N91-23971

Building a Generalized Distributed System Model

R. Mukkamala
E.C. Foudriat

Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529.

Annual Report and Renewal Request

Abstract

The key elements in the first year (1990-91) of our project were:

- Investigate the effects of modeling on distributed system performance predictions.
- Look at possible graphical interfaces to the proposed distributed prototype and simulator system.
- Conduct preliminary studies towards the design of a generalized distributed system.

In the second year of the project (1991-92), we propose to

- Develop detailed designs for the prototype.
- Implement and test the system.
- Conduct further studies on modeling distributed systems.

1 Introduction

In the 1990-91 proposal, we discussed the need for building a modeling tool for both analysis and design of distributed systems. To this end, we have been considering different design architectures for the modeling tool. Since many of the research institutions have access to networks of workstations, we have decided to build a tool running on top of the workstations to function as a prototype as well as a distributed simulator for a computing system.

In addition, we have been investigating the effects of system modeling on performance prediction in distributed systems. While some performance measures such as the average number of participating node set size of a distributed transaction is not very sensitive to the underlying model, measures such as transaction commutativity measures are quite sensitive to the evaluation models.

We have also considered the effects of static locking and deadlocks on the performance predictions of distributed transactions. While the probability of deadlock is considerably small in a typical distributed system, its effects on performance could be significant.

In this report, we summarize our progress in these three areas and describe the details of the proposed work.

2 Distributed System Model: Prototype/Simulator

The main goals of our efforts in building a general tool for simulation and prototyping of distributed systems are:

- A framework to experiment with distributed algorithms/systems.
- Implement in terms of basic primitives (e.g., RPC, reliable communication).
- A good user interface - preferably with graphic and mouse functions.
- Provisions to include user specific code for different components.
- A library of procedures representing typical options for components (e.g. two-phase locking).
- A base for distributed simulation as well as prototyping.
- Efficient mechanisms to monitor and display the activities.

- Powerful performance analysis tools.

To this end, we started looking at a transaction oriented distributed system. Since our aim is to provide a general framework rather than to provide a solution to a particular model, our goal is to provide some of the basic primitives at the bottom layer, and let the user build the needed upper level software. To make the prototype usable for a novice user, we propose to provide a graphic interface through which a user can specify the system configuration. As an example application, we considered distributed database system modeling. As shown in Figure 1, we identified seven major components. Each of these components can be further described in a detailed model. For example, the local manager can be modeled as a coordinator of local concurrency control manager and the transaction resource manager. Given a set of components, the control structure of the system can be represented through directional links. Figure 2 illustrates one such control structure.

After considering several alternates, we decided to base the graphic interface on the lines of the MIT Network simulator. The MIT simulator is developed at Massachusetts Institute of Technology with funding from DARPA. Even though it is intended for simulating communication networks, we have decided to adopt its graphic interfacing routines for our distributed simulator. Since the source code (in C) is available, we are modifying this code to suit our needs. Some of its distinguishing characteristics of the network simulator are:

- Internetwork simulator
- Components include gateways, network links, hosts, TCPs and users.
- Network configuration is displayed on the screen.
- User can control the simulation.
- Network configuration can be modified with the mouse.
- Other simulation parameters can be changed on-line using the mouse.
- Network configuration can be saved for later use.
- Several performance measures may be printed.

Figure 1

Distributed System Model

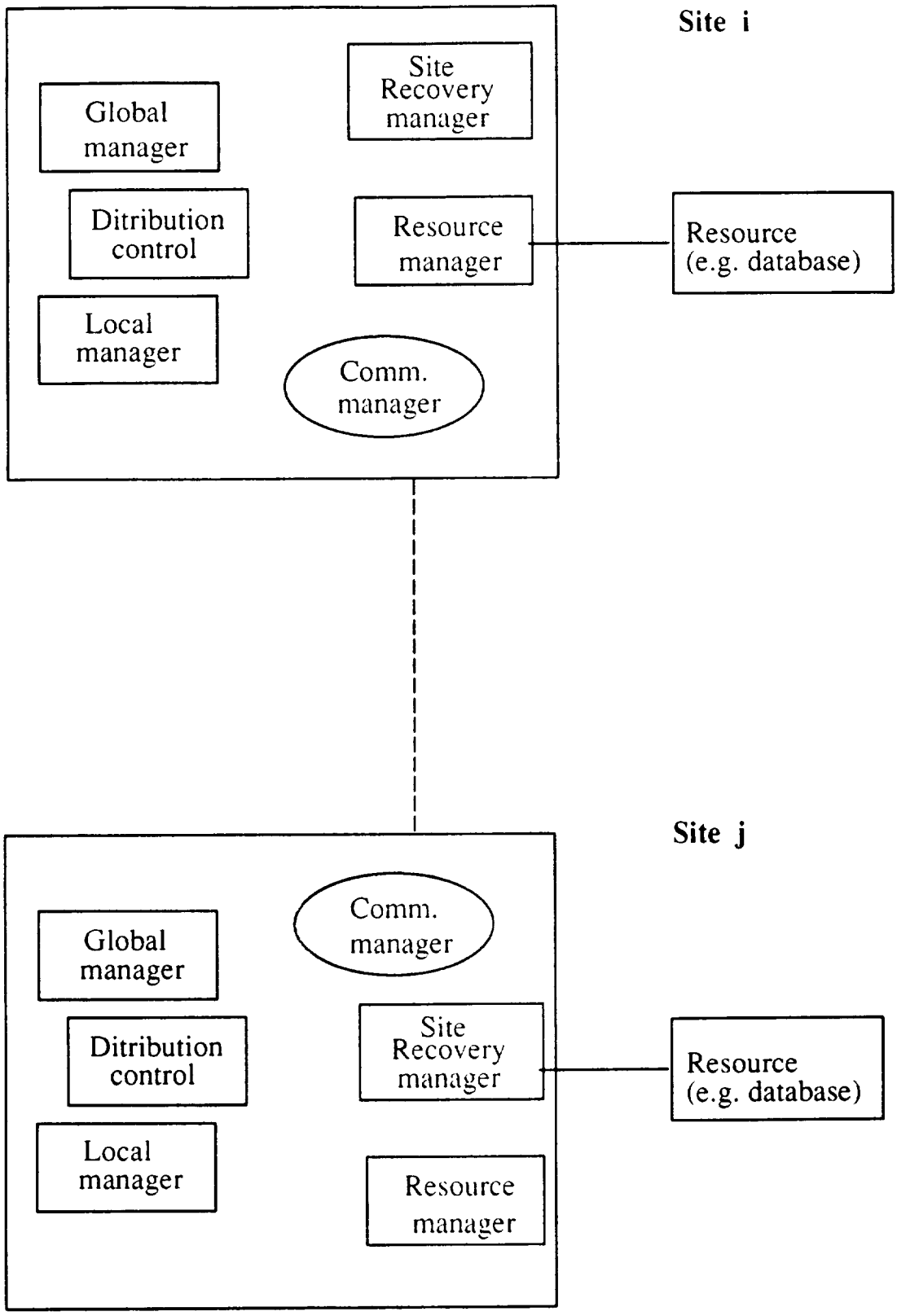
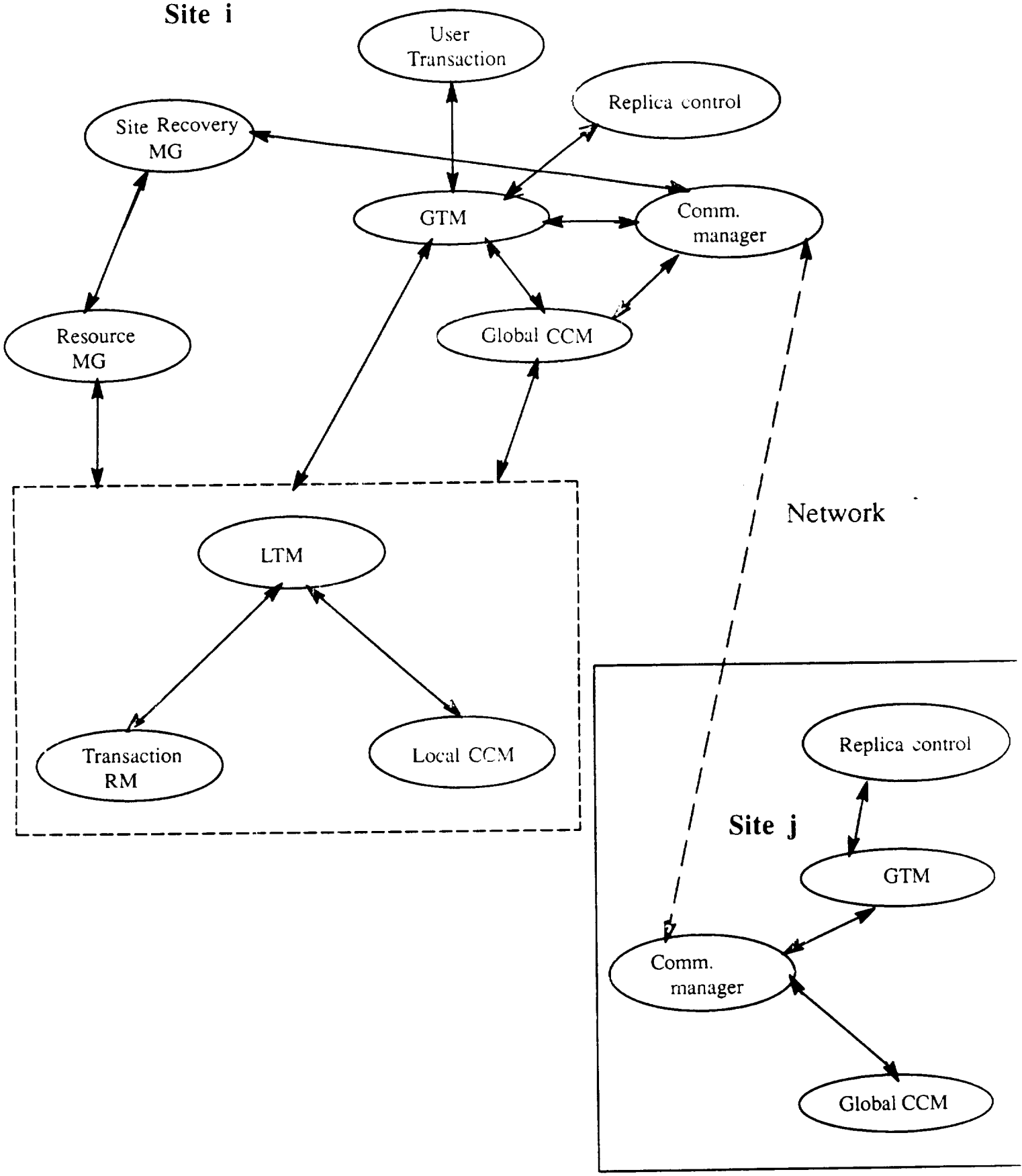


Figure 2


Site i
User Transaction
Replica control
Site Recovery MG
GTM
Comm. manager
Resource MG
Global CCM
LTM
Transaction RM
Local CCM
Network
Site j
Replica control
GTM
Comm. manager
Global CCM

Since process communication is a basic primitive needed in distributed systems, we have decided to provide this as a basic mechanism in our system. Currently, we are experimenting with the Sun RPC system calls to design a high-level primitive. RPC has several advantages including:

- Hiding details of network programming
- Availability of library routines
- Hiding the operating system dependencies
- Availability of the standard data representation using XDR format which allows a simple way of transferring data.

3 Effects of Modeling on Performance Predictions

As a second part of our study, we have conducted investigations to determine the impact of modeling on distributed system performance. Here, we summarize the results of two such studies:

Study 1: Effect of Data Distribution Models on Transaction Commutativity [2]. Recognizing commutativity among transactions appears to reduce the number of rollbacks (at the time of merge) in a partitioned distributed database system [1]. The main objective of this study is to determine the impact of data distribution modeling on the evaluation of the benefits due to commutativity. We studied the effects of six distinct data distribution models on the evaluation of the number of rollbacks. We derived closed form expressions for five of the six models, and used simulation for the sixth model. The conclusions from this study are summarized as follows.

- Random data models that assume only average information about the system result in conservative estimates of system throughput.
- Adding more system information does not necessarily lead to better approximations. In this paper, the system information is increased from model 6 to model 2. Even though this increases the computational complexity, it does not result in any significant improvement in the estimation of the number of rollbacks.

- Transaction commutativity appears to significantly reduce transaction rollbacks in a partitioned distributed database system. *This fact is only evident from the analysis of model 1.* On the other hand, when we look at models 2-6, it is possible to conclude that commutativity is not helpful unless it is extremely high. Thus, conclusions from model 1 and models 2-6 are *contradictory.*

- The replication distribution (i.e., the actual number of copies for each object) seems to effect the evaluations significantly. Thus, accurate modeling of this distribution is vital to evaluation of rollbacks.

Study 2: Effect of Data Distribution and Reliability Models on Transaction Availability [3]. In this study, we selected three abstractions for data distribution modeling and three for node reliability modeling, and constructed six system models. Here, transaction availability is defined as the probability with which all data copies required by a transaction are available at the beginning of its execution. As before, we could derive closed form expressions with five of the six models (using probabilistic analysis), and used simulation for the other model. A transaction was characterized by the number of data objects that it accesses, s. The conclusions derived from this study are summarized as follows.

- By choosing a proper distributed database model, the computational complexity of transaction availability evaluations can be significantly reduced.

- For values of $s \leq 10$, all models result in almost the same transaction evaluation.

- The degree of replication of individual (or group) data objects seems to have a significant effect on transaction availabilities. Thus, when different data objects have different copies, adopting average degree of replication at the system level may not result in accurate availability evaluations.

- The actual distribution of data object copies has some, if not significant, impact on availability evaluation.

- In a heterogeneous environment where different nodes may have different reliabilities, it is sufficient to represent each node by the average node reliability, without affecting the availability evaluations.

Having conducted these studies, we conclude that

- Adopting simple models may drastically reduce the complexity of metric evaluations.
- Choosing analytically tractable models enables easy interpretation of functional dependencies.
- By choosing inappropriate models, for either analytical tractability or conceptual simplicity, it is possible to arrive at incorrect conclusions.
- Model choice is highly dependent on the metric. While a simple model serves well for one metric, it may be insufficient for another metric.

4 Determining the Effects of Locking on Distributed Transactions

Deadlocks are known to deteriorate performance in both centralized and distributed database systems [4,5]. In spite of this, several performance studies have ignored the deadlock problem in their analyses [6]. In [4], Shyu and Li proposed an elegant technique to evaluate the response time and throughput of transactions in a non-replicated DDS. Assuming *exclusive* locking (i.e., only write operations), they model the queue of lock requests at an object as a M/M/1 queue. This results in a closed-form for the waiting time distribution at a node, expressed in terms of the average rates of arrivals of requests and the average lock-holding time.

In general, a database transaction reads from a set of data objects (the read-set) and writes on to a set of data objects (the write-set). In this paper, we consider both the the read and the write operations of database transactions, and propose a technique for performance evaluation.

We make the following observations from evaluations made with our technique.

- As expected, the presence of shared locks has a substantial impact on the probability of deadlock occurrence. When only 1/3 of the accessed data objects are updated, the probability of deadlock is considerably small as compared to when all objects are updated.
- The observations about the deadlock probabilities are also valid for restart probabilities.

- Transaction response times are also quite sensitive to the ratio of shared locks. Here, we compare the response times when deadlocks are ignored with those obtained when deadlocks are considered. The effect of deadlocks is more predominant at higher transaction loads and with smaller values of read ratio. When 1/3 of the accessed objects are updated, the effect of deadlocks is not significant on response time.

- The effect of deadlocks on response time is decreased with the increase in the number of data items. Obviously, this is due to the decrease in probability of conflicts and hence a decrease in deadlock occurrence. When only 1/3 of the accessed data are updated, this effect is almost insignificant. When 2/3 of the accessed data are updated, deadlocks seems to have a noticeable effect on response time.

- When a small number of data objects are accessed, the probability of deadlock is negligible, and hence its effect on response time is small. When more data objects are accessed, the effect of deadlocks on response times is significant.

5 Summary of Accomplishments in 1990-91

We have published the results of our research (since August 1990) in two conferences. In addition, two papers are submitted for publication in international journals. These are:

1. Y. Kuang and R. Mukkamala, "Performance Analysis of Static Locking in Replicated Distributed Database Systems," *Proc. Southeastcon 1991*, pp. 698-701.

2. Y. Kuang and R. Mukkamala, "A Note on the Performance Analysis of Static Locking in Distributed Database Systems", Submitted to IEEE Trans. Computers, December 1990.

3. R. Mukkamala, "Effects of Distributed Database Modeling on Evaluation of Transaction Rollbacks," *Proc. WSC'91*, December 1990, pp. 839-845.

4. R. Mukkamala, "Measuring the Effects of Distributed Database Models On Transaction Availability Measures," Submitted to Performance Evaluation Journal, March 1991.

In addition, our current work on building the prototype for a distributed system should result in several conference and journal papers in 1991-92.

6 Proposed Research Efforts in 1991-92

During the next grant period (August 1991 to July 1992), we propose to continue the study and development of the distributed prototyping and simulator system. The main main problems that we need to solve in this period are:

- Complete the graphic interface design and implement it on Sun workstations.
- Investigate efficient means of offering flexible as well as efficient means of specifying interfacing between system components. We expect this phase to consume considerable time.
- Design, build, and test a specific system using the primitives offered by the system. Experiences from building a specific system should aid us in developing a generalized prototyping tool.
- We propose to use the prototype to evaluate the performance of several distributed mutual exclusion policies. Such a study may result in the development of new policies.
- We propose to do further investigations in modeling of distributed systems and determine their impact on predictive analysis tools.

References

[1] S. Jajodia and P. Speckman, "Reduction of conflicts in partitioned databases," Proc. 19th Annual Con. on Information Sciences and Systems, 1985, pp. 349-335.

[2] R. Mukkamala, "Effects of Distributed Database Modeling on Evaluation of Transaction Rollbacks," *Proc. WSC'91*, December 1990, pp. 839-845.

[3] R. Mukkamala, "Measuring the Effects of Distributed Database Models On Transaction Availability Measures," Submitted for publication, March 1991.

[4] S.-C. Shyu and V. O. K. Li, "Performance analysis of static locking in distributed database systems," *IEEE Trans. Computers*, vol. 39, no. 6, pp. 741-751, June 1990.

[5] Y. C. Tay, R. Suri, and N. Goodman, "A mean value performance model for locking in databases: The no-waiting case," *J. ACM*, vol. 32, no. 3, pp. 618-651, July 1985.

[6] M. Singhal and A. K. Agrawala, "Performance analysis of an algorithm for concurrency control in replicated database systems," *Proc. ACM SIGMETRICS Conf. Measurement Modeling Comput. Syst.*, 1986, pp. 159-169.

APPENDIX