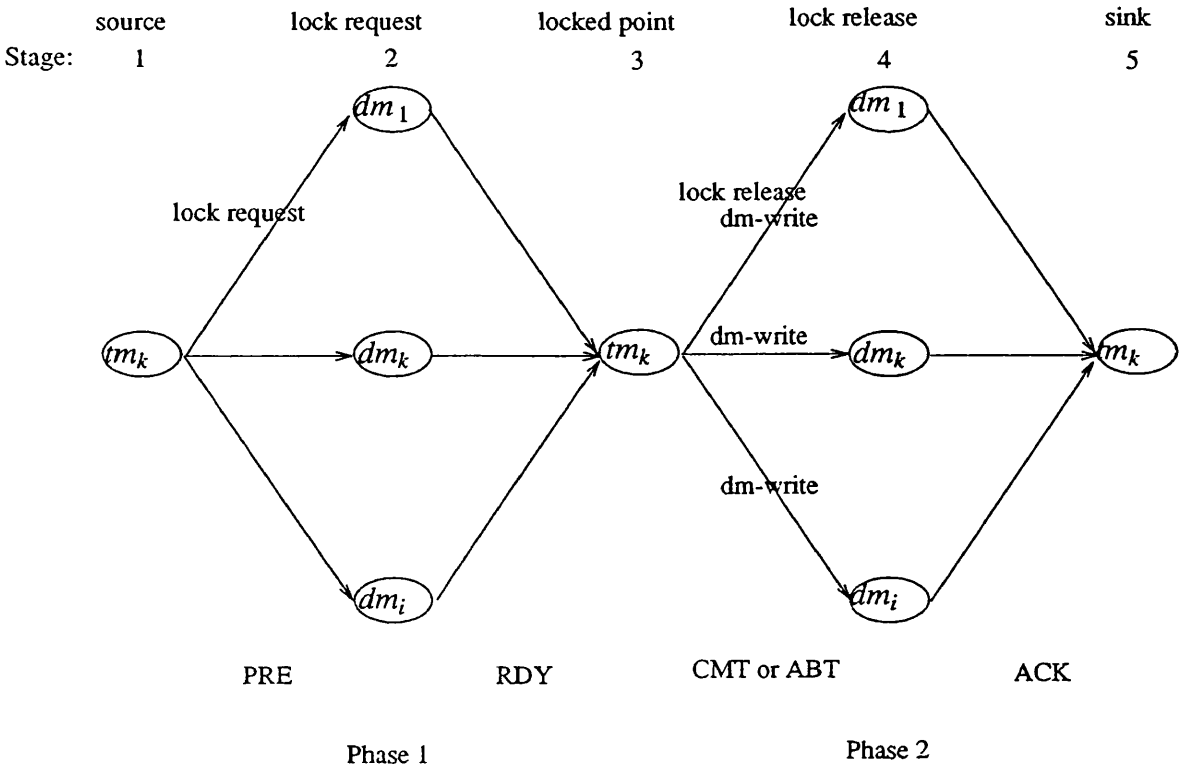


Figure 5.20. Concurrency Control Structure of $Write\,(X)$ of Centralized 2PL

Phase 1:

1.1 The coordinator TM_k sends $Prepare\,(X)$ and lock request (PRE) message to all the participants $\{DM_i,\ X \in DM_i\}$.

1.2 The central node DM_i receives the message; and checks for the required locks. If all locks are granted and it is willing to commit, then it writes the transaction's record in the log and sends ready (RDY) answer message to TM_k; otherwise it sends abort (ABT) message to TM_k. The non-central DM_i receives the message; and then writes the transaction's record in the log and sends RDY to TM_k.

Phase 2:

2.1 Upon receiving the answer message RDY or ABT from $\{DM_i,\ X \in DM_i\}$, TM_k sends commit (CMT) message to all $\{DM_i,\ X \in DM_i\}$ if all of them have voted RDY, otherwise it writes an abort record and sends ABT to all $\{DM_i,\ X \in DM_i\}$.

2.2 After receiving the command message from TM_k, each DM_i writes either an abort or commit record in log; then executes the transaction and releases the locks; and sends the acknowledge (ACK) message to TM_k.

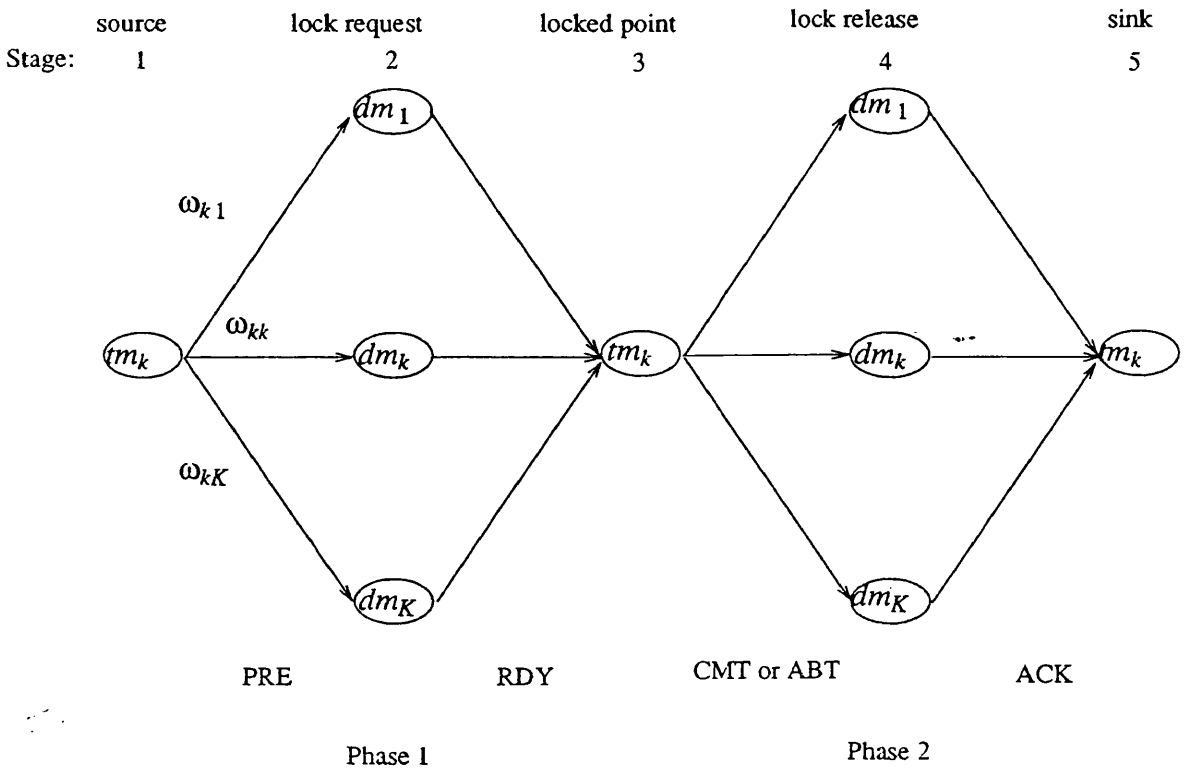


Figure 5.21. Integrated Concurrency Control Structure of Centralized 2PL

2.3 TM_k waits for ACK messages from all $\{DM_i, X \in DM_i\}$; then writes a complete record in the log.

5.3.2 Model Definition

Access Rules

We shall first formally define the read and write rules of the centralized 2PL:

Definition 5.10: Suppose the central DM is denoted by DM_1. A $Read(X)$ issued from TM_k accesses DM nodes with the following rules:

- If $X \in DM_1$, $lock(X)$ and $dm\text{-}read(X)$ at DM_1;
- If $\overline{X \in DM_1}$ and $X \in DM_k$, $lock(X)$ at DM_1 and $dm\text{-}read(X)$ at DM_k;
- If $\overline{X \in DM_1}$ and $\overline{X \in DM_k}$, $lock(X)$ at DM_1 and $dm\text{-}read(X)$ at DM_i ($i \neq 1$, and $i \neq k$).

Definition 5.11: A $Write(X)$ issued from TM_k accesses DM nodes with the following rules:

- If $X \in DM_1$, $lock(X)$ and $dm\text{-}write(X)$ at DM_1, and $dm\text{-}write(X)$ at $\{DM_i, X \in DM_i\}$;

- If $\overline{X \in DM_1}$, $lock(X)$ at DM_1, and $dm\text{-}write(X)$ at $\{DM_i, X \in DM_i\}$.

According to the read and write rules of the centralized 2PL the mean number of accesses generated by one transaction issued at TM_k can be derived by

$$\begin{aligned} N_{acc} = {} & 1 \cdot Prob\{read \cap X \in DM_1\} \\ & + 2 \cdot Prob\{read \cap \overline{X \in DM_1}\} \\ & + N_f \cdot Prob\{write \cap X \in DM_1\} \\ & + (N_f+1) Prob\{wirte \cap \overline{X \in DM_1}\} \end{aligned} \tag{5.57a}$$

Since $Prob\{read\}$ and $Prob\{write\}$ are independent of $Prob\{X \in DM_1\}$, the equation can be rewritten as

$$\begin{aligned} N_{acc} = {} & \left[1 \cdot Prob\{X \in DM_1\} + 2 \cdot Prob\{\overline{X \in DM_1}\}\right] Prob\{read\} \\ & + \left[N_f \cdot Prob\{X \in DM_1\} + (N_f+1) \cdot Prob\{\overline{X \in DM_1}\}\right] Prob\{write\} \end{aligned}$$

Suppose that the probability of a data item being stored in the central node DM_1 is given by

$$Prob\{X \in DM_1\} = p_{dm_1}$$

The read-write ratio is previously defined by

$$Prob\{read\} = 1 - \gamma_{rw_k}$$

and

$$Prob\{write\} = \gamma_{rw_k}$$

Substituting $Prob\{X \in DM_1\}$, $Prob\{read\}$ and $Prob\{write\}$ into the equation (5.57a), we have

$$\begin{aligned} N_{acc} &= \left[p_{dm_1} + 2(1-p_{dm_1})\right](1-\gamma_{rw_k}) + \left[N_f p_{dm_1} + (N_f+1)(1-p_{dm_1})\right]\gamma_{rw_k} \\ &= (2-p_{dm_1})(1-\gamma_{rw_k}) + (N_f+1-p_{dm_1})\gamma_{rw_k} \\ &= (N_f-1)\gamma_{rw_k} - p_{dm_1} + 2 \end{aligned} \tag{5.57b}$$

The probabilities of accessing DMs are defined by the access pattern matrix $\mathbf{W}$ as follows.

Theorem 5.12: The access pattern matrix W of the centralized 2PL is given by

$$\omega_{1j}=\begin{cases}\dfrac{1}{(N_f-1)\gamma_{rw_k}-p_{dm_1}+2} & \text{if } j=1\\[2ex] \dfrac{\gamma_{rw_k}p_{dm_1}(\dfrac{N_f-1}{K-1})+\gamma_{rw_k}(1-p_{dm_1})\dfrac{N_f}{K-1}+(1-\gamma_{rw_k})(1-p_{dm_1})\dfrac{1}{K-1}}{(N_f-1)\gamma_{rw_k}-p_{dm_1}+2} & \text{if } j\neq 1\end{cases} \tag{5.58a}$$

$$\omega_{kj}=\begin{cases}\dfrac{1}{(N_f-1)\gamma_{rw_k}-p_{dm_1}+2} & \text{if } j=1\\[2ex] \dfrac{\gamma_{rw_k}p_{dm_1}(\dfrac{N_f-1}{K-1})+(1-p_{dm_1})\dfrac{N_f}{K-1}}{(N_f-1)\gamma_{rw_k}-p_{dm_1}+2} & \text{if } j=k, j\neq 1\\[2ex] \dfrac{\gamma_{rw_k}p_{dm_1}\dfrac{N_f-1}{K-1}+\gamma_{rw_k}(1-p_{dm_1})\dfrac{N_f}{K-1}+(1-\gamma_{rw_k})(1-p_{dm_1})\dfrac{(K-N_f-1)}{(K-1)(K-2)}}{(N_f-1)\gamma_{rw_k}-p_{dm_1}+2} & \text{if } j\neq k, j\neq 1\end{cases}$$

$$k=2,\ldots,K;\ J=1,\ldots,K \tag{5.58b}$$

Proof: According to the read and write rules, the rate of accessing the central node locally is given by

$$\begin{aligned}\omega_{11} &= \frac{1\cdot Prob\{read\}+1\cdot Prob\{write\}}{N_{acc_k}}\\ &= \frac{1}{N_{acc_k}};\end{aligned}$$

When a transaction is issued from the central node TM_1, we have

$$\begin{aligned}\omega_{1j} &= Prob\{Access_{kj}\,|\,X\in DM_1\cap X\in DM_j\}Prob\{X\in DM_1\cap X\in DM_j\}\\ &\quad + Prob\{Access_{kj}\,|\,\overline{X\in DM_1}\cap X\in DM_j\}Prob\{\overline{X\in DM_1}\cap X\in DM_j\}\\ &= \frac{0\cdot Prob\{read\}+1\cdot Prob\{write\}}{N_{acc_k}}p_{dm_1}\left(\frac{N_f-1}{K-1}\right)\\ &\quad + \frac{1\cdot Prob\{Read(x_j)\}Prob\{read\}+1\cdot Prob\{write\}}{N_{acc_k}}(1-p_{dm_1})\frac{N_f}{K-1}\end{aligned}$$

Since there are N_f copies of a data item X and the probability of reading one of them is uniformly distributed,

$$Prob\{Read(x_j)\}=\frac{1}{N_f}$$

The equation of ω_{1j} can be rewritten as

$$\omega_{1j} = \frac{\gamma_{rw_k}}{N_{acc_k}} p_{dm_1} \frac{N_f-1}{K-1} + \frac{\gamma_{rw_k}}{N_{acc_k}} (1-p_{dm_1}) \frac{N_f}{K-1}$$
$$+ \frac{\frac{1}{N_f} \cdot (1-\gamma_{rw_k})}{N_{acc_k}} (1-p_{dm_1}) \frac{N_f}{K-1}$$
$$= \frac{\gamma_{rw_k} p_{dm_1} \frac{N_f-1}{K-1} + \gamma_{rw_k} (1-p_{dm_1}) \frac{N_f}{K-1} + (1-\gamma_{rw_k})(1-p_{dm_1}) \frac{1}{K-1}}{N_{acc}}$$

In the case of $k \neq 1$, ω_{kk} is given by

$$\omega_{\substack{kk \\ k\neq 1}} = Prob\{Access_{kk} \mid X \in DM_1 \cap X \in DM_k\} Prob\{X \in DM_1 \cap X \in DM_k\}$$
$$+ Prob\{Access_{kk} \mid X \in DM_1 \cap \overline{X \in DM_k}\} Prob\{X \in DM_1 \cap \overline{X \in DM_k}\}$$
$$+ Prob\{Access_{kk} \mid \overline{X \in DM_1} \cap X \in DM_k\} Prob\{\overline{X \in DM_1} \cap X \in DM_k\}$$
$$+ Prob\{Access_{kk} \mid \overline{X \in DM_1} \cap \overline{X \in DM_k}\} Prob\{\overline{X \in DM_1} \cap \overline{X \in DM_k}\}$$
$$= \frac{0 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}} \cdot p_{dm_1} \left[\frac{N_f-1}{K-1}\right] + 0$$
$$+ \frac{1 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}} \cdot (1-p_{dm_1}) \left[\frac{N_f}{K-1}\right] + 0$$
$$= \frac{\gamma_{rw_k}}{N_{acc_k}} \cdot p_{dm_1} \left[\frac{N_f-1}{K-1}\right] + \frac{1}{N_{acc_k}} (1-p_{dm_1}) \left[\frac{N_f}{K-1}\right]$$
$$= \frac{\gamma_{rw_k} p_{dm_1} \left[\frac{N_f-1}{K-1}\right] + (1-p_{dm_1}) \frac{N_f}{K-1}}{(N_f-1)\gamma_{rw_k} - p_{dm_1} + 2}$$

$$\omega_{\substack{kj \\ (j\neq k, j\neq 1)}} = Prob\{Access_{kj} \mid X \in DM_1 \cap X \in DM_k \cap X \in DM_j\} Prob\{X \in DM_1 \cap X \in DM_k \cap X \in DM_j\}$$
$$+ Prob\{Access_{kj} \mid X \in DM_1 \cap \overline{X \in DM_k} \cap X \in DM_j\} Prob\{X \in DM_1 \cap \overline{X \in DM_k} \cap X \in DM_j\}$$
$$+ Prob\{Access_{kj} \mid \overline{X \in DM_1} \cap X \in DM_k \cap X \in DM_j\} Prob\{\overline{X \in DM_1} \cap X \in DM_k \cap X \in DM_j\}$$
$$+ Prob\{Access_{kj} \mid \overline{X \in DM_1} \cap \overline{X \in DM_k} \cap X \in DM_j\} Prob\{\overline{X \in DM_1} \cap \overline{X \in DM_k} \cap X \in DM_j\}$$
$$= \frac{0 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}} p_{dm_1} \left[\frac{N_f-1}{K-1}\right] \left[\frac{N_f-2}{K-2}\right]$$
$$+ \frac{0 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc}} p_{dm_1} \left[1-\frac{N_f-1}{K-1}\right] \left[\frac{N_f-1}{K-2}\right]$$
$$+ \frac{0 \cdot Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc_k}} (1-p_{dm_1}) \left[\frac{N_f}{K-1}\right] \left[\frac{N_f-1}{K-2}\right]$$
$$+ \frac{Prob\{Read(x_j)\} Prob\{read\} + 1 \cdot Prob\{write\}}{N_{acc}} (1-p_{dm_1}) \left[1-\frac{N_f}{K-1}\right] \left[\frac{N_f}{K-2}\right]$$
$$= \frac{1}{N_{acc_k}} \left[\gamma_{rw_k} p_{dm_1} \left[\frac{N_f-1}{K-1}\right] \left[\frac{N_f-2}{K-2}\right] + \gamma_{rw_k} p_{dm_1} \left[\frac{K-N_f}{K-1}\right] \left[\frac{N_f-1}{K-2}\right]\right.$$

$$+\gamma_{rw_k}(1-p_{dm_1})\left[\frac{N_f}{K-1}\right]\left[\frac{N_f-1}{K-2}\right]+\left[\frac{1}{N_f}(1-\gamma_{rw_k})+\gamma_{rw_k}\right](1-p_{dm_1})\left[1-\frac{N_f}{K-1}\right]\left[\frac{N_f}{K-2}\right]$$

and ω_{kj}, $j\neq k, j\neq 1$ given by

$$=\frac{1}{N_{acc_k}}\left[\gamma_{rw_k}p_{dm_1}\frac{N_f-1}{K-1}+\gamma_{rw_k}(1-p_{dm_1})\frac{N_f}{K-1}+(1-\gamma_{rw_k})(1-p_{dm_1})\frac{(K-N_f-1)}{(K-1)(K-2)}\right]$$

Hence [ ].

If we assume that the central node stores all the data items, i.e. $p_{dm_1}=1$, the above equation can be simplified as

$$\omega_{kj}=\begin{cases}\dfrac{1}{(N_f-1)\gamma_{rw_k}+1} & \text{if } j=1\\ \dfrac{\gamma_{rw_k}p_{dm_1}\left[\dfrac{N_f-1}{K-1}\right]}{(N_f-1)\gamma_{rw_k}+1} & \text{if } j\neq 1\end{cases} \qquad k=1,\ldots,K \tag{5.59}$$

Arrival Rate Matrix

The queueing network model of the centralized 2PL can be built in the same way as that of the basic 2PL introduced in chapter 4. The overall structure of the queueing network model is shown in figure 4.2. Applying the method introduced in chapter 4, the arrival rate matrix and Markov chain matrix of the queueing network are given by

$$\Lambda_0=\left[\lambda_1^{(0)}\ \ \lambda_2^{(0)}\ \ \cdots\lambda_K^{(0)}\right] \tag{5.60a}$$

$$\Lambda_1=\mathbf{V}^T\mathbf{A}_1 \tag{5.60b}$$
$$=\left[\lambda_1\omega_{11}+\lambda_2\omega_{21}+\ldots+\lambda_K\omega_{K1}\ \ \lambda_1\omega_{12}+\lambda_2\omega_{22}+\ldots+\lambda_K\omega_{K2}\ \ \cdots\lambda_1\omega_{1K}+\lambda_2\omega_{2K}+\ldots+\lambda_K\omega_{KK}\right]$$

$$\Lambda_2=\mathbf{V}^T\mathbf{A}_2 \tag{5.60c}$$
$$=\left[\lambda_1\omega_{11}+\lambda_2\omega_{12}+\ldots+\lambda_K\omega_{1K}\ \ \lambda_1\omega_{21}+\lambda_2\omega_{22}+\ldots+\lambda_K\omega_{2K}\ \ \cdots\lambda_1\omega_{K1}+\lambda_2\omega_{K2}+\ldots+\lambda_K\omega_{KK}\right]$$

$$\Lambda_3=\mathbf{V}^T\mathbf{A}_3 \tag{5.60d}$$
$$=\left[\lambda_1\omega_{11}+\lambda_2\omega_{21}+\ldots+\lambda_K\omega_{K1}\ \ \lambda_1\omega_{12}+\lambda_2\omega_{22}+\ldots+\lambda_K\omega_{K2}\ \ \cdots\lambda_1\omega_{1K}+\lambda_2\omega_{2K}+\ldots+\lambda_K\omega_{KK}\right]$$

$$\Lambda_4=\mathbf{V}^T\mathbf{A}_4 \tag{5.60e}$$
$$=\left[\lambda_1\omega_{11}+\lambda_2\omega_{12}+\ldots+\lambda_K\omega_{1K}\ \ \lambda_1\omega_{21}+\lambda_2\omega_{22}+\ldots+\lambda_K\omega_{2K}\ \ \cdots\lambda_1\omega_{K1}+\lambda_2\omega_{K2}+\ldots+\lambda_K\omega_{KK}\right]$$

and the communication flow rate matrix is given by

$$\Lambda_c = \sum_{s=1}^{4} \Lambda_s \tag{5.60f}$$

$$= 2\begin{bmatrix} 2\lambda_1\omega_{11} & \lambda_2\omega_{21}+\lambda_1\omega_{12} & \cdots & \lambda_K\omega_{K1}+\lambda_1\omega_{1K} \\ \lambda_1\omega_{12}+\lambda_2\omega_{21} & 2\lambda_2\omega_{22} & \cdots & \lambda_K\omega_{K2}+\lambda_2\omega_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_1\omega_{1K}+\lambda_K\omega_{K1} & \lambda_2\omega_{2K}+\lambda_K\omega_{K2} & \cdots & 2\lambda_K\omega_{KK} \end{bmatrix}$$

and the Markov chain matrix are given by

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_I & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_O \end{bmatrix} \tag{5.61}$$

Locking

Locking in the centralized 2PL system is performed at the central node DM_1. A transaction first requests locks at DM_1 in the lock request service centre as shown in figure 5.22. A lock request has a probability $1-q_1$ of causing conflict and probability q_1 of successfully granting a lock. Upon lock conflict it goes to the blocked queue waiting to be rescheduled; upon granting all the locks, it leaves the lock request service centre.

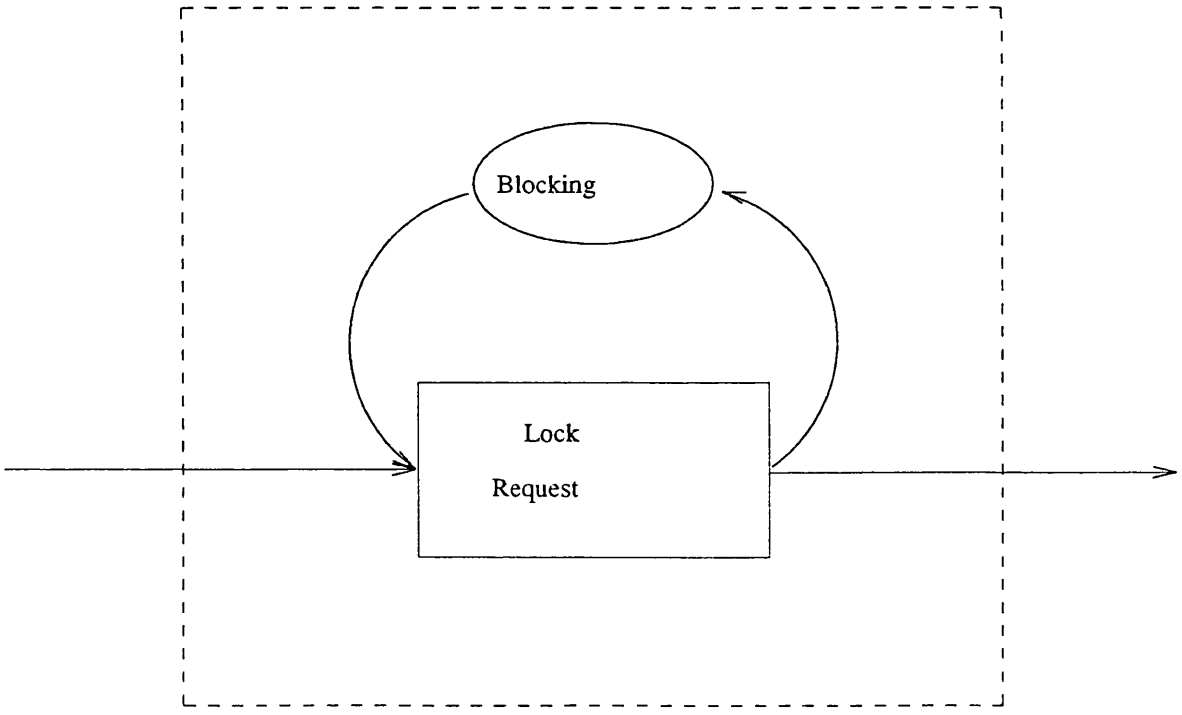


Figure 5.22. Lock Request Service Center at DM_1

We shall still use the method introduced in chapter 4 to calculated the probability of conflict $1-q_1$ in the case of fixed waiting conflict resolution .The conflict-avoidance delay at the central model DM_1 is defined as the sum for the remaining lock request delay $r/2\mu_s$, two way communication delay $2/\mu_t$ and the transaction execution delays μ_e,

$$\frac{1}{\mu_b} = \frac{r}{2\mu_s} + \frac{2}{\mu_t} + \frac{1}{\mu_e} \tag{5.62}$$

where $1/\mu_s$ is the mean service time to obtain one lock, μ_t is the mean service time of transmission delay and μ_e is the mean service time of transaction execution.

The probability of conflicts is defined as

$$\begin{aligned} 1-q_1 &= \frac{\textit{mean number of locks held at node } DM_1}{\textit{number of granules}} \\ &= \frac{NL_1^{(lock)} + NL_1^{(com\,2)} + NL_1^{(com\,3)} + NL_1^{(ex)}}{L} \end{aligned} \tag{5.63}$$

The mean number of locks held at the central node is equal to the sum of the number of locks held by each transaction multipled by the mean number of transactions in the service centres. The locks held in lock request stage at DM_1 are $NL_1^{(lock)}$. From equation (4.55), the total number of locks held at the lock request centre of DM_1 is given by

$$NL_1^{(lock)} = \frac{\lambda_{1_{K+1}}}{q_1^{r_1}\mu_s} \sum_{i=1}^{r_1-1} i q_1^i \tag{5.64}$$

At the communication channels each of the transactions leaving the lock request centre of DM_1 holds r_1 locks. The mean number of transactions holding locks at the 2nd and 3rd stage of communication is given by

$$NL_1^{(com\,2)} + NL_1^{(com\,3)} = r_1 \sum_{i=2}^{K} \left[\frac{a_{1i}^{(2)}}{\lambda_{1i}^{(c)}} \bar{n}_{1_i} + \frac{a_{i1}^{(3)}}{\lambda_{i1}^{(c)}} \bar{n}_{i_1} \right] \tag{5.65}$$

At the execution service centre of DM_1, denoted by 1_{k+2}, the mean number of transactions is given by $\bar{n}_{1_{K+2}}$. Therefore

$$NL_1^{(ex)} = r_1 \bar{n}_{1_{K+2}} \tag{5.66}$$

Substituting (5.64), (5.65) and (5.66) into equation (5.63), we have

$$1-q_1 = \frac{\dfrac{\lambda_{1_{K+1}}}{q_1^{r_1}\mu_s} \sum_{i=1}^{r_1-1} i q^i + r_1 \sum_{i=2}^{K} \left[\dfrac{a_{1i}^{(2)}}{\lambda_{1i}^{(c)}} \bar{n}_{1_i} + \dfrac{a_{i1}^{(3)}}{\lambda_{i1}^{(c)}} \bar{n}_{i_1} \right] + r_1 \bar{n}_{1_{K+2}}}{L} \tag{5.67}$$

Transaction Execution

For the centralized 2PL only the central node requires specific modification. A transaction will perform $dm-read(X)$ or $dm-write(X)$ at the central node if the data item X is stored in DM_1; otherwise a transaction only performs locking at the central node. Since $Prob\{X \in DM_1\} = p_{dm_1}$, the actual arrival rate of the transaction execution centre at DM_1 is reduced to

$$\hat{\lambda}_{1_{K+2}} = \frac{N_{ex_1}}{N_{acc}} \lambda_{1_{K+2}} = p_{dm_1} \lambda_{1_{K+2}} \tag{5.68}$$

Diffusion Approximation Queueing Network

Applying the diffusion approximation method introduced in section 4.4, various mean measurements can be obtained. Therefore q_1 can be solved numerically. The relative throughput at service centre i is given by

$$e_i = p_{0i} + \sum_{j=1}^{m} e_j p_{ji} \qquad (i=1,...,m) \tag{5.69}$$

where

$$m = K(K+2)$$

The arrival rate of service centre i is proportional to the relative throughput.

$$\lambda_i = \lambda_0 e_i \tag{5.70}$$

The utilization of the service centre i is

$$\rho_i = \frac{\lambda_0 e_i}{\mu_i}, \quad \text{if } \lambda_0 e_i < \mu_i \qquad (i=1,...,m) \tag{5.71}$$

Under the condition of heavy traffic, the total number of arrivals to station i in the interval $[0,t]$ will be normally distributed with mean $\lambda_i t$ and variance

$$\sum_{j=0}^{m} [(C_j - 1)p_{ji} + 1]\lambda_j p_{ji} t \qquad (i=1,...,m) \tag{5.72}$$

By constructing the diffusion approximation to the length of individual queue, where $f_i(x_i,t)$ is the density function approximating the length of the i-th queue, the stationary solution of the diffusion approximation equation is given by

$$f_i(x_i)=\begin{cases}\rho_i(e^{-\gamma_i}-1)e^{\gamma_i x_i}, & x_i\geq 1\\ \rho_i(1-e^{\gamma_i x_i}), & 0\leq x_i\leq 1\end{cases}$$

where

$$\gamma_i = -2\beta_i/\alpha_i \tag{5.73}$$

The parameters of the above equations are given as,

$$\beta_i = \lambda_i - \mu_i \tag{5.74}$$

$$\alpha_i = \rho_i\mu_i K_i^2 + \sum_{j=0}^{m}[(C_j-1)p_{ji}+1]\lambda_j p_{ji} \tag{5.75}$$

The squared coefficient of variation of the interarrival time C_i, $(i=1,\cdots,m)$ can be obtained numerically by solving a system of linear equations

$$(C_i-1) - \frac{(1-\rho_i)}{\lambda_i+\lambda_i p_{ii}^2(1-\rho_i)}\sum_{j=0,j\neq i}^{m}(C_j-1)\lambda_j p_{ji}^2 = \frac{\rho_i(K_i^2-1)}{1-(1-\rho_i)p_{ii}^2} \qquad (i=1,...,m) \tag{5.76}$$

where K_i is the squared coefficient of variation of the service time distribution at service centre i.

The approximate average queue length is

$$\bar{n}_i = \rho_i\left[1-\frac{\alpha_i}{2\beta_i}\right] \tag{5.77}$$

The lock request service centre at the central node 1_{K+1} in figure 4.5 is an IS service centre which can be represented by a single server service centre with service capacity equal to $1/\bar{N}_{1_{K+1}}\mu_{1_{K+1}}$. The throughput of lock request service centre 1_{K+1} is defined as

$$\rho_{1_{K+1}} = \lambda_{1_{K+1}}/\bar{N}_{1_{K+1}}\mu_{1_{K+1}} \tag{5.78}$$

where $\bar{N}_{1_{K+1}}$ is given by equation (4.113) and rewritten as

$$\bar{N}_{1_{K+1}} = \frac{1}{2}\left(C_{1_{K+1}} + \frac{2\lambda_{1_{K+1}}}{\mu_{1_{K+1}}} + 1\right) \tag{5.79}$$

$C_{1_{K+1}}$ is the squared coefficient of the variation of the interarrival time at service centre 1_{K+1}, which equals to the sum of the independent variance of the interdeparture time of all the other service centres to centre 1_{K+1}.

$$C_{1_{K+1}} = \frac{1}{\lambda_{1_{K+1}}} \sum_{i=0}^{m} [\,(C_i - 1) p_{i\,1_{K+1}} + 1\,] \lambda_i p_{i\,1_{K+1}} \tag{5.80}$$

The mean throughput of the lock request centre at the central node $\rho_{1_{K+1}}$, the squared coefficient C_i, and the mean queueing length $\bar{n}_{i_k}$ $(i=1,...,m;k=1,...,K)$ can be respectively obtained from equation (5.78), (5.76) and (5.77).

Applying Little's law, the average turn-around time of the system is

$$T_a = \sum_{i=1}^{m} \frac{\bar{n}_i}{\lambda_0} \tag{5.81}$$

5.3.3 Performance Results

The parameters of the evaluation of the centralized 2PL system are the same as basic 2PL. They are given in tables 4.3 to 4.8.

The overall system performance is given in terms of mean turnaround time of the system. The effect of different parameters towards the mean response time is studied. Results of centralized 2PL are given in figures 5.23 to 5.27.

Read-Write Ratio

Figure 5.23 illustrates the system behavior affected by the read-write ratio γ_{rw}. Its parameters are given in table 4.4. It can be clearly seen that under $\gamma_{rw}=0.2$ the response time increases much more slowly than that in other cases.

Data Replication

The response times under different mean number of duplicated copies are given in figure 5.24. The parameters are given in table 4.5. The response time of the system increases as N_f increases. The results show the same pattern as that in other cases except majority consensus. We can also find out that the system with no replication performs much better than others.

Lock Granularity

Figure 5.25 shows the effect of database granularity with parameters given in table 4.6. The mean number of locks for each transaction vs. the total number of granulars is set at 5/500, 15/1500, 25/2500, 35/3500 respectively. The results suggest that the difference between coarse and fine granularity is the smallest among all the 2PL algorithms, because the locking is performed only at the central node.

Read-Write Ratio vs. Data Replication

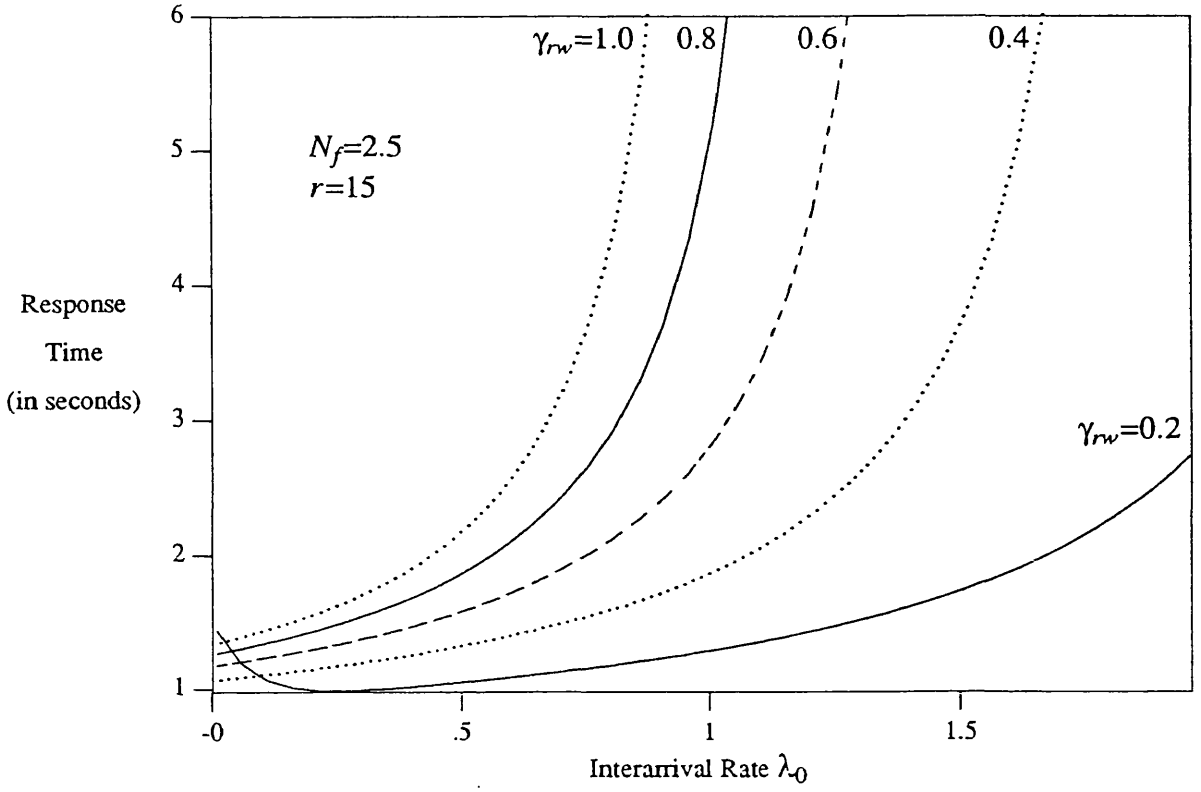


Figure 5.23. Response Times with Different Read-Write Ratio γ_{rw}

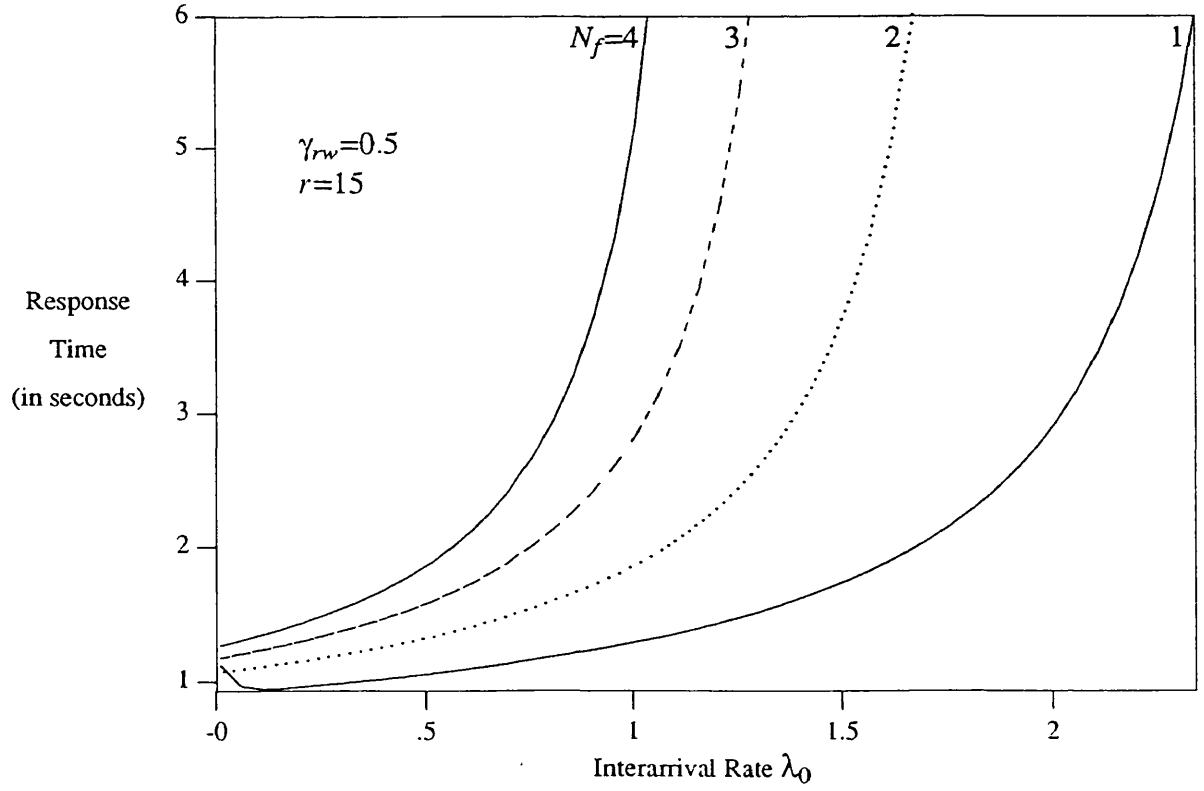


Figure 5.24. Response Time with Different Replicated Copies N_f

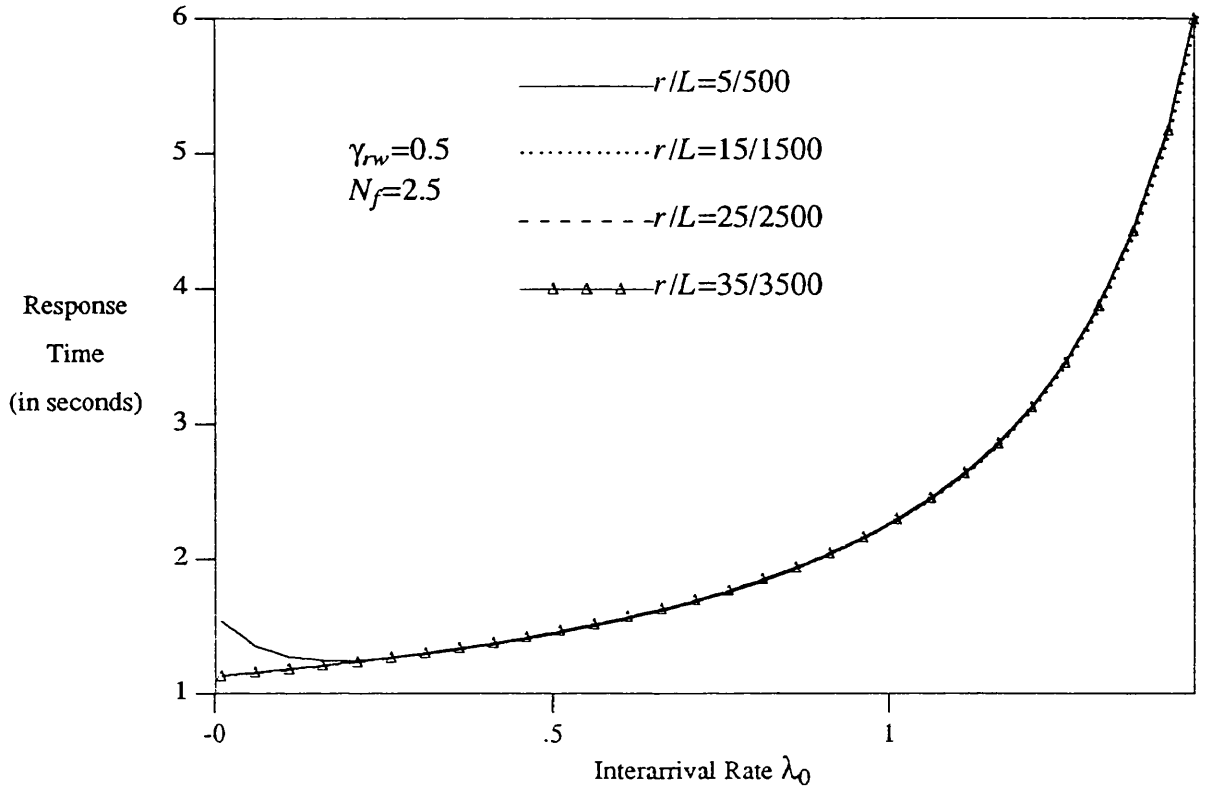


Figure 5.25. Response Times with Different Granularity r/L

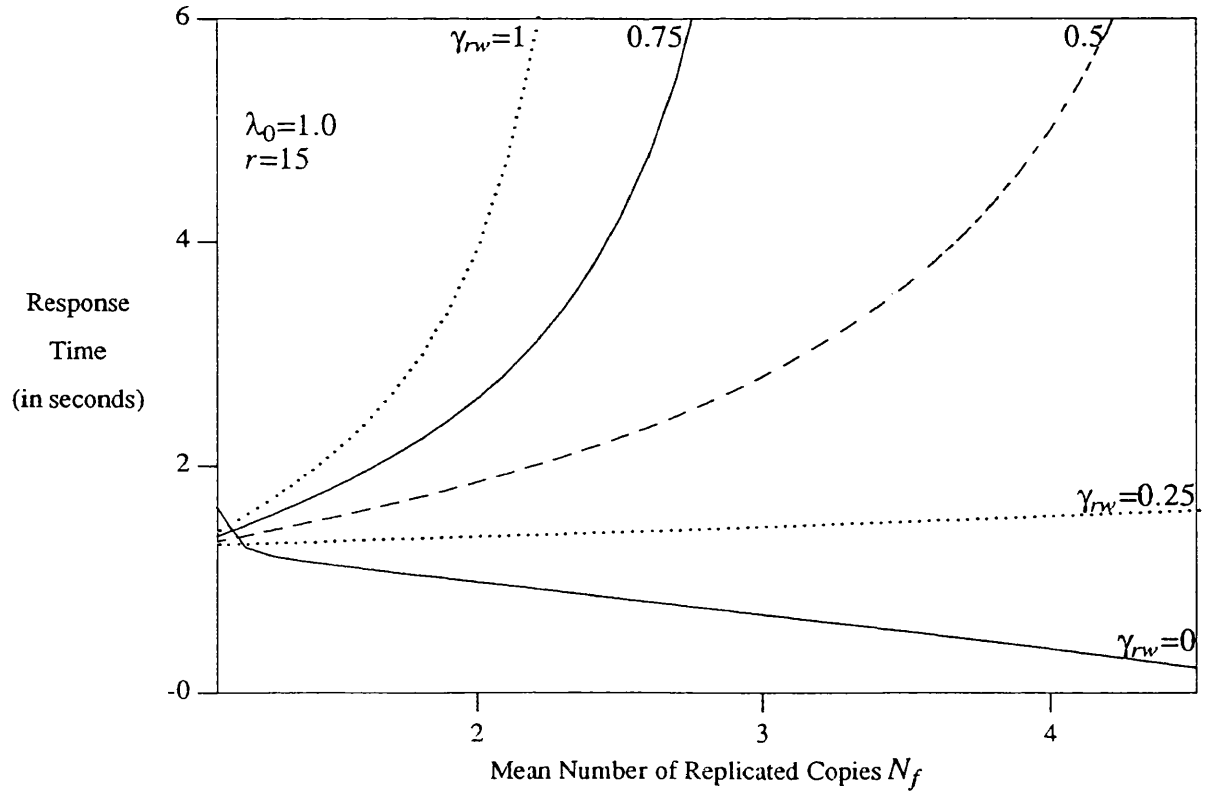


Figure 5.26. Response Times vs N_f with Different Read-Write Ratio γ_{rw}

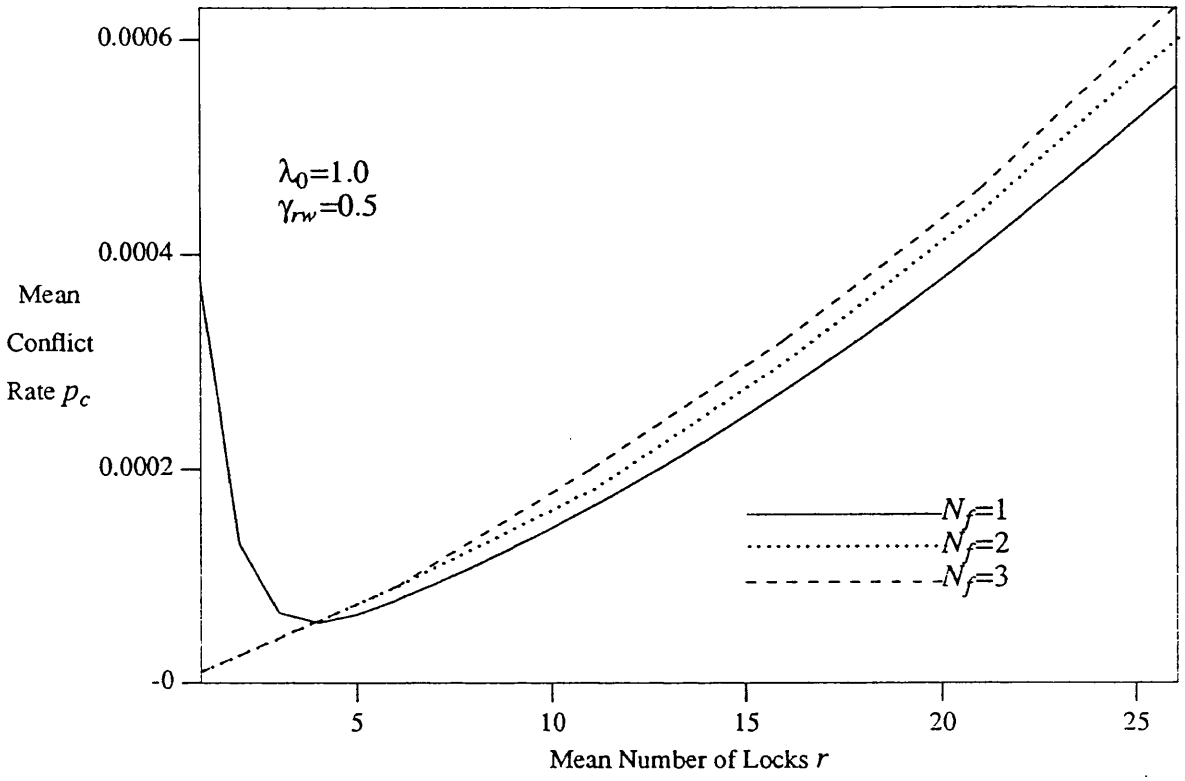


Figure 5.27. Mean Conflict Rates with Different Replicated Copies N_f

Figure 5.26 shows the change of response time with N_f under different γ_{rw}. The response time at the starting pionts, i.e. N_f=1, for the centralized 2PL is relatively smaller than that in other cases. This is because centralized 2PL protocol causes less blocking than other protocols. In the cases of low read-write ratio γ_{rw}=0 and γ_{rw}=0.25, the response times tend to decrease for γ_{rw}=0 and almost hold at the same level for γ_{rw}=0.25. The centralized system is clearly in favor of read oriented transactions.

Conflict Rate

The centralized 2PL algorithm causes much less locking conflicts than any other algorithm as shown in figure 5.27. it is also interesting to notice that the data replication does not affect conflict rates very much as opposed to the other 2PL algorithms. This is because that locking is performed not only at the same data copy but also at the same site. We can conclude from all the 2PL algorithms studied that the wider the locking operations are spread, the higher the locking conflict rates are. Figure 5.27 indicates how the number of duplicated copies N_f effects the conflict rate of lock request. The higher the N_f, the greater the conflict rate p_c. So N_f should be carefully chosen to obtain a good system performance. The parameters are given in table 4.8. There exists a thrash at $0 \le r < 4$ under N_f=1, which means that the mean conflict rate decreases as the mean number of required locks increases. It is likely caused by the error in numerical convergence when solving small q in equation (5.67). However the numerical method used here gives us satisfactory results in almost all the other cases.

5.4 Comparison of Major 2PL Protocols

We have so far studied four major 2PL protocols in distributed databases, i.e. basic 2PL, primary copy 2PL, majority consensus 2PL and centralized 2PL. Their performance results are obtained from the analytic models. It is the aim of this section to compare the performance of these 2PL protocols under various conditions. The performance results of each type of 2PL protocol are illustrated by two lines; one is set to the minimum value; the other to the maximum. The basic 2PL is presented with two solid lines; the primary 2PL with dashed lines; the majority consensus 2PL with dotted lines; and the centralized 2PL with delta solid lines.

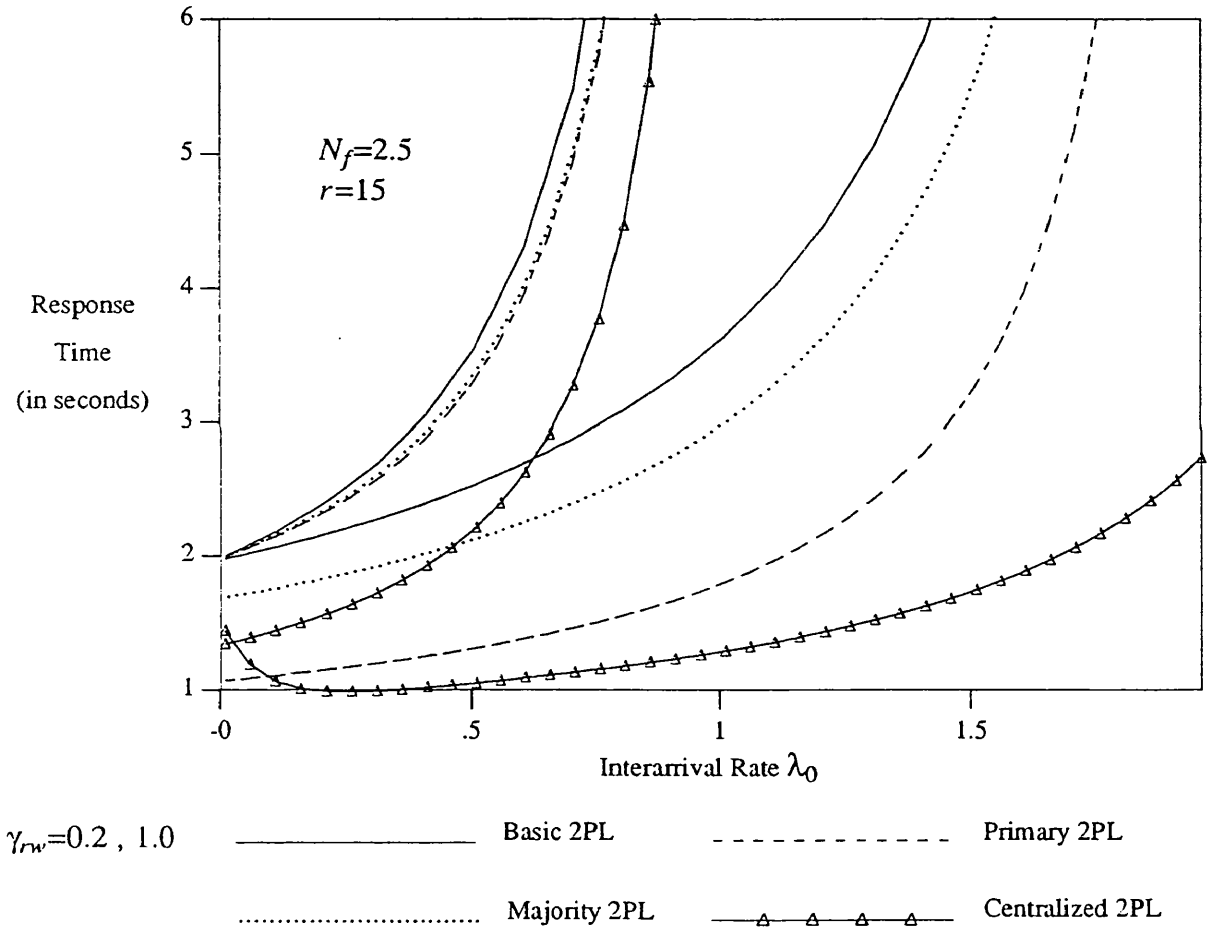


Figure 5.28. Response Times with Different Read-Write Ratio γ_{rw}

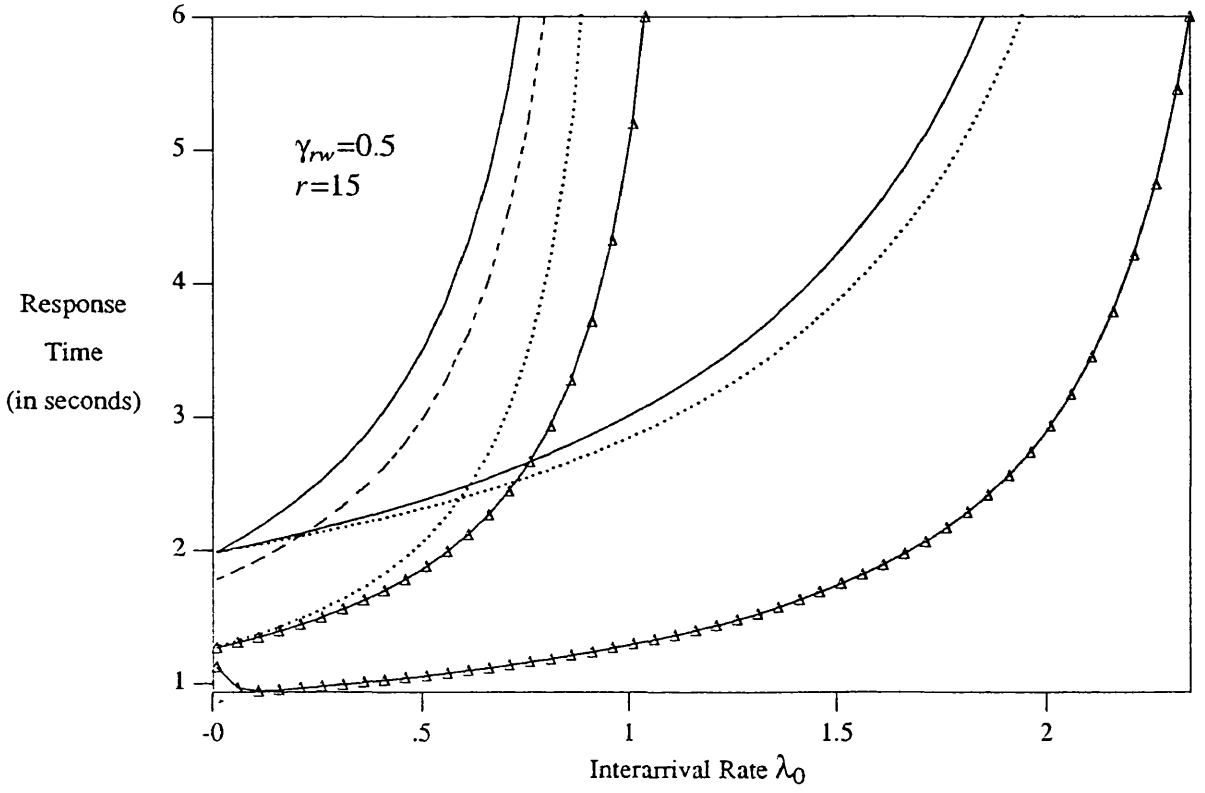


Figure 5.29. Response Time with Different Replicated Copies N_f=1 , 4

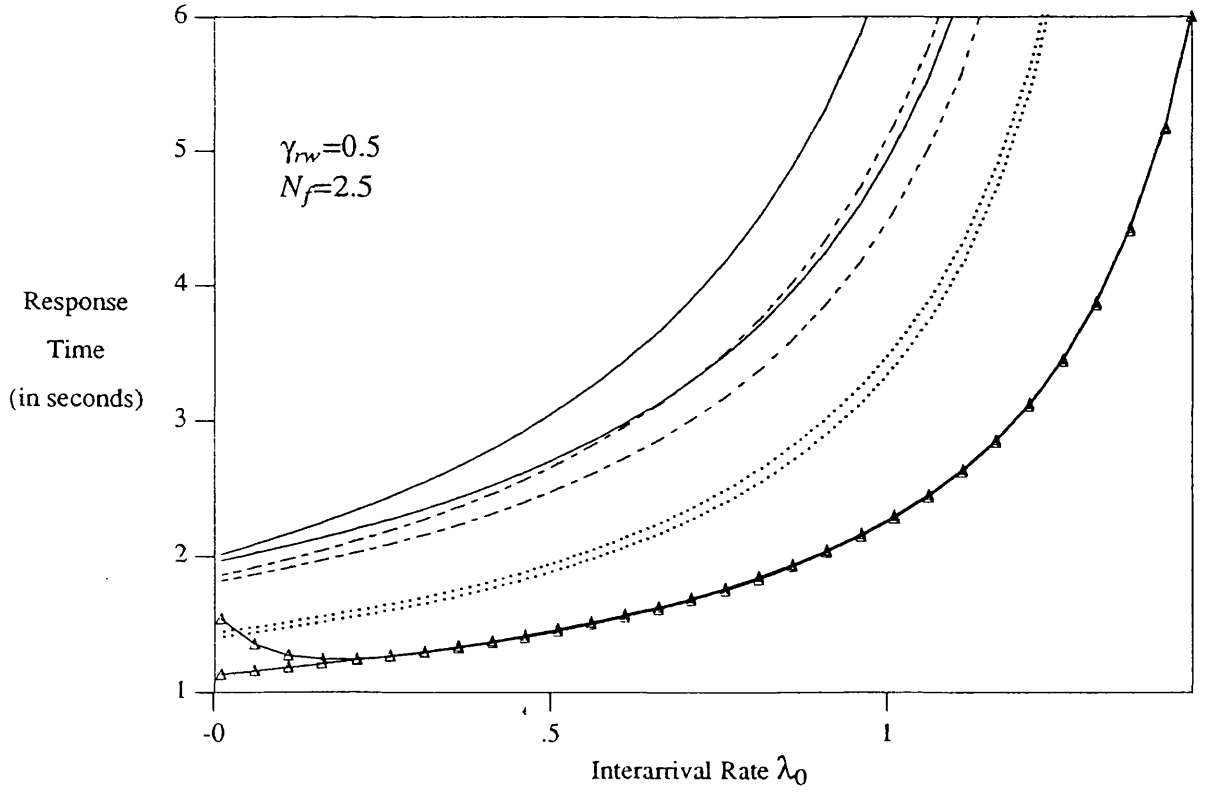


Figure 5.30. Response Times with Different Granularity r/L=5/500 , 35/3500

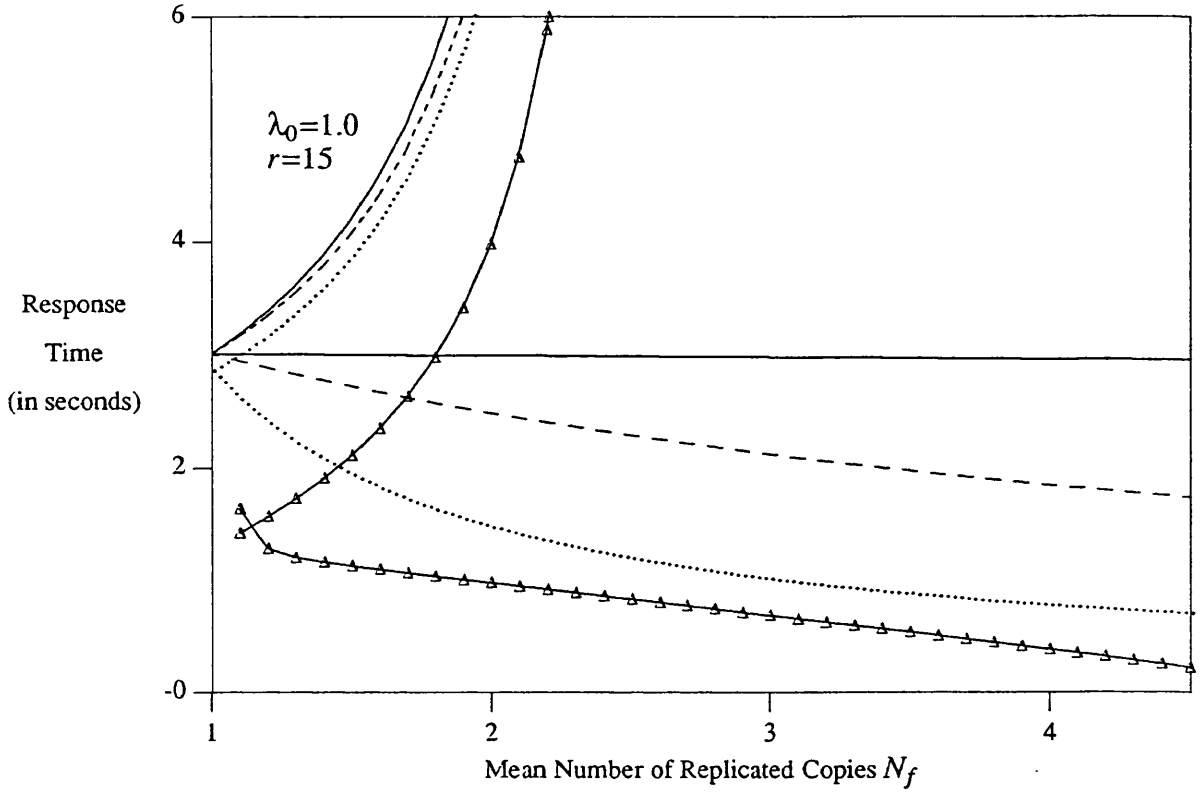


Figure 5.31. Response Times vs N_f with Different Read-Write Ratio γ_{rw}=0, 1

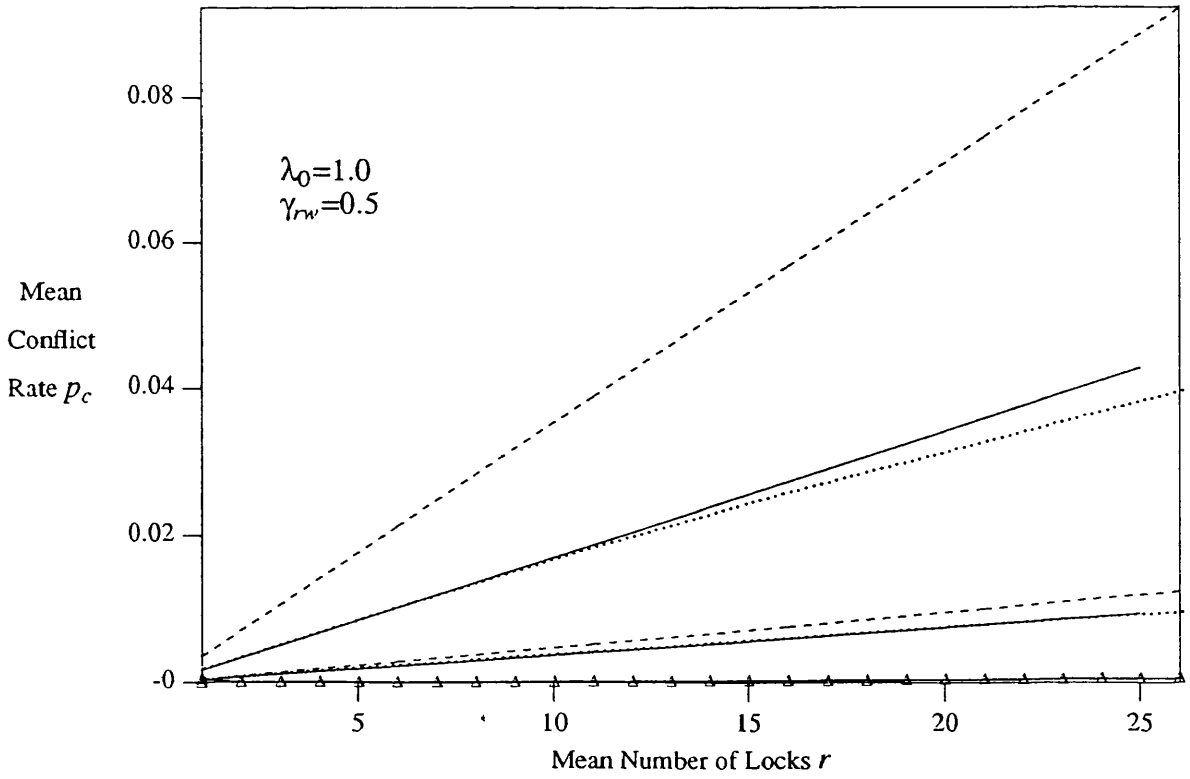


Figure 5.32. Mean Conflict Rate with Different N_f=1 , 3

Read-Write Ratio

The results shown in figure 5.28 indicate that the centralized 2Pl performs best, the primary 2PL is better than the majority 2PL and the basic 2PL has the worst performance. At the extreme case, i.e. $\gamma_{rw}=1.0$, the primary, majority consensus and basic 2PL perform nearly the same, while centralized 2PL is still much better than the rest.

Data Replication

Figure 5.29 illustrates the performance curves with different replicated copies. The best result occurs under $N_f=1$ with centralized 2PL. At low $\lambda_0(\lambda_0<0.6)$ the response times of the centralized and primary 2PL under $N_f=4$ are smaller than those of the basic, primary and majority 2PL under $N_f=1$. But at higher $\lambda_0(\lambda_0>0.6)$, the situation becomes the opposite. The performances of the basic and primary 2PL under $N_f=4$ are the worst among all the cases. The response times increase dramatically as interarrival rate approaches 0.7.

Lock Granularity

The effect of different lock granularities are shown in figure 5.30. Again centralized 2PL outperforms the rest. The performance can be graded in the following order, centralized 2PL first, majority consensus 2PL second, primary 2PL third and basic 2PL fourth. Centralized 2PL is least affected by the lock granularity, while basic 2PL is most affected.

Read-write Ratio vs Data Replication

Figure 5.31 illustrates the effect of read-write ratio vs data replication. Under read only case, i.e. $\gamma_{rw}=0$, the response time decreases as the mean number of replicated copies increase in the centralized 2PL, majority consensus 2PL and primary copy 2PL. While for basic 2PL, it almost holds at the same level. Under write only case, i.e. $\gamma_{rw}=1$, the basic 2PL, primary 2PL and majority consensus 2PL perform almost equally badly, while the centralized 2PL again outperforms the rest. The performance can be graded as the centralized 2PL first, majority consensus 2PL second, primary copy 2PL third and basic 2PL fourth.

Conflict Rate

The mean conflict rates of the four 2PL protocols are presented in figure 5.32, in which centralized 2Pl almost has no conflict rate. The majority 2PL and basic 2PL have higher but similar conflict reates. The range of conflict rates in the primary 2PL is significantly greater than the rest 2PL protocols. Especially under $N_f=3$, the conflict reate of the primary 2PL is much higher than the rest.

Generally speaking, centralized 2PL outperforms the other 2PL algorithms under the test condition. It is interesting to notice that the locking conflict rate is strongly associated with the distribution of the locking activities. The wider the locking activities spread, the higher the conflict rates are. Another interesting finding is that the degree of data replication has a big impact on the response time. By properly choosing

the parameter, optimal performance can be obtained. The finding of lock granularity being less effective to the overall performance is not surprising, since the lock overhead in the test system is quite small. These performance results are useful in comparing various concurrency control algorithms and designing distributed database systems.

5.5 Summary

In this chapter, the modeling method introduced in chapter 4 is consistently extended to model primary 2PL, majority consensus 2PL and centralized 2PL protocols. The analytic model is proven to be applicable to a wide range of distributed database protocols, especially 2PL. Many useful findings are obtained by comparing the evaluation results of these 2PL protocols.

Chapter 6. Empirical Evaluation of an Actual DDB

In the previous chapters, analytic models of distributed database systems have been introduced and validated by various simulation models. However it will be more convincing if the analytic model can be validated against an actual distributed database system. In this chapter, we shall present an implementation of a prototype distributed database system and measure the real system to validate the analytic model.

6.1 Introduction

Various distributed database systems have been designed and implemented in recent years[8]. Most of them employ point-to-point communication and static distributed database configuration. With the availability of the multicast communication facilities from the 4.3BSD UNIX kernel, efficient, robust and reliable distributed databases can be built. In this chapter we introduce a prototype distributed database system called the Distributed Robust File Store (DRFS) which is designed to manage replicated copies of data, and to provide consistent and concurrent access to the data. The reason for calling it a file store rather than a database is that we only concentrate on studying the concurrency control algorithm while simplifying the database interface language. The nature of the multicast protocol enables the system to achieve greater parallelism and thus high performance. High robustness is obtained by the introduction of dynamic configuration of transaction managers. Furthermore novel failure recovery algorithms are used to maintain the reliability of the system. The performance of the implemented system is measured in terms of the throughput and utilization of the system under different work loads. The measurement results are compared with the analytic results.

The sections in this chapter are organized as follows: section 2 describes the architecture of the system; section 3 introduces the algorithms and techniques used to manage the dynamic configuration of the transaction manager group; section 4 gives the concurrency control mechanisms and implementation details of transaction management; section 5 discusses the failure recovery algorithm and its implementation; section 6 presents the analytic model of the distributed database system built on an Ethernet; section 7 compares the measurement results with the analytic results obtained by mathematic modeling. Due to limited space in the thesis, implementation details of the DRFS system can be found in the references[60,50,56,57,51,58,59,52,53].

6.2 Architecture

6.2.1 General Architecture

The DRFS, as shown in figure 6.1, consists of three modules -- the Transaction Manager (TM), the Data Manager (DM) and the Multicast Network (McNet). The Transaction Managers (TMs) control the

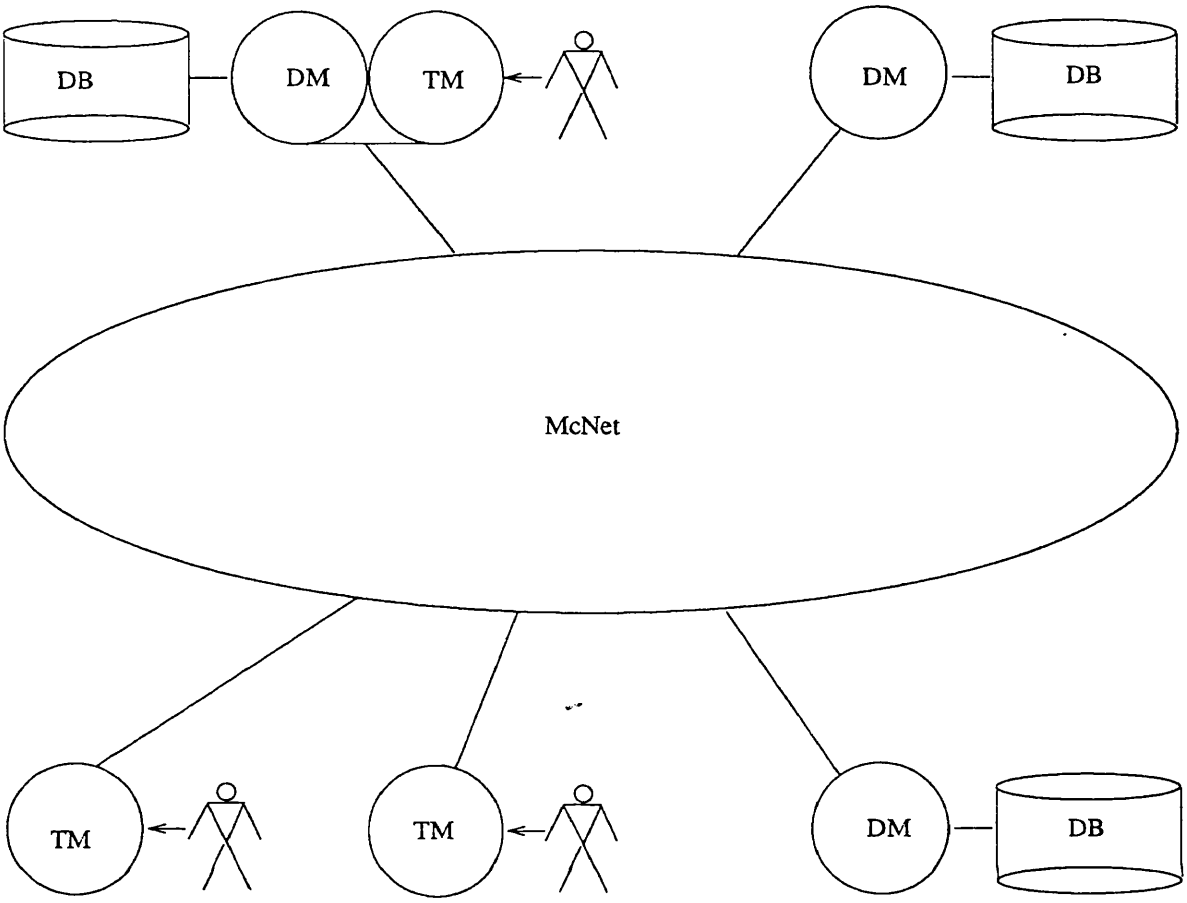


Figure 6.1. Architecture of the DRFS

distributed execution of transactions on user's behalf. They provide access interface for users and control concurrent transaction execution. The DRFS system maintains a dynamic configuration of all the active TMs by storing the current status of all the TMs at each TM site. A TM also contains the global schema of the data stored in the system, which records the names and number of copies of the files in the DRFS.

All data managed by the DRFS are stored by Data Managers (DMs). The DMs manage local data manipulations under the supervision of TMs. All the data controlled by the TMs have to be consistent.

The communication between the TMs and DMs is via a Multicast Network (McNet). The McNet is a multicast communication interface which provides the following services:

- atomic transaction control, which means that an update transaction is either done at all DMs or not at all;
- TMs group management, which maintains a dynamic configuration of the group of all the active TMs.

- failure recovery and retransmission, which guarantees that network failures will be recovered and no inconsistent state will be left after recovery.

6.2.2 Physical Architecture

The physical structure of DRFS is illustrated in figure 6.2. The system is built on a number of Sun workstations connected by an Ethernet local area network. The Sun workstation is a host computer which can be accessed from both the console and remote terminals. Some of the Sun workstations have file stores managed by local DMs. Others are simple workstations without their own file storage. The DMs are always built at the workstations with their own storage while TMs can be built on any types of workstations.

The Ethernet is a broadcast baseband local area network. It uses a carrier sense multiple access (CSMA) protocol to control transmissions and avoid collisions at the physical and link layer of the OSI reference model. The nature of Ethernet enables a broadcast type communications, which has the advantage of transmitting K messages at the cost of one. With the further development of interprocess communication protocols at network and transport layer, a new communication primitive called multicast is made available in 4.3 BSD UNIX system. Multicast is simultaneous transmission of data, rather like broadcast, but to a restricted and well defined set of destinations. It is a desirable communication primitive for the applications requiring communications between a group of processes.

6.2.3 Access Structure

Figure 6.3 illustrates the types of accesses to DRFS from a user. A user can only contact a TM which manipulates distributed replicated data copies. The types of the DRFS operations that a user can issue are Create, Delete, Open and Close files and Read and Write records. Consider a transaction involving creating three copies of a file in DRFS. A user first issues a Create_File command to a TM. Upon receiving a user request the TM coordinates a concurrency control protocol with a set of DMs to perform the distributed file creation transaction. Once a DM receives the file creation request from the TM, it performs locking and local file creation operations under the control of the coordinating TM. The DMs always sit there waiting for calls from a TM. It serves as an interface between file systems and the TMs. On one hand it acts as a participant of a concurrency control protocol with the coordinating TM. On the other hand it controls the file system to perform the required local file operations.

6.2.4 Program Structure

This DRFS architecture, shown in figure 6.4, gives a clearly layered programming structure with user at the top level, TM at the second level, DM at the third level and the actual files at the bottom level. The advantage of the clearly layered structure is to achieve efficiency and transparency. The efficiency lies on the fact that functions of programs at different levels can be modified easily and extended without affecting programs at other levels. The data transparency is achieved using automatic control of the data allocation

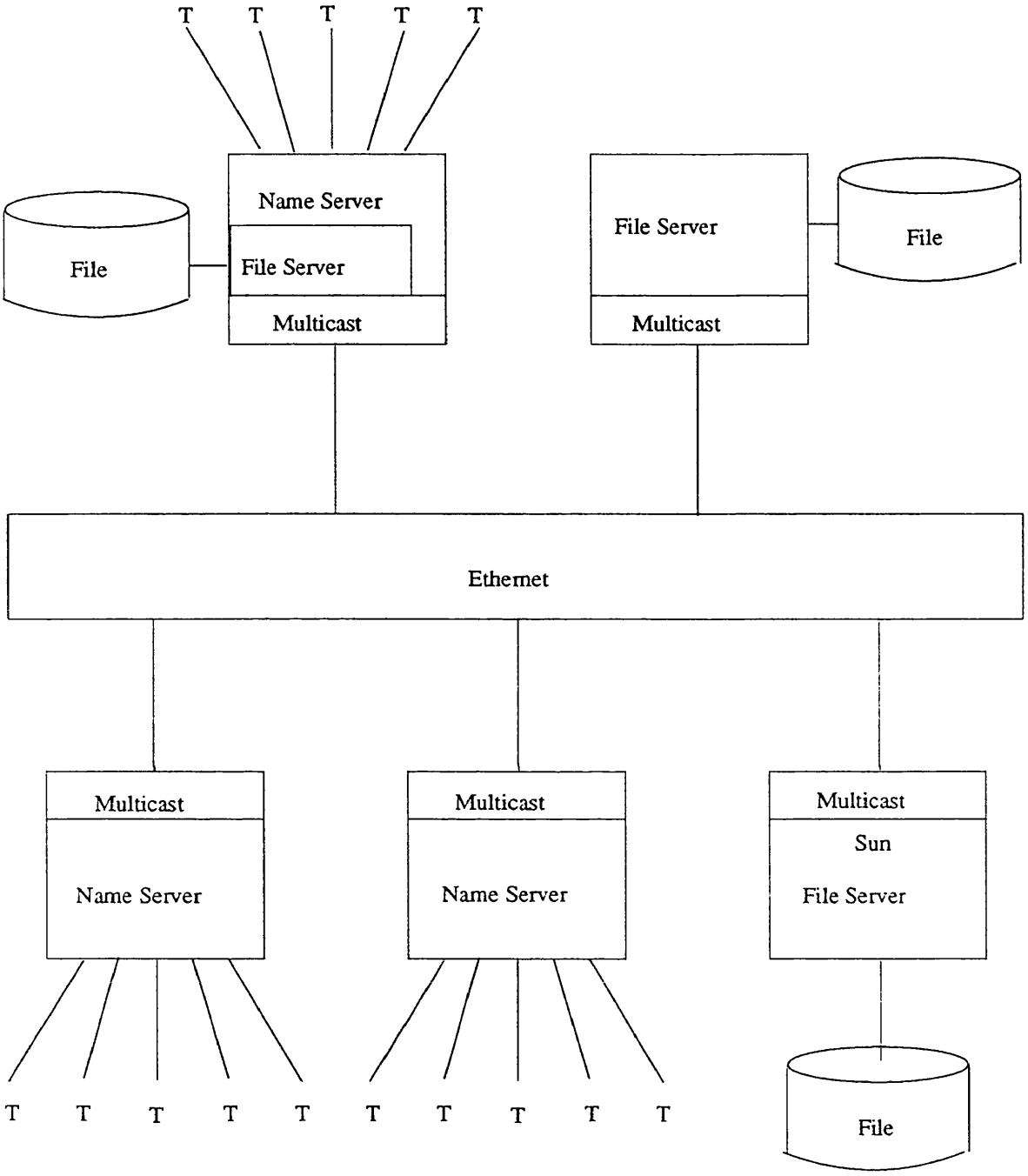


Figure 6.2. The Physical Structure of the DRFS

by the TM. The user needs only to provide the number of replicated copies that he/she wishes to distribute. The TM then chooses the actual locations of the copies for the user.

The TM processes are started by triggering the TM group management protocol to join the TM group. This joining operation is done once for all during the life time of a TM. Once a TM becomes an active member of the TM group, it keeps listening to user requests. When a user request is received, a TM controls the distributed execution of the transaction with the participation of a number of DMs. There are

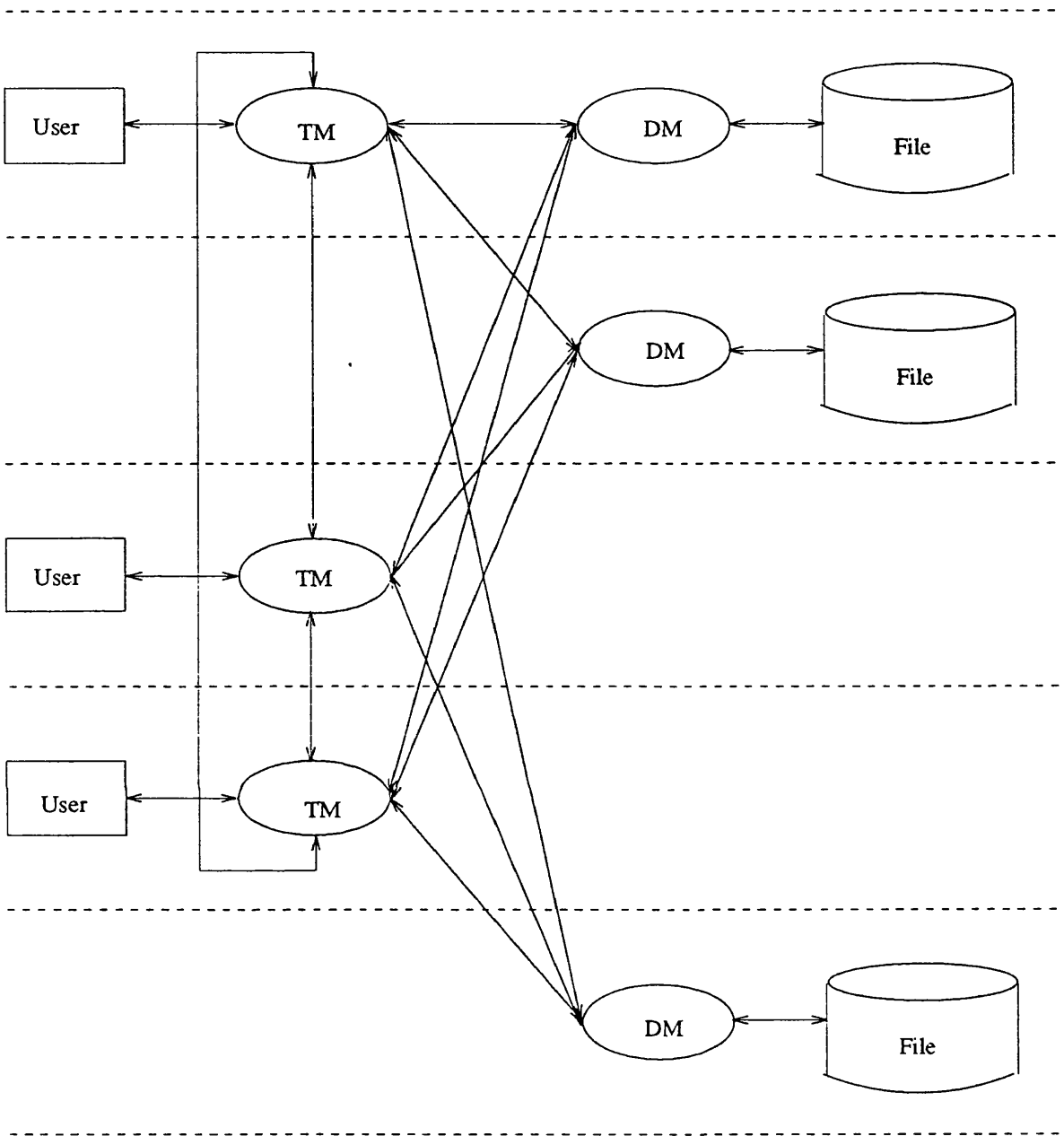


Figure 6.3. Access Pattern of the DRFS

two multicast groups: the TM group with address (tm_group), which supports communication within the TM group, and the TM-DM group with address (dm_group), which is for communication between TMs and DMs.

6.3 Dynamic Configuration of the TM Group

In conventional distributed databases, the configuration of TMs is static. This approach, however, makes the extension of a system difficult and reduces its reliability. In order to enhance the robustness of a distributed system we introduce a technique to maintain a dynamic configuration of the TM group. This

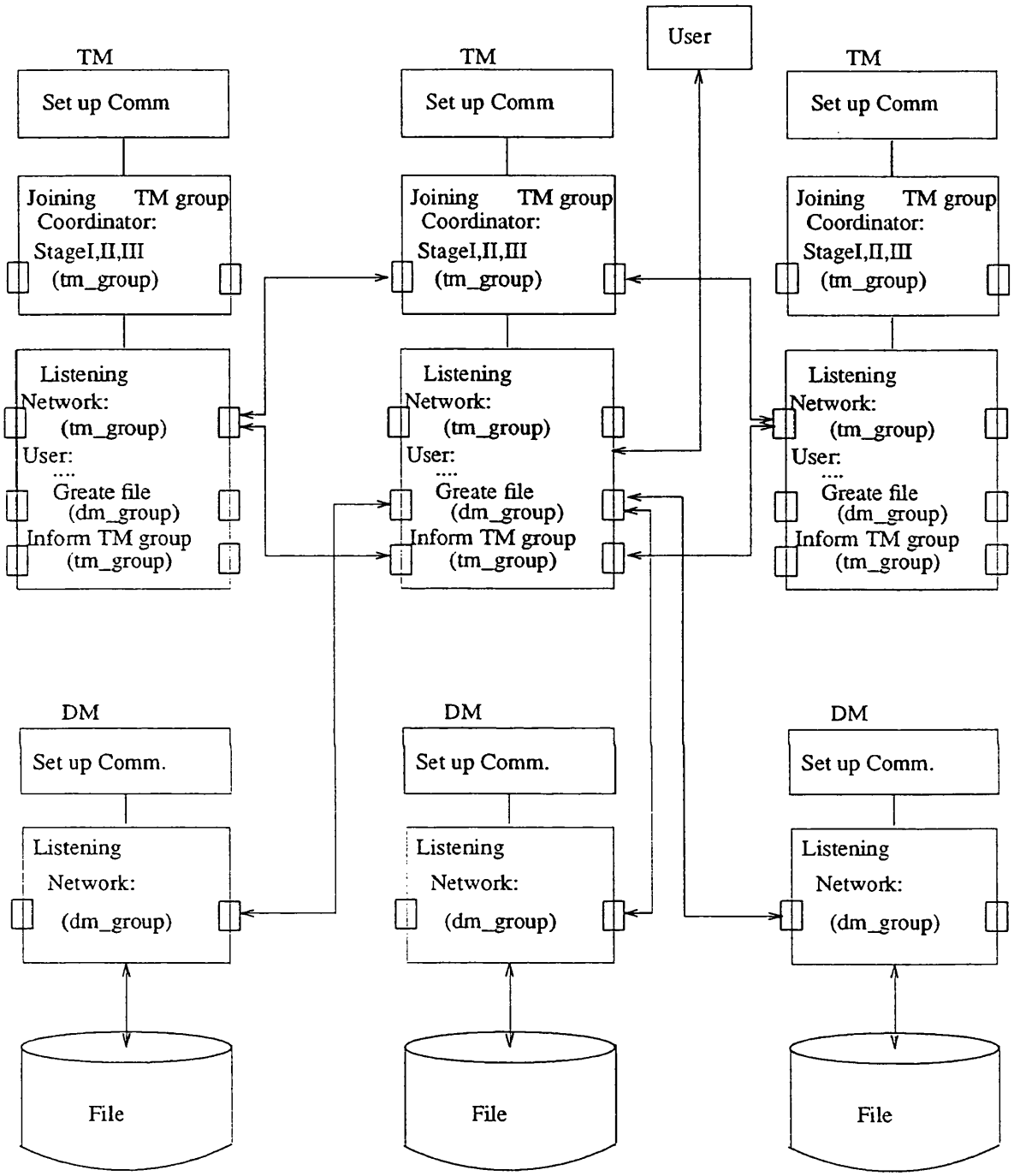


Figure 6.4. Program Structure of the DRFS

means that a TM can join or leave the DRFS at any time. A list of active TMs in the DRFS is always consistently maintained. A new TM service can be easily set up when ever required and an existing TM service can be removed when ever not required. Upon site failure, the affected TM will be eliminated immediately from the TM group. Therefore this additional feature greatly increases robustness and reliability of the system.

6.3.1 Group Management Protocols

Three protocols are used to manage the dynamic configuration of the TM. Protocol P1 involves obtaining the dynamic configuration of TMs; protocol P2 propagates the modified configuration to all the existing TMs and protocol P3 obtains the global data schema for the new TM. Setting up a new TM requires all three protocols (P1, P2 and P3), since the new TM must first acquire the current status of the existing TM group, then adds itself to the group, controls the propagation of the modified configuration to other TMs and finally obtains the global data schema to provide data management service to users. Deleting a TM obviously requires only protocol P2.

The algorithms of protocol P1, P2 and P3 are as follows:

Protocol P1: Get TM Group Membership List

New TM:

- multicasts HELLO to the address of the TM group.

Existing TMs:

- receive the request and send back a copy of the membership list of TMs with command MLIS_RESP. This ensures that even if only one TM responds to the request, the new TM will have full knowledge of the group membership.

New TM:

- waits for reply from the existing TMs.
  - — If no answer is received when timeout, repeat the whole procedure;
  - — If there are more than three time-outs, the new TM sets up an empty global data schema;
  - — Otherwise at least one reply is received and the protocol is successfully completed.

Protocol P2: Join (/Leave) TM Group

Phase 1:

New TM:

- multicasts ADD_ME to the TM group (tm_group).

Existing TMs:

- receive the message and check the accuracy of Memberlist. If it is correct, set up a pre-write of the new list and acknowledge the receipt of the request with OK. The pre-write is used to detect conflict.

Phase 2:

New TM:

- collects replies from existing TMs.
  - — If the new server has got a positive reply from everybody, it multicasts COMMIT.
  - — If one of the replies is SORRY, the sender multicasts ABORT.
  - — Otherwise it re-transmits the request until all known group members have responded or, eventually, it times out and assumes that the missing ones failed. It is then allowed to proceed with the remaining group members.

Existing TMs:

- wait for replies from the new TM.
  - — those who get the COMMIT message delete the pre-write list, update the membership list and acknowledge the request (ACK).
  - — those who get an ABORT message only delete the pre-write list and acknowledge the request (ACK).

New TM:

- has to collect ACKs and retransmit until either it has all of them or (at long last) has to assume failure of the missing ones. If the transaction has to be aborted, it will restart from protocol P1. If all acknowledgements are collected, it multicasts an acknowledgement to the receivers.

Existing TMs:

- listen to the sender's acknowledgement. If they are not received before timeout, a failure recovery procedure has to be trigged.

Protocol 3: Get Global Data Schema

New TM:

- selects an existing TM from the membership list and unicasts a FLOP_REQ to the selected TM to obtain a global data schema.

Selected TM:

- Upon receiving the FLOP_REQ, the selected TM sends back a copy of the global data schema.

New TM:

- If the new TM receives the global data schema within a given time the protocol is terminated; otherwise it repeats the procedure starting from step 1.

6.3.2 Protocol Implementation

Protocol P1 and P3 are one phase protocols and P2 is a modified basic two phase commit protocol. Protocol P1 is coded in subroutine get_memlist() at the coordinator TM and subroutine listenall() at the participant TM; protocol P2 is built in subroutine update_mem() and listenall() and protocol P3 is in get_nslist() and listenall() respectively. Part of the functions performed by listenall() procedure is to listen to the network (tm_group) address. It acts as a participant TM to respond the request from a coordinating TM in protocol P1, P2 and P3 introduced above. It responds to the coordinating TM's HELLO with MLST_RESP, ADD_ME with OK/SORRY, DEL_ME with OK/SORRY, COMMIT with ACK, ABORT with ACK and FLOP_REQ with FLOP_RESP.

In protocol P2 a version-numbering technique has been used to improve the efficiency of the protocol. Each version of the membership list of the TM group is marked with a version number. All the copies of the membership list stored at different TMs have the same version number. The version number is updated consistently in all the TMs whenever there is a change of membership in the TM group. This technique enables the protocol to transmit and update only the version number rather than the entire membership list. Another technique used in P2 is to use prewrite to control locking. At the first phase of P2 a prewrite is used to reserve the right to exclusively use the memberlist for updating. This is done by increasing the version number of the memberlist by one. For example, suppose TM_i and TM_j both want to leave the DRFS when the version number of membership list is n. Suppose TM_i triggers the protocol P2 first. The version numbers at all the TMs are then checked and increased to $(n+1)$. Suppose TM_j triggers P2 when its version number is still n. TM_j then checks its version number with other TM's version number. It is obvious that some conflicts will occur therefore TM_j's update will be rejected.

6.4 Transaction Management

The transaction management deals with the actual distributed data manipulation at the record and file levels. The consistency of the DRFS is maintained by the transaction management protocols. Each transaction is executed as an atomic operation, which means that it is either done completely or not at all. The granularity of lock in the DRFS is record. On the physical file each record is associated with a lock field. Locking is physically performed by setting or resetting the lock field of a record.

6.4.1 Transaction Management Protocol

A transaction is executed between a TM as the coordinator and a set of DMs as the participants. In order to achieve robustness of the system, we employ a majority consensus two phase commit protocol. Record update is propagated to a majority of the DMs holding the record. Reading a record involves asking all of the DMs to send the record with its time stamps and select a most up to date copy from a majority of the replicated records.

Protocol P4: Majority Consensus 2PL

Phase I:

Coordinator (TM):

- Issue an atomic transaction.
- Multicast the transactions to the participants in (dm_group).

Participants (DMs):

- Wait for a transaction.
- Receive a transaction.
- Try to lock.
  - — If locks are granted, send OK to the coordinator.
  - — If locks are not granted, send SORRY to the coordinator.

Phase II:

Coordinator (TM):

- Waits for Answer messages.
  - — If a majority of OK messages are received, it multicasts COMMIT to the participants in (dm_group).
  - — If less then a majority of OK messages are received at time-out or a SORRY message is received, multicast ABORT to all participants in (rm_group).
- Waits for ACK messages from the participants.

Participants (DMs):

- Wait for command message.
  - — If command is COMMIT, execute the transaction and release locks.
  - — If command is ABORT, release locks.
- send ACK to the coordinator.

Coordinator(TM):

- waits for ACK responses
  - — If a majority of ACK messages are received, complete the protocol.
  - — otherwise trigger recovery protocol.

6.4.2 Protocol Implementation

Protocol P4 is designed to deal with data manipulation transactions, such as Create, Delete, Open, Close, Read and Write, under user's request.

A user's request is picked up by the listenall() procedure. Once a user's request is picked up, the TM will start to coordinate a transaction with a group of DMs by calling RFCreate(), RFDelete(), RFOpen(), RFClose(), RRRead(), and RRWrite() accordingly.

In each TM, a global data schema is stored in its stable storage. The global data schema is updated and kept consistent at all the TMs upon file creation and deletion operations.

Since only DMs store the actual data of the DRFS, it must keep a list of the local data files stored in them. We call this list the local data schema in contrast to the global data schema stored in TMs. The local data schema, which contains the names and locks of the data stored in each DM, is stored in the stable storage. Moreover it also keeps a dynamic open-file-table in the main memory to record the status of the files used at run time.

The protocol is implemented in procedure PhaseI_maj() and PhaseII_maj(). PhaseI_maj() procedure performs the first phase of protocol P2 from the coordinator node. It tries to enforce all the participants to take part in the transaction. Those who store the data will respond. At the end of the first phase, a list of DMs, which have responded, is recorded in a vector (testmem). The vector (testmem) is then packed into the multicast COMMIT or ABORT message and sent to all the DMs in the second phase. Upon receiving the message, the DM will first check the name list. If the DM's own name is found in the list, it either COMMITs or ABORTs and then sends an ACK; otherwise it simply clears the transaction and does not send ACK.

The PhaseII_maj() procedure performs the second phase of two phase commit protocol from the coordinator node. It multicasts either COMMIT or ABORT according to the status passed from the first phase (phaseI). It requires all the previously responded DMs to either COMMIT or ABORT the transaction. The name list of the previously responded DMs is packed in and sent with the COMMIT or ABORT command to (dm_group) to enable those particular DMs to participate in the transaction.

A listen() procedure is used to enable the DMs to respond to the TM's request. It acts as the participant of the file manipulation transaction in protocol P4. In the first phase of the transaction, the DM receives a CREAT_F, DEL_F, OPEN_F, CLOSE_F, READ_F, or WRITE_F request from a TM. It then performs required locks at the physical level by calling PREFS_fcreate(), PREFS_fdelete(), PREFS_fopen(), PREFS_fclose(), PREFS_rread(), or PREFS_rwrite() accordingly. In the second phase, the DM receives either a COMMIT or ABORT from the TM. It then executes the transaction by calling FS_fcreate(), FS_fdelete(), FS_fopen(), FS_fclose(), FS_rread(), or FS_rwrite() accordingly. The correct match between the messages received in the first and second phases is maintained by a buffer manager. The message received in the first phase is put into the buffer with its transaction id. The COMMIT or ABORT message received in the second phase is checked against the transaction id in the buffer. If it is matched, the DM will execute the transaction according to the content of the message buffered in the first phase; otherwise it simply throws away the message. At the end of the execution, the message is deleted from the buffer.

6.4.3 Protocol to Update Global Data Schema

Since the file creation and deletion transactions change the content of the global data schema stored at all the TMs, it is necessary to propagate the change to all the TMs by using protocol P5. A TM first gets the file creation or deletion command from user; and then acts as a coordinator of the update transaction while other TMs act as the participants. We use a special one and half phase protocol to execute the propagation. Crash recovery algorithms are used in the protocol to cope with either coordinator failure or participant failure.

Protocol P5 Update Global Data Schema

Phase 1:

Coordinator (TM):

- sends CREATE_F or DEL_F command to all the other participant TMs.

Participants (TMs):

- Upon receiving the CREAT_F or DEL_F message, each participant updates its own global data schema by calling upd_flop() or delete_in_flop() and replies with an ACK.

Phase 2:

Coordinator (TM):

- receives the ACKs and records the name of the acknowledged participant.
- If all the ACKs from the participant TMs are received, it multicasts a further FLOP_ACK to all the TMs; otherwise it passes the name list of the participants who did not reply; and triggers the recover protocol.

Participants (TMs):

- wait for FLOP_ACK from the coordinator. If the FLOP_ACK is correctly received the protocol is terminated; otherwise a recovery protocol will be triggered.

6.5 Failure Recovery

The robustness of the DRFS system lies on its ability to cope with unexpected failures such as network failure, host failure and process failure. These failures can happen at any stage of a transaction. It can cause either the coordinator or the participant of a transaction to fail. A two phase commit protocol is widely admitted as not being resilient to network failures especially to coordinator crash. The recovery procedures are designed to cope with this problem.

The recovery procedures are used recursively in all stages of protocols. To collect K responses, a checkup vector is set up to record the number and names of the receivers. If the number of responses is less than K, the recovery procedures will be triggered. It retransmits the previous command and recursively

collect the K required responses until succeeds.

6.5.1 Participant Failure Recovery

Protocol P2 is used whenever there is a new TM joining or an old one leaving, the member list of the existing TM has to be changed. However, various failures can occur during the execution of the protocol.

During the first phase of protocol P2, the coordinating TM sends ADD_ME or DEL_ME to all the participant TM in the member list. Upon receiving the message from coordinator each participant replies OK to the coordinator. The coordinator is supposed to collect all the answers from the participants. If one or more participants fail to reply when time out, the coordinator then sends a ABORT message to all the participants to abort the transaction. So that no inconsistent updates can ever happen at this stage. The coordinator will restart the transaction after a random wait.

During the second phase of protocol P2, the coordinator sends either COMMIT or ABORT to all the active participants. Upon receiving the message each participant will response with an ACK. The coordinator is then expecting the ACKs from all the participants. It records the names of the participants who have replied with a ACK. If any failure takes place during this stage, the coordinator will be unable to receive all the ACKs, in which case the recovery protocol will be triggered to restore the system to a consistent and correct state. It should be noted that after the coordinator sends COMMIT message, there is no other way but to continue the transaction with the current state. Therefore the recovery protocol tries to delete the failed participant from the active member list and informs all the active TMs about the membership modification.

6.5.2 Coordinator Failure Recovery

The coordinator failure recovery is the most difficult problem associated with the two phase commit protocol. Here we use a novel method in P2 and P4 to deal with this problem. There are two cases we have to deal with. One is the failure during the first phase of two phase commit protocol, the other is the failure at the second phase.

If the coordinator fails at the first phase of P2 after sending the ADD_ME or DEL_ME message to all the participants, it is the responsibility of the participants to trigger the recovery process. Upon receiving ADD_ME or DEL_ME, each participant sets a random time out to monitor the coordinator. If it does not receive any COMMIT or ABORT from the coordinator when time out, it triggers the recovery protocol. Therefore the coordinating TM is deleted from the member list after recover. Upon receiving the recovery message, the rest of the participants removes the monitoring time out.

We shall now consider the case in which the coordinator fails at the second phase after sending COMMIT or ABORT to all participants. If all the participants have received the COMMIT or ABORT message, the transaction is propagated correctly anyway. The failure of the coordinator will then be detected by the next arriving transaction. If somehow one or more participants have not received the

COMMIT or ABORT when time out, it will trigger the recovery process in just the same way as it does in the first phase recovery.

6.5.3 Recovery Protocol

The recovery protocol mainly performs two tasks, to confirm the status of the failed TM and to delete it from the member list.

Protocol P6: Failure Recovery

Coordinator:

- Unicast YOU_DEAD? to the failed node;

Participant:

- If busy, send BUSYF; otherwise send ALIVE;

Coordinator:

- waits for reply from the failed node.
  - — If an ALIVE is received, continue from where it stopped.
  - — If a BUSYF is received, wait for a random period of time and continues from where it stopped.
  - — Otherwise restart from the beginning.
  - — If no answer is receive after sending YOU_DEAD? for three times, delete the crashed TM from membership list.
    - — Multicast DEL_FNS with the reduced membership list to the TM group (tm_group);
    - — Collect all the ACKs from TMs. (*The recover procedure is called recursively if there is a failure at this stage.)

The protocol is implemented in procedure recover() for participant failures and cor_recover() for coordinator failures. The name list of the failed servers is passed as a parameter to the recovery procedure. A checking message called "YOU_DEAD?" is first sent to the failed server to check the failure. If there is no reply after three time out, the failure is confirmed. The failed server is then deleted from the member list. The notification of the failure is propagated to all the TMs by sending a DEL_FNS message and collecting all the ACKs.

It should be noted that failures could happen inside the recovery procedure. Therefore the recovery protocol is used recursively.

6.6 Analytic Model of the DRFS

6.6.1 Analytic Model of the Ethernet

Ethernet is a local area bus network which employs carrier sense multiple access (CSMA) technique[20]. We shall first introduce the modeling method for general broadcast networks and then apply the method to a specific bus network, i.e. Ethernet. Two most popular broadcast protocols are pure ALOHA and slotted ALOHA.

Pure ALOHA

The basic idea of pure ALOHA protocol is very simple[1]. A user can transmit a packet at any time without any sensing or synchronization. If it receives the packet correctly, the transmission is successful; otherwise a collision is assumed to occur during the packet transmission period and the packet has to be retransmitted. The retransmission is followed by a random delay to avoid further collision.

Let us consider a packet transmission using a pure ALOHA algorithm. A packet transmission period is denoted as P secs. Suppose a packet is transmitted at time 0 sec and finished at time P sec, any other packet transmitted within the period of $2P$sec will cause collision. Thus the vulnerable period of pure ALOHA is 0–$2P$sec as shown in figure 6.5.

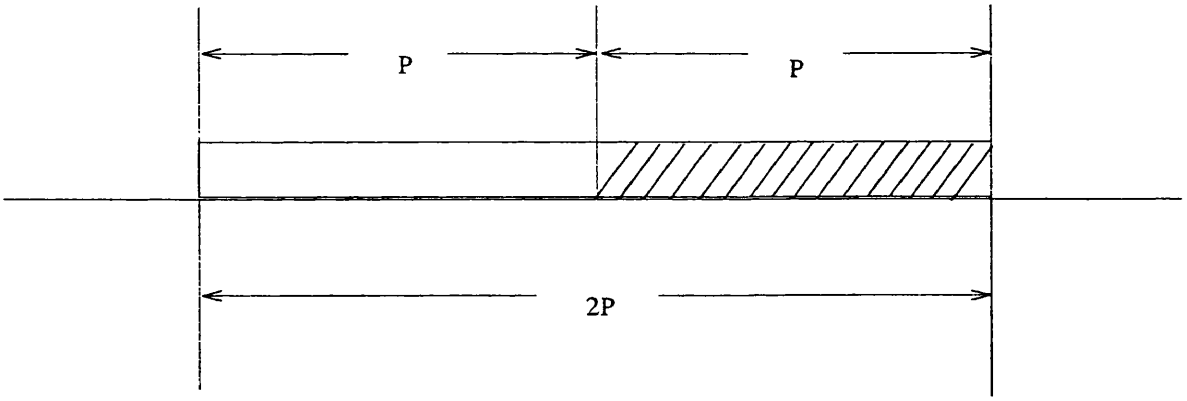


Figure 6.5. Vulnerable Period for Pure ALOHA

Let us define λ as the number of packets successfully transmitted per sec when the system is stable. λ is also the initial arrival rate of the packets. The parameter g is denoted as the number of packets actually generated per sec. g is also denoted as the attempted transmission rate. It is easily seen that the probability of no other packet is generated during $2P$ period is

$$q_p = \frac{\lambda}{g} = Prob\{\ successful\ transmission\ of\ a\ packet\ \}$$
$$= Prob\{\ no\ other\ packet\ is\ generated\ during\ 2P\ period\ \} \qquad (6.1)$$

Suppose the attempted transmission forms a Poisson process, that is, the probability of i transmissions

during time t is

$$P_i(t) = \frac{(gt)^i}{i!} e^{-gt} \tag{6.2}$$

Then

$$q_p = P_0(2P) = e^{-2gP} \tag{6.3}$$

Substituting q_p into the previous equation we have

$$\lambda = g e^{-2gP} \tag{6.4}$$

The above equation, derived by Kleinrock[37], can be used to calculate λ or g. If the initial packet arrival rate λ is given, the generated packet rate g can be calculated from the above equation using numerical methods. On the other hand, λ, which is also interpreted as the channel throughput, can be easily obtained with a given g.

One drawback of the analysis method introduced by Kleinrock for a broadcast network is that the communication channels can only be modeled in isolation. There is no available method to integrate the above modeling technique with those used in modeling other computer system components such as cpu, i/o, etc. The following section addresses this problem.

Phase Method

Here we intend to overcome this problem by introducing a new method to model the broadcast channel in the form of queueing networks, which allows the broadcast channel to be presented in a consistent way with the rest of the distributed database components. Taking the pure ALOHA case for example, a packet may be transmitted for several times before being successfully delivered. Each transmission attempt can be represented by a transmission phase. At the end of each transmission phase, the packet will either enter the next transmission phase with probability $1-q_p$ upon unsuccessful delivery, or leave the system with probability q_p upon successful delivery. The overall model can be represented by an infinite number of phases with a success probability of q_p as shown in figure 6.6.

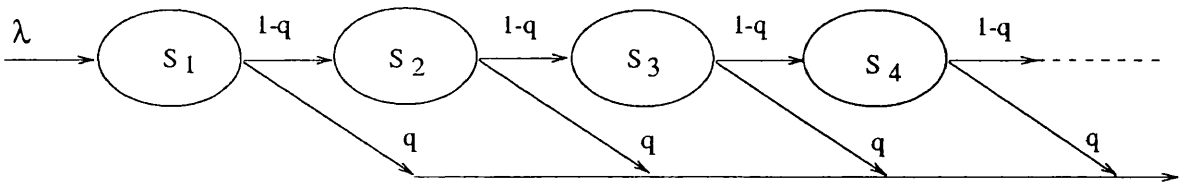


Figure 6.6. Queueing Network Model of Broadcast Communication Channel

The time spent in the first phase for the first transmission is composed of the propagation time d secs and the packet transmission time P. That is

$$S_1 = d+P \tag{6.5}$$

The time spent in the subsequent phases for the following retransmissions is composed of d, P and an additional random wait W_r which is used to avoid further collision. Thus

$$\begin{aligned} S_i &= d+P+W_r \qquad i=2,3,\cdots \\ &= S_2 \end{aligned} \tag{6.6}$$

Applying Cox's phase method[17], the Laplace transform of the above model is given by

$$\begin{aligned} f^*(s) &= \sum_{i=1}^{\infty}(1-q_p)^{i-1}q_p\prod_{k=1}^{i}\frac{1}{sS_k+1} \\ &= \frac{1}{sS_1+1}\left[q_p + (1-q_p)\left[\frac{1}{s\dfrac{S_2}{q_p}+1}\right]\right] \end{aligned} \tag{6.7}$$

The above Laplace transform $f^*(s)$ implies that the queueing model has two phases. The first phase with mean time S_1 is compulsory. It clearly represents the first attempted transmission. After the first phase a packet may be successfully transmitted with probability q_p or may enter the second phase with probability $1-q_p$. The second phase represents the sum of the subsequent retransmissions. The total time used by these retransmissions equals S_2/q_p. The reconstructed model is shown in figure 6.7.

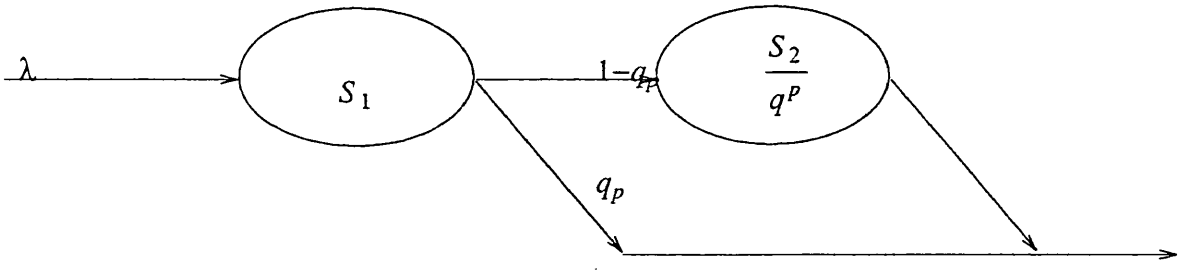


Figure 6.7. Reconstructed Queueing Model for Broadcast Network

The model can be interpreted as a network of queues with two infinite-server (IS) service centres, i.e. the $M/G/\infty$ queues. The mean number of customers, i.e. packets, in the system can be immediately obtained by

$$\bar{n} = \lambda S_1+\lambda(1-q_p)\frac{S_2}{q_p} \tag{6.8}$$

and the mean transmission time

$$\bar{R} = S_1+\frac{1-q_p}{q_p}S_2 \tag{6.9}$$

Substituting S_1 and S_2 in the above equations, we have

$$\bar{n} = \lambda\left[d+P+\frac{1-q_p}{q_p}(d+P+W_r)\right] \tag{6.10}$$

and

$$\bar{R} = \left[d+P+\frac{1-q_p}{q_p}(d+P+W_r)\right] \tag{6.11}$$

The results are extremely simple and easy to be applied to the queueing network model.

Slotted ALOHA Protocol

The difference between slotted ALOHA and pure ALOHA is that slotted ALOHA divides transmission time into slots which are equal to the transmission time of a packet, i.e. P sec. Each transmission is further synchronized to the slotted time interval. Collisions only occur during one slot period. Therefore the vulnerable period for slotted ALOHA is P sec rather than $2P$sec. Upon collision a packet must wait a random K slots before retransmission. Adopting the similar method used in analyzing pure ALOHA, we can easily obtain the probability of no collision by

$$q_p = \frac{\lambda}{g} = e^{-gP} \tag{6.12}$$

The phase method for broadcast network introduced above can be easily applied to the slotted ALOHA. Thus the number of packets in the system is give by

$$\bar{n} = \lambda\left[d+P+\frac{1-q_p}{q_p}(d+P+W_r)\right] \tag{6.13}$$

and the transmission delay is given by

$$\bar{R} = d+P+\frac{1-q_p}{q_p}(d+P+W_r) \tag{6.14}$$

It should be noted that the propagation delay and the random wait of the slotted ALOHA are different from that of the pure ALOHA. They are measured by slots. Suppose the random wait of a packet is uniformly distributed among 1 to K slots, that is, the packet will be retransmitted in one of the slots, $d+P, d+2P, \cdots, d+KP$ with equal probability. Therefore the average number of slots spent in the random wait is

$$\frac{K-1}{2}$$

Thus

$$W_r = \frac{K-1}{2} \cdot P \tag{6.15}$$

Substituting W_r in equations (6.13) and (6.14)
we obtain

$$\bar{R} = d+P+\frac{1-q_p}{q_p}(d+P+\frac{K-1}{2}P) \tag{6.16}$$

and

$$\begin{aligned}\bar{n} &= \lambda\left[d+P+\frac{1-q_p}{q_p}(d+P+\frac{K-1}{2}P)\right] \\ &= g\left[d+P+\frac{K-1}{2}P\right]-\lambda\left[\frac{K-1}{2}P\right]\end{aligned} \tag{6.17}$$

It is interesting to notice the similarity between the results obtained by using the phase method introduced by the author and that obtained by Kleinrock[37]. The mean number of packets $\bar{n}$ obtained by two methods are exactly the same. The mean packet delay $\bar{R}$ differs only in

$$q_t = Prob\{previously\ blocked\ packet\ is\ successfully\ transmitted\} \tag{6.18}$$

given by Kleinrock and

$$q_p = Prob\{blocked\ packet\ is\ successfully\ transmitted\} \tag{6.19}$$

given by the author.

The author's method has an advantage over method introduced by Kleinrock[37] in the sense that the author's method is consistent for all the broadcast type communication networks, and it is also consistent with evaluation models used to estimate other components of computer systems such as cpu and i/o etc; while the method introduced by Kleinrock can only evaluate some special algorithms, such as slotted ALOHA and the evaluation can only be done in isolation.

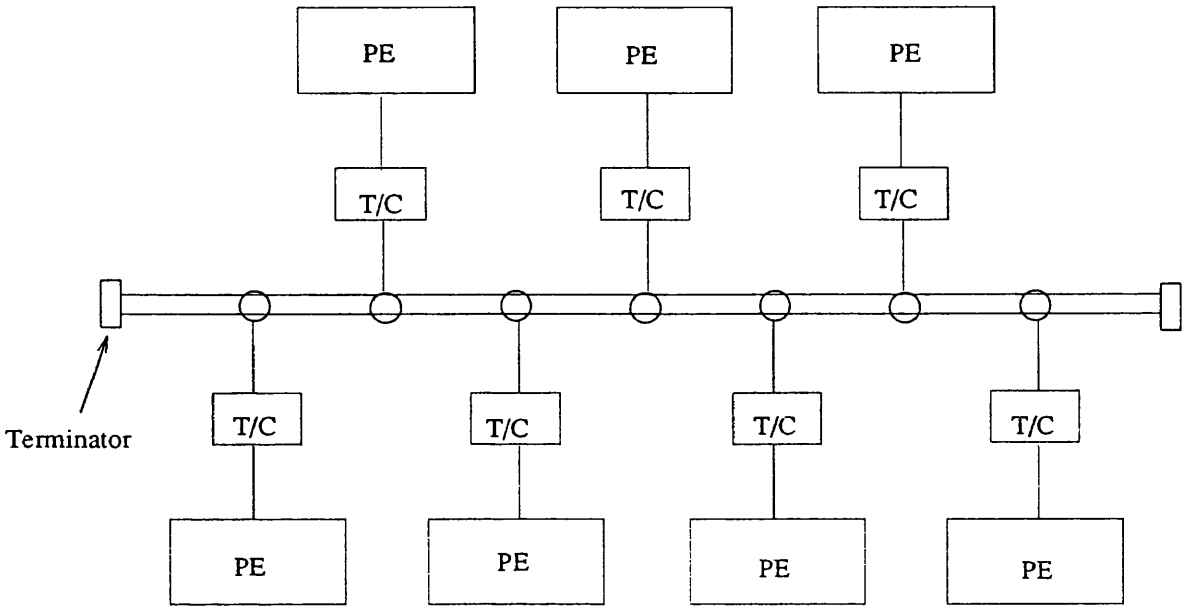
Ethernet

Now we can use the above method to evaluate the performance of Ethernet, shown in figure 6.8. Ethernet is a bus network, which is used to connect a number of processing elements (usually computer, terminal and i/o devices). The transmission medium of the bus channel is usually coaxial cable. Any number of processing element can be connected to the channel by a transceiver.

The characteristics of the bus network are

- broadcast communication;

- baseband transmission mode;
- small propagation delay.



T/C = transceiver

PE = Processing Element

Figure 6.8. Bus Network

The nature of the bus network, especially the characteristics of short propagation delay, determines its access strategy. The bus network employs a carrier sensing technique. The technique called carrier sense multiple access (CSMA) is designed for local area bus networks. A sending user first listens to the channel. If a carrier is sensed, the transmission has to be delayed according to some algorithm; otherwise the packet is transmitted immediately. The transmission is vulnerable during a period at the beginning of the transmission. This period is equal to the one way propagation delay $d/2$, which is much smaller than the transmission delay. Therefore the vulnerable period of the bus network is very short compared with that of ALOHA protocol. There are two types of transmission: slotted and unslotted. The slotted transmission is achieved by synchronizing the channel into slots. A slot time is usually equal to one way propagation delay.

The access control algorithm for the bus network can be classified into two mainly categories: Deference/Acquisition (D/A) and the Collision Detection (CD) algorithms. Since Ethernet emploies the later, we shall develop its evaluation method in details.

Collision Detection CSMA

The collision detection CSMA, also known as Listen-While-Talk (LWT), is used to detect collisions during transmission. As soon as a collision is detected, the transmission is terminated by transmitting a jamming signal to ensure that all other users detect the interference and, therefore abort.

A typical LWT Ethernet is defined by IEEE 802.3 standard with the following parameters

$$\begin{aligned} data\ rate &= 10Mbs \\ slot\ time &= 512bits \\ preamble &= 64bits \\ jam\ signal &= 32—48bits \\ interframe\ spacing\ i.e.\ \alpha &= 9.6—10.6microsecs \end{aligned}$$

A station, wishing to transmit, packs the data into a packet. It first listens to the Ether to sense a carrier. Once the end of the carriers is sensed, it delays for a certain time to allow hardware to settle, i.e. interframe spacing α, and then transmits a preamble followed by the packet. The transmission is subject to collision during a period of α sec., where α is also the one way propagation delay. Once the collision period is passed, a station is said to have acquired the Ether and other stations are deferred for the duration of the transmission. One remarkable feature of Ethernet is its ability to detect collisions while transmitting. Upon collision a station has to wait for a random period of time to retransmit the packet. The retransmission is governed by a binary exponential backoff algorithm. For the nth retransmission, the transmission is delayed by r slots, where r is a uniformly distributed random integer in the range $0<r\leq 2^k$, $k=\min(n, 10)$.

We shall first model the retransmission delay. In order to simplify the modeling, we assume $k=n$. This assumption can be usually justified for non-heavily loaded network, since the retransmission times is usually less than 10. Let us denote W_k as the random delay of the kth retransmission. The probability of a packet being delayed by r slots upon kth retransmission is given by

$$P_k(r) = \begin{cases} \dfrac{1}{2^k} & 1\leq r\leq 2^k \\ 0 & otherwise \end{cases} \tag{6.20}$$

The mean of $P_k(r)$ can be obtained by

$$W_k = \sum_{r=0}^{\infty} rP \cdot P_k(r)$$
$$= \sum_{r=1}^{2^k} \frac{rP}{2^k}$$
$$= \left[\frac{1+2^k}{2}\right] \cdot P \tag{6.21}$$

where P is the transmission time.

LaBarre[40] has derived a throughput equation for nonpersistent LWT CSMA protocol, which is written as

$$\lambda = \frac{ge^{-\alpha gP^2}}{gP(e^{-\alpha gP^2}+\alpha P) + (1+\alpha gP^2)(1-e^{-\alpha gP^2})^2 + 1} \tag{6.22}$$

where $\alpha = d/2$ is the one way propagation delay.

It should be noted that for a CSMA protocol, a deference time should be included. It is the time delay between the start of the sensing and the moment at which a non-busy channel is sensed. Let us denote p_b as the probability of sensed busy. Then gp_b is the number of transmissions being blocked by sensed busy and gp_b/λ is the average number of times over which a packet is blocked. For each sensed busy, a packet waits for a random period W_r. The average deference time of a packet is therefore equal to gp_b/λ multiplied by one random delay W_r. That is

$$\frac{gp_b}{\lambda} W_r = \textit{average deference time of a packet} \tag{6.23}$$

It should be noted that due to the carrier sense nature of the CSMA protocol, the probability of a successful transmission is no longer equal to λ/g. Recalling the pure ALOHA protocol, g is both the number of packets generated per sec and the number of packets attempted for transmission per sec. But in CSMA CD protocol the two are not equal. The number of packet attempted for transmission per sec $\hat{g}$ is equal to the number of packets generated g multiplied by the probability $(1-p_b)$. That is

$$\hat{g} = g(1-p_b) \tag{6.24}$$

where p_b, the probability of being blocked by sensed busy, is given by Tobagi[82]

$$p_b = 1 - \frac{1+\alpha gP^2}{1+gP(1+\alpha P+\bar{Y})} \tag{6.25a}$$

and

$$\bar{Y} = \alpha P - \frac{1}{gP}(1-e^{-\alpha gP^2}) \tag{6.25b}$$

The probability of a successful transmission q_p is therefore given by

$$q_p = \frac{\lambda}{\hat{g}} = \frac{\lambda}{g(1-p_b)} \tag{6.26}$$

Let us consider the average transmission time and the average number of packets in the system. The phase method proposed by the author can be used here. The time spent in phase one equals the sum of the successful transmission time and the deference time. That is

$$S_1 = P+\alpha+\frac{gp_b}{\lambda}W_1 = P+\alpha+\frac{3}{2}P\frac{gp_b}{\lambda} \tag{6.27}$$

For the subsequent transmissions (i=2,3, $\cdots$). The transmission time S_i is composed of the interference delay caused by the last conflicting transmission α, the jamming signal delay β after the transmission, the deference delay to avoid further conflict, the one way propagation delay α to clear the channel, the packet transmission time P and the deference time. The retransmission delay is therefore given by

$$S_i = 2\alpha + \beta + P + \frac{gp_b}{\lambda}W_{i-1}$$

Substituting W_{i-1} into the above equation, it becomes

$$S_i = \sigma + \frac{gp_b}{\lambda}2^{i-2}P$$

where

$$\sigma = 2\alpha + \beta + P + \frac{P}{2}\frac{gp_b}{\lambda}$$

It should be noted that the above retransmission delay W_i changes for each retransmission on Ethernet due to the binary exponential backoff algorithm. We have to extend the phase method to model this more robust retransmission algorithm. The model consists of an infinite number of phases, where phase i has the service time of S_i. A transmission will terminate successfully with probability q_p and continue to retransmit with $1-q_p$. The Laplace-Stieltjes transform of the model is given by

$$f^*(s) = \sum_{i=1}^{\infty}(1-q_p)^{i-1}q_p \prod_{k=1}^{i}\frac{1}{sS_k+1} \tag{6.28}$$

The mean is given by

$$-\frac{d}{ds}f^*(s)\Big]_0 = -\sum_{i=1}^{\infty}(1-q_p)^{i-1}q_p\sum_{k=1}^{i}-S_k$$

$$= \alpha + \frac{1-q_p}{q_p}\sigma + \frac{P}{2q_p-1}\frac{gp_b}{\lambda} \tag{6.29}$$

The results of the above model suggest that the whole Ethernet can be modeled by an infinite-server (IS) service centre with mean service time given by the above equation. The generating rate g can be obtained from equation (6.22) and the probability of being blocked by sensed busy p_b is given by equation (6.25a) and (6.25b).

The average transmission time is therefore given by

$$\bar{R} = \alpha + \frac{1-q_p}{q_p}\sigma + \frac{P}{2q_p-1}\frac{q_p p_b}{\lambda} \tag{6.30}$$

and the average number of packets in the system

$$\bar{n} = \lambda\bar{R} = \alpha\lambda + \frac{1-q_p}{q_p}\sigma\lambda + \frac{q_p p_b P}{2q_p-1} \tag{6.31}$$

It is interesting to notice that the phase method introduced by the author is consistent in all the CSMA protocols studied. It models the bus network in a natural way and has clear physical explanations. The results produced here are consistent with those available by using others' ad hoc methods. The phase method can be used conveniently in queueing networks to model broadcast communication networks.

6.6.2 Analytic Model of the Overall System

As previously introduced, the DRFS is composed of two tiers, the TM-TM tier and the TM-DM tier. The TM-DM tier manages the distributed executions of transactions between TMs and DMs. The modeling of the TM-DM tier for transaction management is of particular interest since it is the core part of the system. The transaction management, as discussed in section 6.4, is controlled by a majority consensus two phase commit protocol, i.e. P4. To model the concurrency control and locking performance of the TM-DM tier, it is necessary to employ the model derived for the general majority consensus 2PL protocol on a packet switching network introduced in section 5.2. However, since DRFS is built on a bus network (Ethernet) rather than a packet switching network, we shall combine the models of Ethernet in section 6.7.1 and the majority consensus 2PL in section 5.2.

Suppose we model a DRFS system in which TMs and DMs do not share workstations. Its communication structure of the DRFS is shown in figure 6.9. Applying the method introduced in section 5.2, the DRFS system can be modeled by an open queueing network with general service time distributions. The equations used to solve the model are from (5.27) to (5.56). However the model requires two modifications. Since TM and DM are not built on the same workstation, a transaction generated at

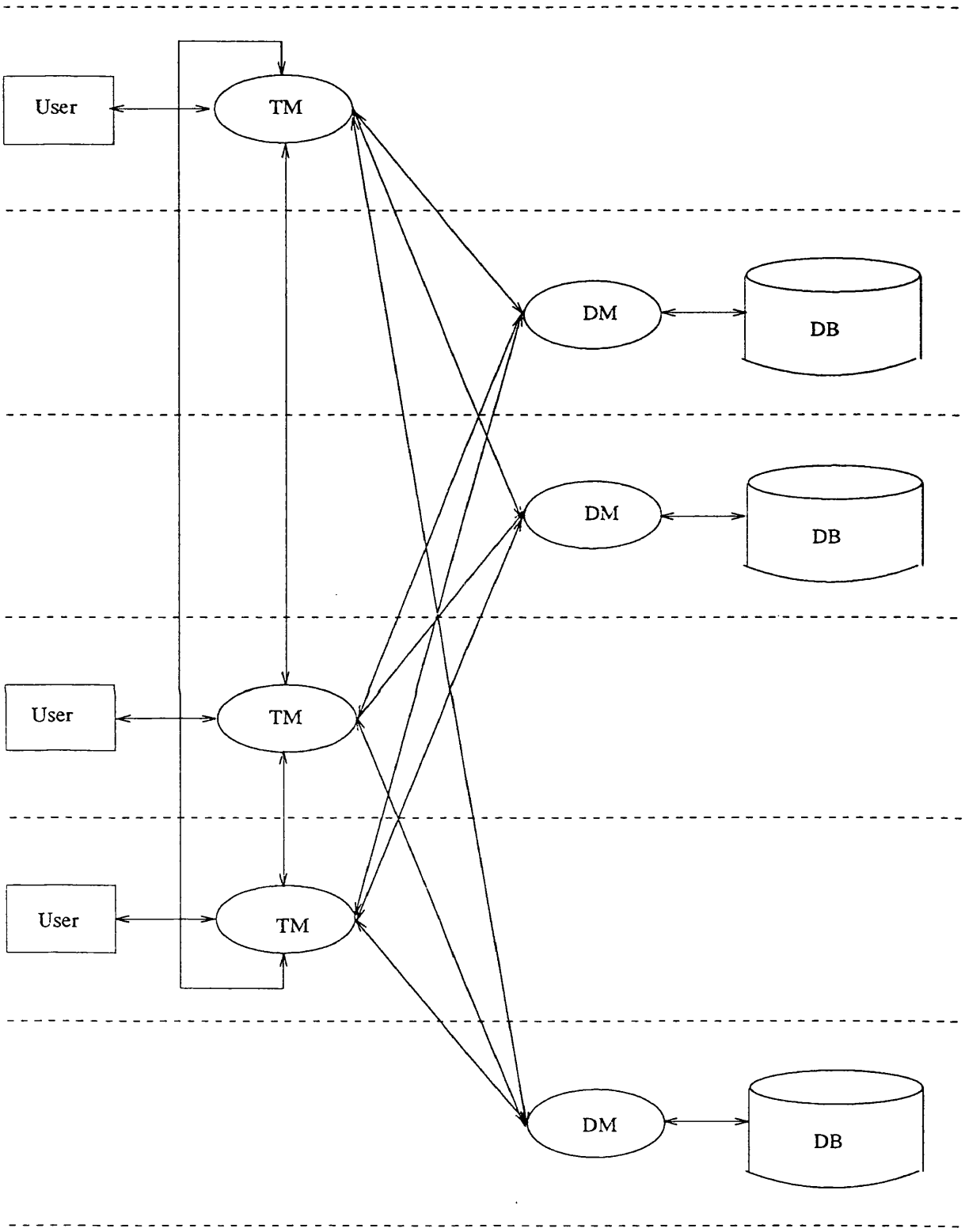


Figure 6.9. The Communication Structure of the DRFS

TM will only access remote DMs. The access pattern matrix is modified by

$$\omega_{ij} = \begin{cases} \frac{1}{N_f} & i \in TM,\ j \in DM \\ 0 & otherwise \end{cases} \tag{6.32}$$

In a packet switch network, the communication channel between station i and k is modeled as a single server service centre; while for a bus network, the whole bus can be modeled as one infinite-server (IS) service center as introduced in section 6.6.1. Since an IS service centre can be decomposed into many IS service centers, for the sake of simplicity we can use an independent IS service centre for each i to k communication, denoted by i_k. According to equation (4.111), the mean number of busy servers of the i to k service centre is given by

$$\bar{N}_{i_k} = \frac{1}{2}\left[C_{i_k} + \frac{2\lambda_{i_k}}{\mu_{i_k}} + 1\right] \tag{6.33}$$

where

$$C_{i_k} = \frac{1}{\lambda_{i_k}} \sum_{j=0}^{m} [(C_j - 1)p_{ji_k} + 1]\lambda_j p_{ji_k} \tag{6.34}$$

is the squared coefficient of the variation of the interarrival time at service centre i_k. The mean service time of the Ethernet IS service centre μ_{i_k} is given by equation (6.29). That is

$$\mu_{i_k} = \alpha + \frac{1-q_p}{q_p}\sigma + \frac{P}{2q_p - 1}\frac{gp_b}{\lambda} \tag{6.35a}$$

Where p_b is given by equations (6.25a) and (6.25b). In the DRFS implementation, the majority consensus 2PL protocol is built on the top of the transport layer protocol UDP/IP. In order to save cpu time of the listening server, the protocol uses a small time-out delay to select incoming messages. Therefore a further delay factor W_p should be added to the service time μ_{i_k}. That is

$$\mu_{i_k} = \alpha + \frac{1-q_p}{q_p}\alpha + \frac{P}{2q_p - 1}\frac{gp_b}{\lambda} + W_p \tag{6.35b}$$

Since all IS service centers share one Ethernet, the overall throughput of the Ethernet is the sum of all the interarrival rates to the network. However the cost of multicasting a message to N_{acc} receivers is the same as unicasting the message. Therefore at each phase of the majority consensus 2PL Ethernet only transmit $1+N_{acc}$ rather than $N_{acc}+N_{acc}$ messages. The overall throughput of the Ethernet is therefore given by

$$\lambda = \frac{1+N_{acc}}{2N_{acc}} \sum_{i=1}^{K}\sum_{k=1}^{K} \lambda_{i_k} \tag{6.36}$$

Substituting λ into equation (6.22) we can obtain the generating rate g from

$$\frac{1+N_{acc}}{2N_{acc}} \sum_{i=1}^{K} \sum_{k=1}^{K} \lambda_{i_k} = \frac{ge^{-\alpha gP^2}}{gP(1+2\alpha P+2\alpha Pe^{-\alpha gP^2}) - (1-e^{-\alpha gP^2})^2 + 1} \quad (6.37)$$

The probability of blocking by sensed busy is given by equation (6.25a) and (6.25b) respectively. The model can be solved iteratively by numerical method.

Previous researches can only evaluate distributed database built on point-to-point network. We have not only introduced a novel metod to evaluate bus netowrk (i.e. Ethernet) but also integrate it naturally with the overall queueing netowrk model; thus our anaylytic model covers all the types of networks.

6.7 Performance Measurements and Comparison of Results

The DRFS system is tested by using performance measurement method. The analytic model of the real system is also built and analytic results are obtained. The comparisons of the measurement and analytic results are shown in figure 6.10 and 6.11.

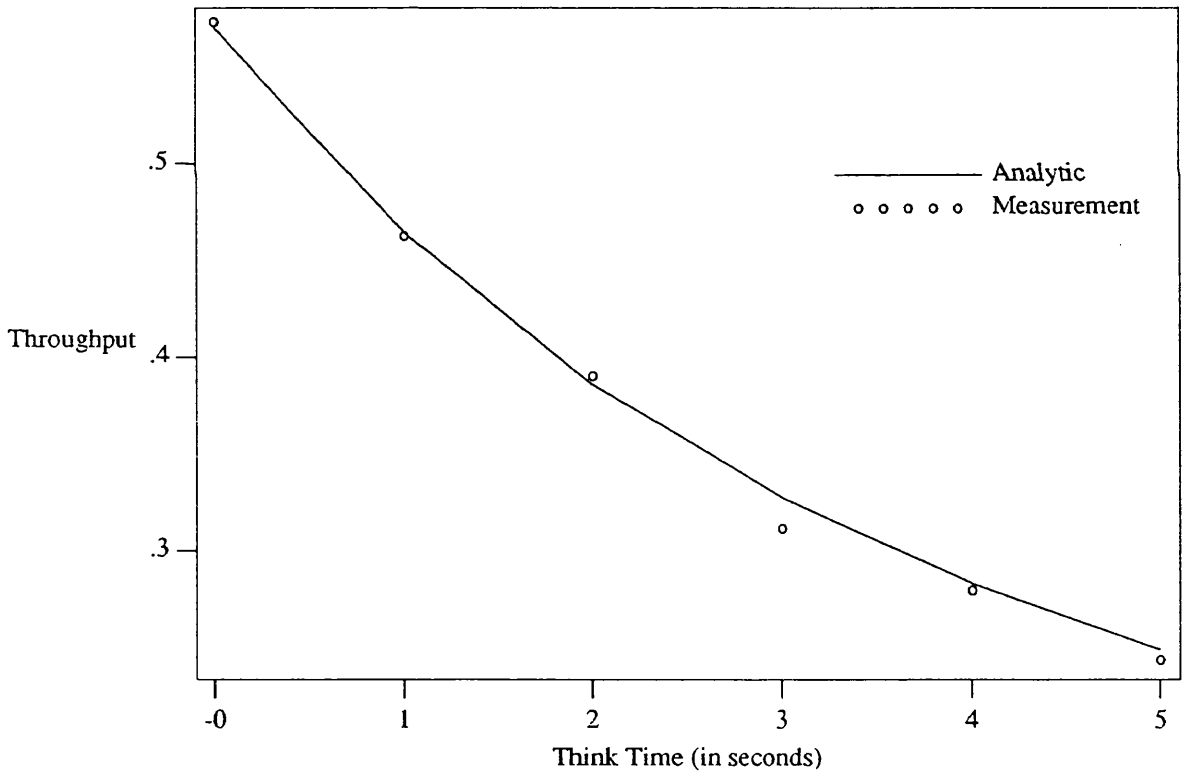


Figure 6.10. Throughput of the DRFS System

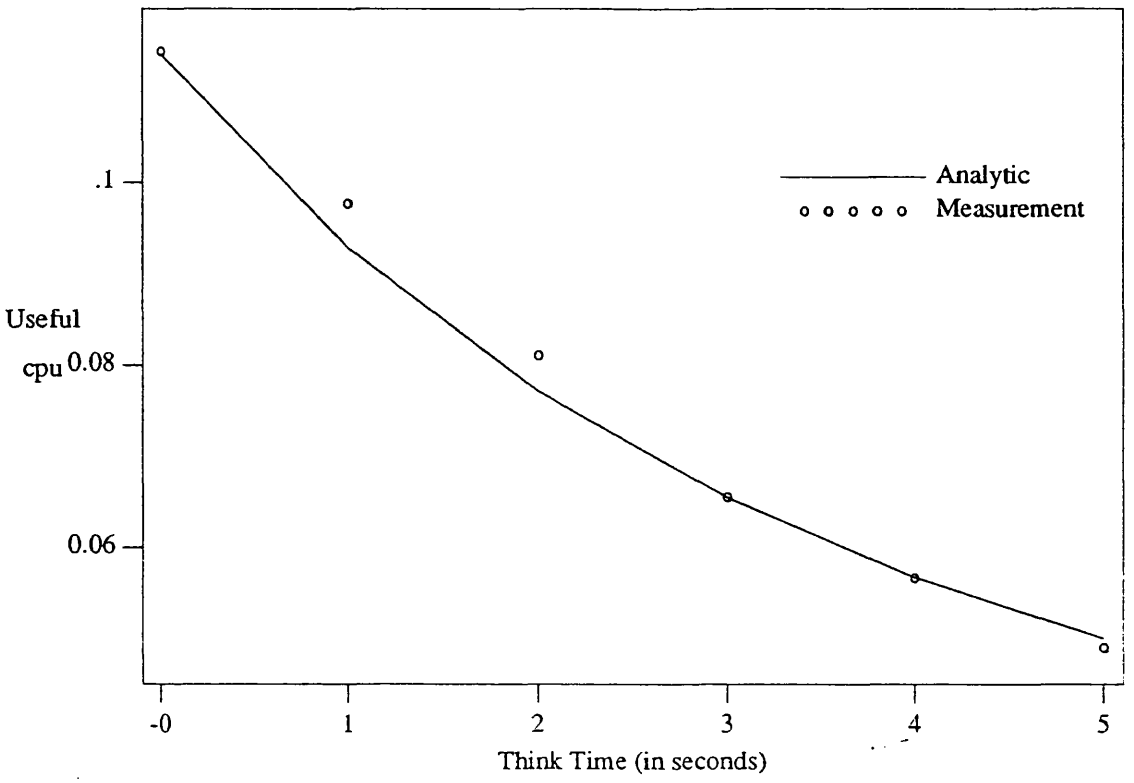


Figure 6.11. Cpu Utilization at a *DM* in the DRFS System

The DRFS being measured is composed of two *TM*s and three *DM*s as shown in figure 6.9. The aim of the measurements is to obtain the mean throughputs and system utilization of the DRFS. The files stored in three DMs contains 100 records, and each record owns a lock. At each *TM*, a large number of transactions are generated by a test program. Between two successive transactions a wait is imposed to simulate the think time. By varying the think time from 0 sec to 5 secs, a group of mean measurement results are obtained. The throughput of the DRFS is measured by counting the number of the transactions completed during the test period. The useful cpu time is obtained by running the "time" command to generate a profile of the DM server process.

In order to obtain the analytic results of the tested DRFS, several measurement tests have been run to obtain the mean and variance of the service times of locking, reading and writting records, and protocol delay. These measured parameters are then used in the analytic model of the DRFS. The results are illustrated in figure 6.10 and 6.11 along with the measurement results.

The analytic model of the DRFS represents very detailed characteristics of the system, such as the concurrency control algorithm, database structure, Ethernet protocol algorithm, data granularity, degree of data replication, service time distributions of the cpu and i/o devices, etc. It aims to give an accurate evaluation of the real system and provide validation for the analytic model. These two goals are achieved as indicated by the close agreement between the results. Furthermore, from this validation we have presented the real physical interpretations of the analytic model and gained many useful experiences on

how to apply performance evaluation theory into real systems.

The comparison of the two results indicates a satisfactory agreement between the analytic model and the real system. Therefore the analytic model is validated by not only the simulation results but also more convincingly an actual distributed database. This kind of emparical valoidation has not been done before, because it is not a simple task to implement an actual distributed database and it is also difficult to include detailed characteristics of the system in the analytic model. We have not only managed these two difficult tasks, but also achieved a high degree of accuracy in emparical validation.

6.8 Summary

In this chapter, an actual distributed database management system is implemented. The system uses basic 2PL and majority consensus 2PL algorithms and it furhter employs many novel features such as multicast communication premitives, dynamic TM configuration, failure recovery etc. Novel analytic model has been introduced to evaluate bus-network and its integration with the overall model of the distributed database. Results from empirical evaluation agrees with the anlytic results at a high degree of accuracy; thus adding further validation support to the theory.

Chapter 7. Conclusions

7.1 Conclusions

In this thesis, consistent and systematic methods have been used to study the performance of concurrency control protocols in both centralized and distributed databases by using analytic modeling, simulation and measurement. The models have captured many important features, such as transaction blocking, priority locking, multiple transaction classes, concurrency control algorithms, collision resolution algorithms, service time distributions, locking granularity, data replication, network topology and network protocols.

Original contributions have been made in the thesis to develop and extend the modeling techniques throughout all the levels. We can define the basic level as a fundamental representation of the architecture and concurrency control algorithms in a distributed database system; the first order level as the factors which have a first order effect to the performance of a distributed systems, such as the mean measurements and queueing disciplines; the second order level as the additional factors which only have a second order effect on the performance, such as the variations and distribution functions.

At the basic level the introduction of the communication flow rate matrix, access pattern matrix, and Markov chain matrix has brought a valuable step toward the representation of the fundamental architecture of the concurrency control algorithms and the underlying network topology of a distributed database in a consistent and systematic manner.

At the first order level, novel methods have been introduced to model waiting in blocked queue, priority queueing and multiclass transactions. Many important factors such as the degree of data replication, degree of the locality, read-write ratio and locking granularity are modeled to achieve a high degree of accuracy.

At the second order level evaluation methods have been introduced in the thesis to model distributed databases with non-Poisson interarrival distributions and general service time distributions. The second order factor, i.e. the variation of a random variable in the queueing network is represented by applying extended diffusion approximation method. Moreover the service time distributions of various database operations and data transmissions are given by statistical analysis and measurements.

The analytic models proposed in this thesis are validated by simulation results obtained by both the author and Ries and Stonebraker. The comparison of results shows a very close agreement between the two. The comparison of results obtained from the analytic model and from the actual distributed database system implemented by the author also indicate a close agreement between the two. The research introduced in the thesis provides a significant step forward in performance eveluation of distributed

databases. The main contributions made in the thesis are as follows.

One of the most difficult problems associated with database modeling is to evaluate the waiting time of blocked transactions. In this thesis we have introduced a novel method to solve the problem. An analytic model is used to represent a centralized database as a network of queues. In particular, the blocked transactions in the waiting queue are modeled by an infinite number of infinite-server service centres, each of which represents a waiting phase of the blocked transactions. The stochastic nature of blocking and collision resolution algorithms is therefore accurately modeled. The overall model of the centralized database with blocking is represented by an open system composed of $M/G/1$ and IS queues with two priority levels. The technical methods used to solve the analytic model are developed in the thesis. Firstly, since locking operation has priority over transaction execution operation, the method to model two priority levels has been introduced. Secondly, the distribution function of the departure process of the transaction execution jobs, i.e. second priority class jobs, is derived to model the waiting phase of the blocked transactions. Thirdly, the transactions with non-exponential service time distributions are modeled by including the second moment of the service time distribution in the model.

The open model of the centralized database is further extended to the closed model by introducing an approximation method, from which a very useful approximation result has been obtained: under the condition of small lock overhead, the departure process of the transaction execution job is approximately Poisson and its mean depends only on the overall throughput of the closed database.

Further numerical technique has been introduced to model the multiclass transactions in the blocked queue. The service demands, lock conflict rates and lock sizes of different classes of transactions are represented. An interesting phenomenon, that the transactions with smaller service time and lock size will circulate in the system more quickly than others, has been noticed.

In order to validate our analytic model of centralized database, we have used Ries and Stonebraker's simulation model and results. Their model simulates the centralized database at a very detailed level, such as transaction blocking phenomenon, collision resolution algorithms, locking and transaction operations, lock script and multiclass transactions. By comparing our analytic results with their simulation results, we obtain a close agreement between them.

A novel method is introduced to systematically define a distributed database system with two phase commit concurrency control protocol. A distributed database is composed of network service centres, computer service centres and locking service centres. The data flow between these service centres depends on the concurrency control protocol, data replication, lock granularity, read-write ratio, network topology etc. In this thesis, we first introduce the concept of access pattern matrix and communication flow matrix, which represents the flow of data between service centres. The arrival rate matrix is further derived to define the interarrival rates of transactions to each service centre and finally the Markov chain matrix can be derived from the communication flow matrix to define the overall queueing network. The method very

much simplified the complicated and tedious task of manually obtaining the Markov chain matrix of a distributed database. And it is also proven to be very useful in developing the distributed locking model in section 4.3.1.

Transaction blocking in distributed databases has not been analytically modeled before. In this thesis we have first introduced a phase method to model distributed locking with fixed waiting. The model can represent each lock operation, the stages of locking and the stochastic nature of locking. The method can be easily applied to model various different types of locking algorithms.

Previous researches only model distributed databases with exponential service time distributions. In order to model general service time distribution and non-Poisson interarrival time, we have adopted and further extended the diffusion approximation method. In the thesis new equations have been derived to model infinite-server service centres. Furthermore the extended diffusion approximation method is also used to model distributed locking. The overall distributed database model is constructed by an open queueing network with general service time distributions. The method can be extended to model various distributed database systems and the numerical solution can be obtained with good convergence.

The overall distributed database model has captured most of the features of the system in a depth which has not been reached before. It includes the communication flow of the concurrency control protocol, locking algorithm, collision resolution algorithm, data replication, lock granularity, read-write ratio, priority locking, service time distribution and interarrival time distribution. The model therefore can represent the real system much more accurately.

The model for distributed database is also consistent and easy to apply to various concurrency control protocols. The same model is used to model four different 2PL protocols. They are basic 2PL, primary copy 2PL, majority consensus 2PL and centralized 2PL. The formal definition method, i.e. the access pattern matrix, communication flow matrix, etc., is proven to be particularly useful in defining these models. It makes the model definition task much easier and efficient. Furthermore since the same modeling method is applied to these distributed database systems, the comparison of results are fairer. The method can, in general, be easily applied to various other distributed systems with different concurrency control protocols.

A simulation model has been built to validate the analytic model of distributed databases. The simulation model represents most of the characteristics of a distributed database. It also simulates various service time distributions to validate the extended diffusion approximation approach. The comparison of analytic results and simulation results clearly shows that the extended diffusion approximation model can represent the system much more accurately than conventional queueing network method. The overall analytic model achieves a high degree of accuracy in comparison with the simulation model.

In order to further validate the analytic method and show the real application of the modeling, we have implemented an actual distributed database system. The prototype system is built on several Sun

workstations over Ethernet, a bus network. One of the main features of the system is multicast communication. Other features include basic 2PL and majority consensus 2PL protocols, dynamic TM configuration and failure recovery. The prototype system is tested and measured.

In order to model the actual distributed database with multicast communication, an analytic model is introduced to evaluate Ethernet. The method can not only evaluate isolated Ethernet but also integrate it with the rest of the modeling method introduced in the thesis. Thus an overall analytic model for the prototype distributed database with multicast is built. It represents detailed characteristics of the system, such as concurrency control protocol, database structure, multicast protocol, Ethernet structure, data granularity, data replication, service time distribution, etc.

The comparison of the analytic results and the real system measurement results shows a close agreement. The analytic method is therefore further validated by real system measurement. Moreover the validation also presents the real physical meaning of the analytic model.

The comparison of various concurrency control protocols shows that, in general, centralized 2PL outperforms basic, primary copy and majority consensus 2PL. It is interesting to notice that the locking conflict rate is strongly associated with the distribution of the locking activities. The wider the locking activities are spread, the higher the conflict rates are. Another interesting finding is that the degree of data replication has a big impact on the response time. By properly choosing the parameter, optimal performance can be obtained. The finding of lock granularity being less effective to the overall performance is not surprising, since the lock overhead in the test system is quite small. These performance results are useful in comparing various concurrency control algorithms and designing distributed database systems.

In conclusion, many original methods have been introduced in the thesis to model concurrency control and locking in both centralized and distributed databases in greater depth than before. All the analytic methods introduced in the thesis are validated by either author's simulation model, others' simulation model or real system measurement. Various concurrency control protocols have been modeled and analyzed and many interesting results have been obtained in the thesis.

7.2 Future Directions

Research could be carried further in the following directions. Firstly, in order to further improve the accuracy of performance evaluation, there is a requirement to represent the functionality of the concurrency control in more details, such as that achieved by simulation. Secondly the assumption of no site or network failure can be released by introducing the probability of failures in the model. Thirdly the effect of work load is not modeled in the thesis. However it is not difficult to include it in the queueing network model in a consistent way. Fourthly the modeling of both exclusive and shared lock in a distributed system requires further attention. New probability descriptives such as the percentage of shared lock and exclusive lock can be introduced in the analytic model.

Appendix A. List of Notions

α	one way propagation delay
β	jamming signal delay
$\beta^{(i)}(s)$	Laplace-Stieltjes transform of the service time distribution of the ith priority level
γ_{ik}	mean message rate for those originated at node i and terminated at node k
γ_{rw_k}	read-write ratio of a transaction generated at TM_k
δ	inter-track movement time
$\theta^{(2)}$	residual interarrival time of the second priority customer
λ	mean interarrival rate
λ_0	mean interarrival rate at the source
λ_b	mean interarrival rate of a blocking service centre
$\lambda_{cpu}^{(i)}$	mean interarrival rate of the ith priority customer at a cpu service centre
$\lambda_{io}^{(i)}$	mean interarrival rate of the ith priority customer at an i/o service centre
$\lambda_{ij}^{(c)}$	mean interarrival rate of communication channel from node i to node j
$\lambda_k^{(s)}$	mean interarrival rate of node k at stage s
$\lambda^{(i)}$	mean interarrival rate of the ith priority customers
Λ_c	communication flow rate matrix
Λ_i	throughput of service centre i
Λ_s	arrival rate matrix at stage s
μ	mean service rate of a job
$\mu(n)$	aggregated service rate of the inner model of a computer system at the mutiprogramming level of n
μ_σ	mean interdeparture time of a transaction execution centre

μ_{σ_ζ}	mean residual interdeparture time of a transaction execution centre
μ_a	mean interarrival rate
μ_{ag}	aggregated service rate of a locking service centre with fixed waiting
μ_b	mean service rate of a blocked queue with fixed waiting
μ_{lock}	mean service rate of lock request
μ_{k_i}	mean service rate of a class i transaction in service centre k
μ_s	mean service rate of requesting one lock
$\mu^{(i)}$	mean service rate of the ith priority customer
$\mu^{(i)}(s, 1)$	Laplace-Stieltjes transform of the busy period of the ith priority level
$\nu_\tau^{(1)}$	initial waiting time of residual busy period of the first class customer
$\xi_n^{(i)}$	number of ith priority customers in the system prior to the arrival of the nth customer
Π_i	stationary distribution of i customers in the system
Π_{ik}	path between node i and k
ρ	utilization
σ	second moment of the service time
σ_a	standard deviation of interarrival time
σ_b	standard deviation of service time
σ_a^2	variance of interarrival time
σ_s^2	variance of service time
$\sigma^{(i)}$	second moment of the ith priority customer
$\sigma^*(s)$	Laplace-Stieltjes transform of the service time distribution of one locking operation
τ_n	epoch of the nth arriving customer
ω_{ij}	probability of a transaction accessing node j from node i

a	overall system utilization
$a^{(i)}$	utilization at the ith priority level
$a_{ij}^{(s)}$	communication flow rate from node i to j at stage s
$A(t)$	cumulative number of arrivals up to time t
$\mathbf{A}_i$	communication flow matrix between stage i and $i+1$
$b(t)$	service time density function
$b^{(1)}$	busy period of the first priority customer
$B(t)$	service time distribution function
Blk	length of a block
c	number of transaction classes
c_a	coefficient of interarrival time
c_i	capacity of the ith communication channel
c_i	completion time of a second priority customer
c_s	coefficient of service time
C_i	squared coefficient of the variation of the interarrival time at service centre i
$C_{cpuj}^{(i)}$	service capacity of the class j customer at the ith priority level at the cpu service centre
$C_{ioj}^{(i)}$	service capacity of the class j customer at the ith priority level at the i/o service centre
d	propagation time of a packet transmission
$D(t)$	cumulative number of departures up to time t
$D_{cpuj}^{(i)}$	service demand of the class j customer at the ith priority level at the cpu service centre
$D_{ioj}^{(i)}$	service demand of the class j customer at the ith priority level at the i/o service centre
$dm-read(X)$	$Read(X)$ performed on a physical database by a DM

$dm-write(X)$	$Write(X)$ performed on a physical database by a DM
DM_k	data manager at node k
e_i	relative throughput of service centre i
$f^*_{lock}(s)$	Laplace-Stieltjes transform of the service time distribution of locking operations to grant r locks
f^*	Laplace transform of function f
$f^*_b(s)$	Laplance-Stieltjes transform of the service time distribution of the waiting centre
$f(x,t)$	probability of x jobs at time t
$File$	length of a file
g	packet generation rate (i.e. number of packets generated per second)
$\hat{g}$	attempted packet transmission rate
$g^*_\zeta(s)$	residual life time of the density function of the interdeparture process of the second priority customer
$g^*(s)$	Laplace-Stieltjes transform of the density function of the interdeparture process of the second priority customer
G	normalizing factor of an open queueing network
$G(n)$	normalizing constant when the number of customers in the system is n
$G(t)$	distribution function of interarrival time
$\hat{G}(t)$	approximate service time distribution of the interdeparture distribution of the second priority customer
$G^*(s)$	Laplace transform of $G(t)$
Gap	length of a gap between blocks
$H^{(i)}(t)$	service time distribution function of the i priority customer
K	number of nodes in a distributed database
K_i	squared coefficient of the variation of the service time
l	number of levels of an index

l_i	message length of the ith communication channel
$\underline{l_i}$	vector of a 1 in the ith component and zeros in the rest
L	total number of locks in the database
L_k	total number of locks in the database at node k
$lock(X)$	apply lock on data item X
m	number of service centres
n_a	average number of accesses for each record
n_i	number of customers at service centre i
n_p	mean number of paging activities of a customer
n_r	number of records in a file
n_r'	number of indexable records
$\overline{n}_i$	mean queueing length of service centre i
$\overline{n}_{k_i}$	mean queueing length of the ith communication channel at node k
$\overline{n}_{k_i}^{(s)}$	mean queueing length of the ith communication channel at node k at stage s
$\overline{n}_{k_i}$	mean queueing length of class i transaction in service centre k
$\overline{n}^{(i)}$	mean queueing length of the ith priority customer
N	number of customers in the system
N_{acc_k}	mean number of database accesses to DM generated by one transaction at a TM
N_f	mean number of replicated data copies
N_g	total number of granules in the database
N_{li}	mean number of locks held by class i transactions
$\overline{N}$	mean number of busy servers in an infinite-server service centre
$\underline{N}$	vector of the total number of transactions in each class
$NL_k^{(com\,2)}$	mean number of locks held at the communication channel in stage 2 at node k

$NL_k^{(com\,3)}$	mean number of locks held at the communication channel in stage 3 at node k
$NL_k^{(lock)}$	mean number of locks held at the lock request centre at node k
$\hat{p}(n)$	diffusion approximation to the probability that the queueing length is n
p_b	probability of sensed busy
p_{fb}	probability of feedback
p_{ij}	probability of a customer moving from centre i to j
P	packet transmission period
$P(n)$	probability of the state of n customers
$P(n_0, n_1, ..., n_k)$	steady-stage probability of a network state with $k+1$ service centres
$P_i(n_i)$	factor corresponding to the steady-state probability of the state of service centre i in isolation
$P_i(t)$	probability of i transmissions during time t
$Prob\{Access_{ik}\}$	probability of accessing node k from node i
$Prob\{Read\}$	probability of a transaction being of read type
$Prob\{Write\}$	probability of a transaction being of write type
q	probability of successfully granted one lock
q_b	probability of successfully granted r locks
q_k	probability of successfully granted one lock at DM_k
q_p	probability of successful transmission of a packet
q_t	probability of previously blocked packet being successfully transmitted
$Q(t)$	number of customers at time t
r	average number of locks held by one transaction
$\bar{r}$	mean number of locks required by one transaction in all transaction classes
rl	rotational lantency
rpm	number of disk revolution per minute

R_{k_i}	mean response time of class i transaction in service centre k
$\overline{R}_i$	mean response time of service centre i
$Read(X)$	transaction of reading logical data item X into database
Rec	length of a record
s_c	initial startup time of an i/o device
S_i	time spent in the ith phase of transmission
S_n	service time of the nth customer
sk	seek time of a disk
t_B	block transfer time
t_F	record fetch time
t_R	record transfer time
t'	bulk transfer rate
T	raw transfer ratio of an i/o device (in chapter 2)
T_a	average turn-around time of a closed queueing network
TM_k	transaction manager at node k
$U_{cpuj}^{(i)}$	utilization of the class j customer at the ith priority level at the cpu service centre
U_i	utilization of service centre i
$U_{ioj}^{(i)}$	utilization of the class j customer at the ith priority level at the i/o service centre
U_{k_i}	utilization of class i transaction in service centre k
w_i	pre-waiting time of the second class customer
$\mathbf{W}$	access pattern matrix
W_a	waste of a disk
W_k	random delay of the kth retransmission
W_p	time out delay at transport layer
W_r	random waiting to avoid further collision

$W^{(i)}$	mean waiting time of the ith priority customer
$Write\,(X)$	transaction of writing logical data item X into database
$\overline{x}$	mean service time
x_i	physical data item at node i
X	logical data item
y	number of index records in a block
y_{eff}	effective number of index records in a block

Appendix B. Theorem Proof

The theorem proofs of chapter 3 are included.

Theorem 3.1: The process $\{\chi_t^{(2)}, t\in[0,\infty)\}$ which is regenerative with respects to the renewal precess $v_t, t\in[0,\infty)$ is Markovian with a standard transition matrix $\pi_{ij}(t)$, $i,j\in S$ such that for all $i,j\in S$,

$$(i)\ p_{ij}(t)=\sum_{k\in S} r_{ik}\pi_{kj}(t),\quad t\geq 0, \tag{B.1a}$$

(ii) for $n=0,1,...;v>u\geq 0$,

$$Pr\{\chi_{t_h}^{(2)}=j_h, u<t_h\leq v, h=1,...,m \mid S_{n+1}=v, S_n=u, \chi_{S_n}^{(2)}=i\}$$

$$=p_{ij_1}(t_1-u)\pi_{j_1j_2}(t_2-t_1)\cdots\pi_{j_{m-1}j_m}(t_m-t_{m-1}), \tag{B.1b}$$

for $j_1,...,j_m,i\in S$, with $u<t_1\leq t_2\leq\cdots\leq t_m\leq v$, and $m=2,3,...$

Proof: Within the time of two successive departures of the second priority customers there is no decrease of $\chi_t^{(2)}$, while the increase of $\chi_t^{(2)}$ follows a Poisson process with mean $\lambda^{(2)}$. Therefore there exists a standard transition matrix $\pi_{ij}(t)$, $i,j\in S$, such that for all $i,j\in S$

$$\sum_{k\in S} r_{ik}\pi_{kj}(t)=p_{ij}(t)$$

where

$$p_{ij}(t-u)=\begin{cases} Pr\{\chi_t^{(2)}=j \mid \chi_u^{(2)}=i, S_n=u, S_{n+1}=v\}, & 0<u\leq t\leq v \\ 0, & t<u \end{cases} \tag{B.2}$$

without losting generality, we assume that $u=0$, therefore

$$p_{ij}(t)=\begin{cases} Pr\{\chi_t^{(2)}=j \mid \chi_0^{(2)}=i, S_n=0, S_{n+1}=v\}, & 0<t\leq v \\ 0, & t<0 \end{cases}$$

It should be noted that between two successive departures of the second priority customers $S_n=u\leq t\leq v=S_{n+1}$ the number of arriving seoond priority customers forms a Poisson process. Therefore during $t-u$ time interval

$$p_{ij}(t-u) = \frac{(\lambda^{(2)}(t-u))^{j-i}}{(j-i)!} e^{-\lambda^{(2)}(t-u)} \tag{B.3}$$

It can be rewritten as

$$p_{ij}(t) = \frac{(\lambda^{(2)}t)^{j-i}}{(j-i)!} e^{-\lambda^{(2)}t} \tag{B.4}$$

This is obviously a pure birth process with transition matrix given by

$$\pi_{ij}(t) = \frac{(\lambda^{(2)}t)^{j-i}}{(j-i)!} e^{-\lambda^{(2)}t} \tag{B.5}$$

and

$$r_{ij} = \lim_{t\to 0} p_{ij}(t) = \begin{cases} 1, & \text{if } j=i \\ 0, & \text{if } j\neq i \end{cases} \tag{B.6}$$

Hence

$$\begin{aligned} \sum_{k\in S} r_{ik}\pi_{kj}(t) &= r_{ii}\pi_{ij}(t) \\ &= \pi_{ij}(t) \\ &= p_{ij}(t) \end{aligned}$$

which satisfies the first condition of a regenerative process being a Markovian.

From the definition of $p_{ij}(t)$ we can immediately obtain the right hand side of the second condition.

$$\begin{aligned} p_{ij_1}(t_1-u)\pi_{j_1j_2}(t_2-t_1)\cdots\pi_{j_{m-1}j_m}(t_m-t_{m-1}) &= \frac{[\lambda^{(2)}(t_1-u)]^{(j_1-i)}}{(j_1-i)!} e^{-\lambda^{(2)}(j_1-i)} \frac{[\lambda^{(2)}(t_2-t_1)]^{(j_2-j_1)}}{(j_2-j_1)!} e^{-\lambda^{(2)}(j_2-j_1)} \cdot \\ &\quad \frac{[\lambda^{(2)}(t_m-t_{m-1})]^{(j_m-j_{m-1})}}{(j_m-j_{m-1})!} e^{-\lambda^{(2)}(j_m-j_{m-1})} \\ &= \frac{[\lambda^{(2)}(t_m-u)]^{(j_m-i)}}{(j_m-i)!} e^{-\lambda^{(2)}(t_m-u)} \end{aligned}$$

In order to obtain the left hand side of the second condition, we use the Markov property of the inter-arrival process

$$
\begin{aligned}
&Pr\{\chi^{(2)}_{t_h}=j_h, u<t_h\le v, h=1,\ldots,m \mid S_{n+1}=v, S_n=u, \chi^{(2)}_{S_n}=i\} \\
&\quad = Pr\{\chi^{(2)}_{t_m}=j_m, t_{m-1}<t_m\le v \mid \chi^{(2)}_{t_h}=j_h, u<t_h\le t_m, h=1,\ldots,m-1, S_{n+1}=v, S_n=u, \chi^{(2)}_{S_n}=i\}\cdot \\
&\qquad Pr\{\chi^{(2)}_{t_h}=j_h, u<t_h\le t_m, h=1,\ldots,m-1 \mid S_{n+1}=v, S_n=u, \chi^{(2)}_{S_n}=i\} \\
&\quad = Pr\{\chi^{(2)}_{t_m}=j_m, t_{m-1}<t_m\le v \mid \chi^{(2)}_{t_{m-1}}=j_{m-1}\}\cdot \\
&\qquad Pr\{\chi^{(2)}_{t_h}=j_h, u<t_h\le t_m, h=1,\ldots,m-1 \mid S_{n+1}=v, S_n=u, \chi^{(2)}_{S_n}=i\} \\
&\quad = Pr\{\chi^{(2)}_{t_m}=j_m, t_{m-1}<t_m\le v \mid \chi^{(2)}_{t_{m-1}}=j_{m-1}\}\cdot \\
&\qquad Pr\{\chi^{(2)}_{t_{m-1}}=j_{m-1}, t_{m-2}<t_{m-1}\le t_m \mid \chi^{(2)}_{t_{m-2}}=j_{m-2}\}\cdot \\
&\qquad \ldots \\
&\qquad Pr\{\chi^{(2)}_{t_1}=j_1, u<t_1\le t_2 \mid S_{n+1}=v, S_n=u, \chi^{(2)}_{S_n}=i\} \\
&\quad = \frac{[\lambda^{(2)}(t_1-u)]^{(j_1-i)}}{(j_1-i)!}e^{-\lambda^{(2)}(j_1-i)}\frac{[\lambda^{(2)}(t_2-t_1)]^{(j_2-j_1)}}{(j_2-j_1)!}e^{-\lambda^{(2)}(j_2-j_1)}\cdots \\
&\qquad \frac{[\lambda^{(2)}(t_m-t_{m-1})]^{(j_m-j_{m-1})}}{(j_m-j_{m-1})!}e^{-\lambda^{(2)}(j_m-j_{m-1})} \\
&\quad = \frac{[\lambda^{(2)}(t_m-u)]^{(j_m-i)}}{(j_m-i)!}e^{-\lambda^{(2)}(t_m-u)}
\end{aligned}
$$

Hence the second condition of the theorem is also satisfied [].

Theorem 3.2: The Laplace-Stieltjes transform of the completion time c_i of a second priority customer with preemptive resume priority discipline is given by

$$
\begin{aligned}
E\{e^{-sc_i}\} &= \sum_{k=0}^{\infty}\int_0^{\infty} e^{-st}\frac{(\lambda^{(1)}t)^k}{k!}e^{-\lambda^{(1)}t}dH^{(2)}(t)\{\mu^{(1)}(s,1)\}^k \\
&= \beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\} \quad , \qquad \mathrm{Re}\, s\ge 0 \ , \qquad \text{(B.7)}
\end{aligned}
$$

with

$$
\beta^{(2)}(s)=\int_0^{\infty} e^{-st}dH^{(2)}(t) \qquad \text{(B.8)}
$$

and $\mu^{(1)}(s, 1)$ defined as the Laplace-Stieltjes transform of the busy period of the first priority level. $\mu^{(1)}(s, 1)$ is the zero with the smallest absolute value of

$$
z=\beta^{(1)}\{s+(1-z)\lambda^{(1)}\}\ , \quad \mathrm{Re}\ s\ge 0 \qquad \text{(B.9)}
$$

with

$$
\beta^{(1)}(s)=\int_0^{\infty} e^{-st}dH^{(1)}(t) \qquad \text{(B.10)}
$$

Proof: The probability of k interruptions from the first priority customers during a service time t of a second priority customer is given by

$$\frac{\{\lambda^{(1)}t\}^k}{k!}e^{-\lambda^{(1)}t}$$

Let $B^{(1)}(t)$ be the distribution function of the busy period of first priority customer. The total completion time of a second priority customer with k interruptions can be easily written as

$$H^{(2)}(t)*(B^{(1)}(t))^{k*} ,$$

where $(B^{(1)}(t))^{k*}$ is the k-fold convolution of the busy period with itself. The Laplace-Stieltjes transform of the above equation is given by

$$\int_0^\infty e^{-st}dH^{(2)}(t)*(B^{(1)}(t))^{k*} = \int_0^\infty e^{-st}dH^{(2)}(t)\cdot\{\mu^{(1)}(s,1)\}^k \tag{B.11}$$

Since the interruption time and service time are independent to each other it follows that for the completion time of second priority customer we have

$$\begin{aligned}
E\{e^{-sc_i}\} &= \sum_{k=0}^{\infty}\int_0^\infty e^{-st}\frac{\{\lambda^{(1)}t\}^k}{k!}e^{-\lambda^{(1)}t}dH^{(2)}(t)\{\mu^{(1)}(s,1)\}^k \\
&= \int_0^\infty e^{-(s+\lambda^{(1)})t}\sum_{k=0}^{\infty}\frac{\{\lambda^{(1)}t\}^k}{k!}\{\mu^{(1)}(s,1)\}^k dH^{(2)}(t) \\
&= \int_0^\infty e^{-(s+\lambda^{(1)})t}e^{\lambda^{(1)}\mu^{(1)}(s,1)t}dH^{(2)}(t) \\
&= \int_0^\infty e^{-\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\}t}dH^{(2)}(t) \\
&= \beta^{(2)}\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\}
\end{aligned}$$

Theorem 3.3: The Laplace-Stieltjes transform of the pre-waiting time w_n of a second priority customer is given by

$$E\{e^{-sw_n}\} = \frac{\lambda^{(2)}}{s+\lambda^{(2)}}\cdot\frac{\lambda^{(2)}-s+\lambda^{(1)}[\mu^{(1)}(s,1)-\mu^{(1)}(\lambda^{(2)},1)]}{[\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)](1-s/\lambda^{(2)})} \tag{B.12}$$

Proof: We note that the moments at which a second priority customer leaves system is the regeneration points of the $\chi_t^{(2)}$ process, because at such moments no first priority customer are present at the system. If the nth second priority customer leaves the system empty, the pre-waiting time of the $(n+1)$th customer, i.e. the time between this moment and the moment at which the server becomes available to the $(n+1)$th arriving second priority customer is the sum of the inter-arrival time of the $(n+1)$th second priority customer, denoted by $\theta_{n+1}^{(2)}$, and the duration of a residual busy period of the $\chi_t^{(1)}$ - system. This residual

busy period $b_n^{(1)}$ starts with a initial waiting time $\upsilon_\tau^{(1)}$ which is equal to the virtual waiting time of the $\chi_t^{(1)}$ - system at the moment τ at which the second priority customer arrives. The interarrival time of the $(n+1)$the second priority customer is given by

$$\theta_{n+1}^{(2)} = \frac{\lambda^{(2)}}{s+\lambda^{(2)}} \tag{B.13}$$

It is due to Cohen that[15]

$$E\{e^{-sb_n^{(1)}}\} = \int_{t=0}^{\infty}\int_{\sigma=0}^{\infty} \exp[-\{s+\lambda^{(1)}(1-\mu^{(1)}(s,1))\}\sigma]\, d_\sigma Pr\{\upsilon_t^{(1)}<\sigma \mid \upsilon_0^{(1)}=0\} e^{-\lambda^{(2)}t} d\lambda^{(2)}t$$

$$= \frac{\lambda^{(2)}-s+\lambda^{(1)}[\mu^{(1)}(s,1)-\mu^{(1)}(\lambda^{(2)},1)]}{[\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)](1-s/\lambda^{(2)})} \tag{B.14}$$

It is evident that the pre-waiting time of the $(n+1)$th customer of second priority level is given by

$$w_{n+1} = \theta_{n+1}^{(2)} + b_{n+1}^{(1)} \tag{B.15}$$

From (3.2) and (3.3), we have

$$E\{e^{-sw_{n+1}}\} = E\{e^{-s\theta_{n+1}^{(2)}}\}\cdot E\{e^{-sb_{n+1}^{(1)}}\}$$

$$= \frac{\lambda^{(2)}}{s+\lambda^{(2)}}\cdot\frac{\lambda^{(2)}-s+\lambda^{(1)}[\mu^{(1)}(s,1)-\mu^{(1)}(\lambda^{(2)},1)]}{[\lambda^{(1)}+\lambda^{(2)}-\lambda^{(1)}\mu^{(1)}(\lambda^{(2)},1)](1-s/\lambda^{(2)})}$$

if Re $s\geq 0$. Hence [].

Appendix C. Evaluation of Service Time Distributions

Introduction

In recent years, considerable amount of research work has been done in analytic performance evaluation of databases and computer networks[6,27,37,66,77,81]. However in most analytical models, exponential service time distributions are assumed without actually verifying it with the real system. But the usefulness of the analytic model depends very much on the specifications of the service time distribution of each component in the system[70]. Unfortunately very little study has been done in this area. The reasons are due to the difficulty and expensiveness of collecting the necessary data for the evaluation, lack of good methods to estimate the theoretical distribution accurately from the observed data, and the complexity of using general service time distributions in analytic model.

Although Coffman and Wood have pointed out that the distribution of interarrival time is a biphase or triphase hyperexponential distribution[14], and Fuchs and Jackson have suggested that the user think time has a lognormal or gamma distribution[23], very little study has been done in the estimation of the service time distribution. By ignoring the actual service time distribution and assuming an exponential distribution, the analysis of computer system could be made easier, but its accuracy suffers. In order to evaluate the existing computer systems accurately or provide a valuable guide to computer system design, the specification of the service time distribution is necessary[23,70].

In this section, we introduce an efficient method to evaluate the theoretical distribution functions of the service time of some computer system components and to decompose the distributions into several stages which can be easily applied in analytic models. Our method consists of three major steps. Firstly experimental models have to be built to collect the statistical data. i.e. the service time of the system components. For instance the inter-computer communication service times, and the database processing time are recorded which form the samples of the service time probability distribution function. Secondly a method is introduced to evaluate the actual distribution function based on the collected data, and the results of χ^2 tests of goodness of fit is presented for each fit to prove the hypothetic distribution to be acceptable. The probability distribution functions obtained from our model suggest that the distributions are of hyperexponential nature with starting point not necessaryly at zero. Thirdly Laplace transformation is performed to decompose the distribution into a form which can be presented with several stages. This allows an easy application in analytical evaluations[54].

Experimental Models of Service Time Distribution Measurements

The experimental models we have built are designated to evaluate the performance of two different types of components, the inter-computer communication and database processing. The model for inter-

computer communication is built on a local area network Ethernet. The communication is based on the 4.3BSD Interprocess Communication (IPC) facilities for UNIX[43]. The basic building block for this communication is the socket. Sockets exist within communication domains which are an abstraction introduced to bound common properties of processes. All our experimental workstations run the 4.3BSD Unix operating system. The experiment is carried out by using User Datagram Protocol (UDP).

The experimental model consists of two processes as shown in figure C.1. One is the coordinate process, the other is the participant process. The coordinate process generates UDP packets, while the participant process receives and sends them back to the coordinator. The coordinate process therefore gathers the statistical information of the round trip delay of the interprocess communication. One way communication delay can be easily obtained.

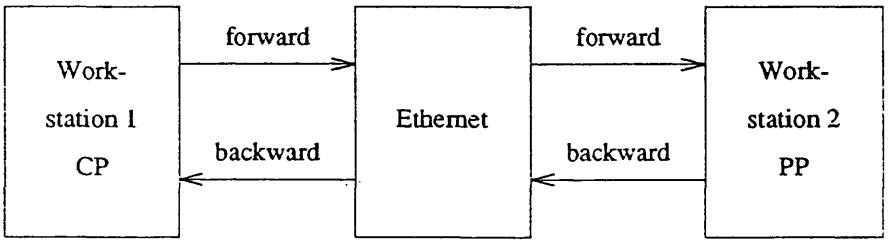


Figure C.1. Interprocess Communication Model

Two types of communications have been tested. One is called connection-oriented communication. A connection is set up before communication starts; then each interprocess communication only involves sending and receiving sockets. The other one is called connectionless communication which can be typically found in the datagram facilities in contemporary packet switched networks. For each communication a datagram socket is created with a name bound to it. Each communication delay of the connection-oriented communication is comparatively smaller than that of the connectionless if the time spent in setting up the connection is excluded. This connection set-up time can be omitted if the number of communications of each connection is very big.

In order to get the service time of the communication excluding the queueing delay and background load, the experiments were carried out in the evening when the Ethernet traffic is negligible. Data were collected on a weekly basis in the midnights. The packet size is 1024 bytes. Packets were sent in sequence to avoid queueing.

In order to test the generality of our method, another experiment model is also used to evaluate the database processing time. The experiment is based on the relational database UNIFY on Pyramid-9020 mainframe. Three types of database queries are being tested. Type I is a simple and frequently used read-from-a-relation type of query. Type II is a more complicated join type query. Type III is a collection of various different types of queues. All the data are collected in terms of service time of database processing.

Estimation of Probability Density Function

Here we introduce a method to approximately represent the observed discrete service time distribution by a probability distribution function composed of exponential series with x_0 away from zero, where x_0 is the starting point. That is

$$F(x)\begin{cases}\geq 0 & \text{if } x \geq x_0 \\ = 0 & \text{if } x < x_0\end{cases} \tag{C.1a}$$

The method consists of two steps. First the points of the observed discrete distribution are all shifted horizontally to the left to a distance of x_0, so that a new distribution is produced with starting point at zero; then after curve fitting a probability distribution function $G(x)$ is obtained by applying to the method stated bellow. The actual probability distribution function of the service time therefore is equal to

$$F(x) = G(x - x_0) \tag{C.1b}$$

The shifted discrete distribution data are given as (n+2) equi-spaced points (x_k, P_k), $(k=0,1,\cdots,n+1)$ where

$$P_k = Prob\{X \leq x_k\}\ , \quad (k=0,\ldots,n+1) \tag{C.2}$$

and X is the continuous random variable of the service time. This can then be transformed into (x_k, f_k), $k=0,\ldots,n$ where

$$\begin{aligned} f_k &= Prob\{x_k < X \leq x_{k+1}\} \\ &= Prob\{X \leq x_{k+1}\} - Prob\{X \leq x_k\} \\ &= P_{k+1} - P_k, \qquad (k=0,\ldots,n) \end{aligned} \tag{C.3}$$

Suppose $G(x)=Prob\{X \leq x\}$ has a hyperexponential form,

$$G(x) = \begin{cases} 1 - \sum_{i=1}^{m} w_i e^{-\lambda_i x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{C.4a}$$

and

$$\sum_{i=1}^{m} w_i = 1 \tag{C.4b}$$

It is obvious that

$$f_k = G(x_{k+1}) - G(x_k)\ , \quad (k=0,...,n) \tag{C.5}$$

A systems of non-linear equations can be set up from the above equation as follows

$$\sum_{i=1}^{m} w_i e^{-\lambda_i x_k} - \sum_{i=1}^{m} w_i e^{-\lambda_i x_{k+1}} = f_k\ , \quad (k=0,...,n) \tag{C.6}$$

Now let $x_k=k\cdot h$ where h is the equi-spaced step. Then the equations can be rewritten as

$$\sum_{i=1}^{m} w_i(1-e^{-\lambda_i h})e^{-\lambda_i kh} = f_k\ , \quad (k=0,...,n) \tag{C.7}$$

Let

$$C_i = w_i(1-e^{-\lambda_i h})\ , \quad (i=i,...,m) \tag{C.8a}$$

$$V_i = e^{-\lambda_i h}\ , \quad (i=1,...,m) \tag{C.8b}$$

Equation (C.7) can be represented as

$$\begin{cases} C_1 + C_2 + \cdots + C_m = f_0 \\ C_1V_1 + C_2V_2 + \cdots + C_mV_m = f_1 \\ C_1V_1^2 + C_2V_2^2 + \cdots + C_mV_m^2 = f_2 \\ \cdots + \cdots + \cdots + \cdots = \cdots \\ C_1V_1^n + C_2V_2^n + \cdots + C_mV_m^n = f_m \end{cases} \tag{C.9}$$

By applying to the method introduced by Lanczos and Prony[41], V_i, $(i=1,...,m)$ can be solved after some transformations with the condition $n\geq 2m-1$. Then

$$\lambda_i = -\frac{\ln V_i}{h}\ , \quad (i=1,...,m') \tag{C.10}$$

Because of the possibility of complex or negative λ_i , the actual number of λ_i, i.e. the number of exponentials could be less than m. Here suppose that the actual number of λ_i found from equation (C.9) and (C.10) are m'. Without loss of generality we still use m rather than m', but bear in mind that now $m=m'$.

It seems that w_i, $(i=1,...,m)$ can be solved from equation (C.8a), (C.8b), and(C.9), but this can not guarantee that the resultant $G(x)$ is a probability distribution function. In order to solve this problem, w_i, $(i=1,...,m)$ are obtained by using a least squared method.

$$E = \sum_{k=0}^{n} [\, f_k - \sum_{i=1}^{m} w_i(1-e^{-\lambda_i h})e^{-\lambda_i kh}\,]^2 \tag{C.11a}$$

$$\frac{\partial E}{\partial w_j} = 0\,, \quad (j=1,...,m) \tag{C.11b}$$

The necessary and sufficient conditions for $G(x)$ to be a probability distribution function is that (assuming, without loss of generality $\lambda_1 < \lambda_2 < \cdots < \lambda_m$)[75]

$$\sum_{i=1}^{m} w_i = 1 \tag{C.12a}$$

$$w_1 > 0\,, \text{ and}$$

$$\sum_{i=1}^{k} w_i \lambda_i \geq 0, \quad (k=1,...,m) \tag{C.12b}$$

Thus (C.11a) and (C.11b) becomes

$$E = \sum_{k=0}^{n} [f_k - \sum_{i=1}^{m} w_i (1-e^{-\lambda_i h})]^2 + \xi(\sum_{i=1}^{m} w_i - 1) \tag{C.13}$$

$$\begin{cases} \dfrac{\partial E}{\partial w_j} = 0 \\ \sum\limits_{i=1}^{m} w_i = 1 \end{cases}, \qquad (j=1,...,m) \tag{C.14}$$

The $w_i, i=1,...,m$ can be solved from the above equation. Therefore all the parameters of $G(x)$ have been estimated. Thus the actual probability distribution function of service time is

$$F(x) = G(x-x_0)$$

$$= \begin{cases} 1-\sum\limits_{i=1}^{m} w_i e^{-\lambda_i (x-x_0)} & \text{if } x \geq x_0 \\ 0 & \text{if } x < x_0 \end{cases} \tag{C.15}$$

And the correspondent probability density function (p.d.f.) is

$$f(x) = \begin{cases} \sum\limits_{i=1}^{m} w_i \lambda_i e^{-\lambda_i (x-x_0)} & \text{if } x \geq x_0 \\ 0 & \text{if } x < x_0 \end{cases} \tag{C.16}$$

Numerical methods have been used to solve equation (C.9) and (C.14) for the estimation of the theoretical distributions of both the connection-oriented and connectionless communication service times. The resultant service time distribution is a biphase, hyperexponential distribution,

$$f_{cm}(x) = w\lambda_1 e^{-\lambda_1(x-x_0)} + (1-w)\lambda_2 e^{-\lambda_2(x-x_0)} \quad \text{(C.17)}$$

For connection-oriented communication the parameters of the above equation are x_0=6.0ms, w=0.7025, λ_1=1.1913/ms, and λ_2=0.1224/ms. A χ^2 goodness of fit test has been performed in which theoretical p.d.f. is fitted to the observed distribution of the random variables. With 12 degrees of freedom the chi squared value equals to χ^2=12.26, so the assumption is acceptable at the significance level of 25%. For the connectionless communication the parameters are x_0=16.7ms, w=0.775, λ_1=0.9435/ms, and λ_2=3.7189/ms. With 5 degrees of freedom, the chi squared value equals to χ^2=4.68; thus the assumption is proven to be acceptable at the significance level of 25%.

The theoretical distribution of three different types of database queries forms an exponential distribution with starting point at x_0,

$$f_{db}(x) = \lambda e^{-\lambda(x-x_0)} \quad \text{(C.18)}$$

The correspondent parameters of type I query are x_0=925.0ms, λ=0.01681/ms. The chi squared value is χ^2=12.85 with 11 degrees of freedom. The acceptance significance level is again at 25%. The parameters of type II query are x_0=3617.67ms , and λ=0.00122/ms. The chi squared value is χ^2=10.28 with 7 degrees of freedom. The chi squared significance level is 10%.

We can see from the results of the goodness of fit test that the above method can give a very fine estimation of the actual distribution. The acceptance significance level is quite high in our examples, i.e. between 10% to 25%. Other methods of estimating exponential mixtures such as the method of moments[34,67] and the method of maximum likelihood[19,30] are unable to provide such a good fit from the observed measurements especially when the observed distributions have more than two exponential components.

The Service Model for Analytic Performance Evaluation

We can see from the above examples that a wide variety of service time distributions of computer systems can be represented with a hyperexponential p.d.f.

$$f(x) = \begin{cases} \sum_{i=1}^{m} w_i \lambda_i e^{-\lambda_i(x-x_0)} & \text{if } x \geq x_0 \\ 0 & \text{if } x < x_0 \end{cases} \quad \text{(C.19)}$$

with its mean

$$E[X] = x_0 + \sum_{i=1}^{m} \frac{w_i}{\lambda_i} \quad \text{(C.20a)}$$

and the variance

$$V[X] = 2\sum_{i=1}^{m} \frac{w_i}{\lambda_i^2} - \left[\sum_{i=1}^{m} \frac{w_i}{\lambda_i}\right]^2 \tag{C.20b}$$

The coefficient of variance equals to

$$Cov[X] = \frac{\sqrt{2\sum_{i=1}^{m} \frac{w_i}{\lambda_i^2} - (\sum_{i=1}^{m} \frac{w_i}{\lambda_i})^2}}{x_0 + \sum_{i=1}^{m} \frac{w_i}{\lambda_i}} \tag{C.20c}$$

The kind of distribution in the form of equation (C.19) provides a model for a wide range of service times characterized by a maximum randomness, i.e. maximum $Cov[X]$ when $m=1$ and $\lambda=\frac{1}{E[X]-x_0}$[54] and no randomness when some $\lambda_i \to \infty$ but $\sum_{i=1}^{m} \frac{w_i}{\lambda_i}=C$, as we can see from equation (C.20c).

Now let us form the Laplace transform of the equation (C.19).

$$f^*(s) = e^{sx_0} \sum_{i=1}^{m} \frac{w_i \lambda_i}{s+\lambda_i} \tag{C.21}$$

This equation can also be represented as

$$f^*(s) = e^{sx_0} g^*(s) \tag{C.22}$$

where

$$g^*(s) = \sum_{i=1}^{m} \frac{w_i \lambda i}{s+\lambda_i} \tag{C.23}$$

Equation (C.22) represents the convolution of two distributions with their Laplace transforms equal to e^{sx_0} and $g^*(s)$ respectively. The former is the Laplace form of the deterministic distribution with the constant service time equal to x_0, while the later also has simple probability interpretations: it has m stages of service each of which has the mean service time $1/\lambda_i$, $(i=1,...,m)$. For each customer there is a probability w_i of entering the ith stage and then the service consists of a single stage[17]. The diagram of this service model is illustrated in figure C.2.

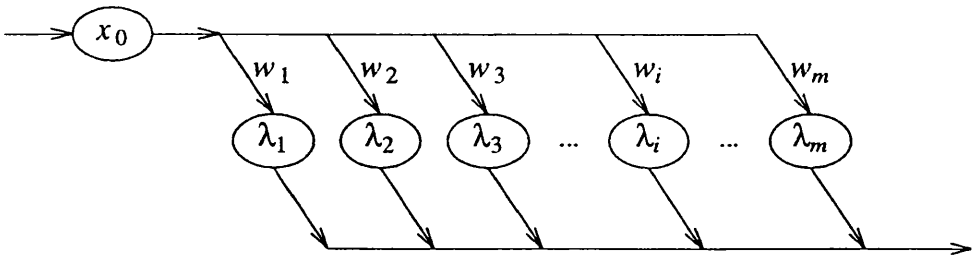


Figure C.2. The Model of Service Time Distribution

The method introduced in this section provides method to estimate the service time distributions of real systems and construct the correspondent stochastic model for the analytic performance evaluation. The method is also robust in the sense that different types of computer service centre such as connection-oriented and connectionless communications can have the same structures of the distribution only with different parameters, so are different types of database query processing.

References.

1. Abramson, N., "Packet switching with satellites", *National Computer Conference, AFIPS Conference* **42**pp. 659-702 (1973).

2. Agrawal, R., Carey, M., and Livny, M., "Models for studying concurrency control perforamnce: alternatives and implications", *Proc. of the ACM SIGMOD Int. Conf. on Manage. of Data*, (May 1985).

3. Alsberg, P.A. and Day, J.D., "A principle for resilient sharing of distributed resources", *Proc. 2nd Int. Conf. Software Eng.*, pp. 562-570 (Oct. 1976).

4. Badal, D.Z., "The analysis of the effect of concurrency control on distributed database system performance", *6th VLDB*, pp. 376-383 (Oct. 1980).

5. Bard, Y., "Some extensions to multiclass queueing network analysis", *Performance of Computer Systems , Aratbo et al Eds.*, pp. 51-62, North-Holland (1979).

6. Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, F.G., "Open, closed, and mixed networks of queues with different class of customers", *J. ACM* **22**pp. 248-260 (April 1975).

7. Bernstein, P.A. and Goodman, N., "Timestamps based algorithms for concurrency control in distributed database systems", *Proc. 6th Int. Conf. VLDB*, (Oct. 1980).

8. Bernstein, P.A. and Goodman, N., "Concurrency control in distributed database systems", *ACM Comput. Surveys* **13** (2)pp. 185-221 (June, 1981).

9. Buzen, J.P., "Fundamental operational laws of computer system performance", *Acta Infromatica* **7**pp. 167-182 (1976).

10. Carey, M., "Modeling and evaluation of database concurrency control algorithms", *Ph.D dissertation*, (Aug. 1983).

11. Carey, M. and Stonebraker, M., "The performance of concurrency control algorithms for database management systems", *Proc. of the 10th Int. Conf. on VLDB*, , VLDB Foundation (Aug. 1984).

12. Ceri, S. and Pelagatti, G., *Distributed databases, principles and systems*, McGraw Hill (1985).

13. Coffman, E.G.Jr., Muntz, R.R., and Trotter, T., "Waiting time distribution for processor-sharing systems", *J. ACM* **17**pp. 123-130 (1970).

14. Coffman, E.G.Jr. and Wood, R.C., "Interarrival statistics for time sharing systems", *Comm. ACM* **9**(**7**)pp. 500-503 (July 1966).

15. Cohen, J.W., *The single server queue*, Wiley-Interscience, New York (1969).

16. Courtois, P.J., "Decomposability, instabilities, and saturation in multiprogramming systems", *Comm. ACM* 18pp. 371-377 (1975).

17. Cox, D.R., "A use of complex probabilities in the theory of stochastic processes", *Proc. Cambridge Phil. Soc.* 51pp. 313-319 (1955).

18. Cox, D.R., *The theory of stochastic process*, Methuen, London (1965).

19. Dempster, A.P., Laird, N.M., and Rubin, D.B., "Maximum likelihood from incomplete data via the EM algorithm", *J. Royal Statist. Soc.*, Series B, 39pp. 1-38 (1977).

20. Digital,, Intel,, and Xerox,, *The Ethernet, a local area network, data link layer and physical layer specifications*. September 1980.

21. Doob, J.L., *Stochastic Processes*, Wiley, New York (1953).

22. Ferrari, D., "Considerations on the insularity of performance evaluaton", *IEEE Transactions on Software Engineering* SE-12 (6)pp. 678-683 (June 1986).

23. Fuchs, E. and Jackson, P.E., "Estimates of distributions of random variables for certain computer communications traffic models", *Comm. ACM* 13(12)pp. 752-757 (Dec. 1970).

24. Garcia-Molina, H., "A concurrency control mechanism for distributed databases which use centralized locking controllers", *Proc. 4th Berkeley Workshop Distributed Database and Computer Networks*, (Auguest 1979).

25. Garcia-Molina, H., "Perforamance of update algorithms for replicated data in a distributed databases", *Ph.D dissertation*, , Computer Science Dept., Stanford Univ., Stanford, Calif. (June 1979).

26. Gelenbe, E., "On approximate computer system models", *J.ACM* 22pp. 261-269 (April 1975).

27. Gelenbe, E. and R.Muntz, R., "Probabilistic models of computer systems --- part I (exact results)", *Acta Informatica* 7pp. 35-60 (1976).

28. Gray, J.N., "Notes on database operating systems", *In Operating Systems -- An Advanced Course, R. Bayer, R.M.Graham and G.Seegmuller, Eds.*, pp. 393-481, Springer-Verlag (1978).

29. Hac, A., "A decomposition solution to a queueing network model of a distributed file system with dynamic locking", *IEEE Transactions on Software Engineering* SE-12, No.4pp. 521-530 (April 1986).

30. Hasselblad, V., "Estimation of finite mistures of distribution from the exponential family", *J. Amer. Statist. Assoc.* **64**pp. 1459-71 (1969).

31. Irani, K.B. and Lin, H.K., "Queueing network models for concurrent transaction processing in a database system", *Proc. of ACM SIGMOD 79*, pp. 134-142 (1979).

32. Jackson, J.R., "Networks of waiting lines", *Oper.Res.* 5pp. 518-521 (1957).

33. Jackson, J.R., "Jobshop-like queueing systems", *Mana.Sci.* **10(1)**pp. 131-142 (Oct. 1963).

34. Kabir, A.B.M.L., "Estimation of parameters of a finite mixture of distributions", *J. Royal Statist. Soc.*, **Series B, 30**pp. 472-478 (1968).

35. Kiefer, J. and Wolfowitz, J., "On the theory of queues with many servers", *Trans. of the Amer. Math. Soc.* **78**pp. 1-18 (1955).

36. Kleinrock, L., *Communication nets; stochastic message flow and delay*, McGraw-Hill, New York (1964).

37. Kleinrock, L., *Queueing systems, computer applications*, Wiley-Interscience, New York (1976).

38. Knuth, D.E., *The art of computer programming: sorting and searching*, Addison Wesley (1973).

39. Kobayashi, H., "Application of the diffusion approximation queueing systems: part I and II", *J.ACM* **21**pp. 316-328, 459-469 (1974).

40. LaBarre, G.E., "Analytic and simulation results for CSMA connection protocols", *Electron. System Div., AFSC, Hanscom AFB, Massachusetts* **ESD-TR-76-126** (May 1979).

41. Lanczos, C., *Applied analysis*, Sir ISAAC Pitman & Sons Ltd, London.

42. Lavenberg, S.S., *Computer performance modeling handbook*, Academic Press (1983).

43. Leffler, S.J., Fabry, R.S., and Joy, W.N., "A 4.2BSD interprocess communication primer", *Draft*, (Sept. 1985).

44. Leung, C.H.C. and Lu, J.Z., "The architecture of real-time distributed databases for computer network monitoring", *Workshop on Distributed Systems -- Theory and Practice in Computer Distributed Systems: Theory and Practice, H.Zedan (ed)*, (1988).

45. Lu, J.Z., "The current state of the art of DDBs", *Internal Notes 1714*, (1985).

46. Lu, J.Z., "The distributed database --- PROTEUS", *internal note 1758*, (1985).

47. Lu, J.Z., "A proposed distributed databases architecture for network management", *Internal Note 1790*, (1985).

48. Lu, J.Z., "A survey of distributed database protocols --- a RPC approach", *Internal Note 1878*, (10 Jan. 1986).

49. Lu, J.Z., "An analytic model for static locking in a centralized database system", *Internal Note 2195*, (Oct. 1987).

50. Lu, J.Z., "Further development of DRFS and its suitability for general purpose distributed computing", *A.3136*, (April 1988).

51. Lu, J.Z., "Distributed robust file store: initial design with multicast approach", *A.3129*, (January 1988).

52. Lu, J.Z., "An implementation of distributed robust file store", *A.3137*, (June 1988).

53. Lu, J.Z., "The failure recovery algoritym of the DRFS", *A.3135*, (March 1988).

54. Lu, J.Z. and Leung, C.H.C., "A statistical study of the service time distributions of computer systems", *Internal Note 2136*, (July 1987).

55. Lu, J.Z. and Leung, C.H.C., "Performance evaluation of distributed database locking with a diffusion approximation approach", *Internal Note 2137*, (July 1987).

56. Lu, J.Z. and Paliwoda, K., "Testing the DRFS naming server", *A.3133*, (February 1988).

57. Lu, J.Z. and Paliwoda, K., "The DRFS testing cases", *A.3134*, (February 1988).

58. Lu, J.Z. and Paliwoda, K., "The DRFS naming server with multicast", *A.3131*, (January 1988).

59. Lu, J.Z. and Paliwoda, K., "Program structures of the DRFS naming server with multicast", *A.3132*, (January 1988).

60. Lu, J.Z., Paliwoda, K., and Wilbur, S.R., "Using multicast in a distributed robust file store", *Alvey Conference*, (July 1988).

61. Menasce, D.A. and Nakanishi, T., "Optimistic versus pessimistic concurrency control mechanisms in database management systems", *Information System* 7(1)pp. 13-27 (1982).

62. Menasce, D.A. and Nakanishi, T., "Performance evaluation on a two-phase commit based protocol for DDBs", *Proc. of the ACM Symp. on Principles of Database Systems*, pp. 247-255, ACM New York (Mar. 1982).

63. Mitra, D. and Weinberger, P.J., "Probabilistic models of database locking: solutions, computational algorithms, and asymptotics", *J.ACM* **31 (4)**pp. 855-878 (Oct. 1984).

64. Newell, G.F., "Approximate stochastic behaviour of n-server service systems with large n", *Lecture Notes in Economics and Mathematical Systems, No.87 Springer-Verlag*, (1973).

65. Reed, D.P., "Naming and synchronization in a decentralized computer system", *Ph.D dessertation*, , Dept. of Electrical Engineering, MIT, Cambridge, Mass. (Sept. 1978).

66. Reiser, M., "A queueing network analysis of computer communication networks with window flow control", *IEEE Trans. on Commun.* **27**pp. 1199-1209 (Aug 1979).

67. Rider, P.R., "The method of moments applied to a mixture of two exponential distributions", *Annals. of Math. Stats.* **32**pp. 143-147 (1961).

68. Ries, D.R. and Stonebraker, M., "Effects of locking granularity on database management system", *ACM ToDS* **2(3)**pp. 233-246 (Sept. 1977).

69. Ries, D.R. and Stonebraker, M.R., "Locking granularity revisited", *ACM ToDS* **4(2)**pp. 210-227 (June 1979).

70. Sackman, H., *Experimental investigation of user performance in time shareing computing systems: retrospect, prospect and the public interest.* May 1967.

71. Sevcik, K.C., "Priority scheduling disciplines in queueing network models of computer systems", *Proc. IFIP Congress 77*, pp. 565-570, North-Holland Publishing Co. (1977).

72. Sevcik, K.C., "Comparison of concurrency control methods using analytic models", *Information Processing 83, R.E.A.Mason ed.*, pp. 847-856, North-Holland (1983).

73. Sheth, A.P., Singhal, A., and Liu, M.T., "An analysis of the effect of network parameters on the performance of distributed database systems", *IEEE SE* **SE-11 (10)**pp. 1174-1184 (Oct. 1985).

74. Shum, A.W. and Spirakis, P.G., "Performance analysis of concurrency control methods in database systems", *Performance '81, F.J.Kylstra ed.*, pp. 1-19, Elserire North-Holland (1981).

75. Steutel, F.W., "Note on the infinite divisibility of exponential mixtures", *Ann. Math. Statist.* **38**pp. 1303-1305 (1967).

76. Stonebraker, M., "Concurrency control and consistency of multiple copies of data of distributed INGRES", *IEEE Transactions on Software Engineering* **SE-5(5)** (May 1979).

77. Tay, Y.C., Goodman, N., and Suri, R., "Locking performance in centralized database", *ACM ToDS* **10 (4)**pp. 415-462 (Dec. 1985).

78. Tay, Y.C., Suri, R., and Goodman, N., "A mean value performance model of locking in databases: the no waiting case", *J.ACM* **32 (3)**pp. 618-651 (July 1985).

79. Thomas, R.H., "A solution to the concurrency control problem for multiple copy databases", *Proc. 1978 COMPCON Conf. (IEEE)*, (1978).

80. Thomas, R.H., "A majority consensus approach to concurrency control for multiple copy databases", *ACM ToDS* **4**(2)pp. 180-209 (June 1979).

81. Thomasian, A., "Performance evaluation of centralized databases with static locking", *IEEE SE* **SE-11** (4)pp. 346-355 (Apr. 1985).

82. Tobagi, F.A., "Random access techniques for data transmission over packet switched radio networks", *Ph.D Thesis, Computer Sci. Dept. , School of Eng. and Appl. Univ. of California, Los Angales, UCLA-England 17499*, (Dec. 1974).

83. Wiederhold, G., *Database design*, McGraw-Hill, New York (1983).

84. Yao, A., "Random 3-2 trees", *Acta Informatica* 2(9)pp. 159-170 (1978).