INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

TRANSACTION MANAGEMENT IN MOBILE MULTIDATABASES

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

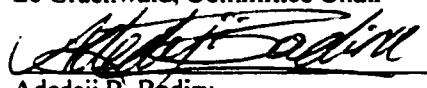
By
RAVI A. DIRCKZE
Norman, Oklahoma
1999

TRANSACTION MANAGEMENT
IN MOBILE MULTIDATABASES

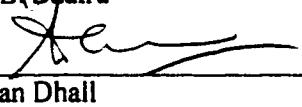
A Dissertation APPROVED FOR THE
DEPARTMENT OF COMPUTER SCIENCE

BY

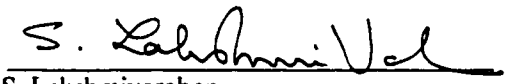
Le Gruenwald, Committee Chair



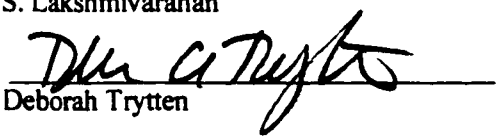
Adedeji B. Badiru



Sudarshan Dhall



S. Lakshmivarahan



Deborah Trytten

ACKNOWLEDGEMENT

Inspiration, knowledge, and encouragement are drawn from many sources. At the very beginning of this dissertation I would like to acknowledge some of the people who encouraged me to pursue my doctorate, those who inspired me, and those who helped me achieve a wealth of knowledge.

First, I would like to thank my advisor Dr. Le Gruenwald. At every stage, you provided me with the advice and guidance that I needed to enhance my knowledge and to improve my research. It was certainly a pleasant and fruitful learning experience. I would also like to thank the rest of my committee, Drs. A. Badiru, S. Dhall, S. Lakshmivarahan, and D. Trytten, who have supported and enriched my doctoral study in more ways than one.

Next, I would like to thank my parents for their love and support in all my endeavors. You have been my primary source of encouragement during my entire college education. I would also like to say a BIG thanks you to my wife and daughter for their patience love and support (especially your patience!). You never fail inspire me. I would also like to thank my sister who has been by my side at all times.

Finally, I would like to thank the OU database group for their support and wish them good luck in their endeavors. Go OUDB!

TABLE OF CONTENT

ABSTRACT

The Internet and advances in wireless communication technology have transformed many facets of the computer environment. Virtual connectivity through the internet has lead to a new genre of software systems, i.e., cooperating autonomous systems - systems that cooperate with each other to provide extended services to the user. Multidatabase systems - a set of databases that cooperate with each other in order to provide a single logical view of the underlying information - is an example of such systems. Advances in wireless communication technology dictate that the services available to the wired user be made available to the mobile user.

This dissertation studies transaction management in the mobile Multidatabase environment. That is, it studies the management of transactions within the context of the mobile and Multidatabase environments. Two new transaction management techniques for the mobile Multidatabase environment i.e., the PS and Semantic-PS techniques are proposed. These techniques define two new states (Disconnected and Suspended) to address the disconnectivity of the mobile user. A new Partial Global Serialization Graph algorithm is introduced to verify the isolation property of global transactions. This algorithm verifies the serializability of a global transaction by constructing a partial global serialization graph. This algorithm relies on the propagation of (serialization) information to ensure that the partial graph contains sufficient information to verify serializability of global transactions. The unfair treatment of mobile transactions due to their prolonged execution time is minimized through pre-serialization. Pre-serialization allows mobile transactions to establish their serialization order prior to completing their execution.

Finally, analytical evaluation and simulation is carried out to study the performance of these techniques and to compare their performance to that of the Kangaroo [DHB97] technique. Although the PS and Semantic-PS techniques enforce the isolation property, the evaluation results establish that the service time for these techniques in not significantly greater than that of the Kangaroo technique. In addition, the simulation establishes that pre-serialization effectively minimizes the unfair treatment of mobile transactions.

Chapter 1

PROBLEM STATEMENT

1.1 Objective

On August 19-21, 1998, a group of 16 distinguished database system researchers from academe, industry, and government including J. Gray, M. Stonebraker, P. Bernstein, H. Garcia-Molina, and J. Ullman met at Asilomar, California, to assess the database system research agenda for the next decade. The goal of the meeting was to discuss the current database system research agenda and to report their recommendations. The group discussed their recommendations in [Bernstein et.al.98] where they encouraged the database community to eschew the incremental, "delta-X" research that focuses on improving some widely understood idea X. Instead, they challenged the database community to explore problems whose main applications are decades off, and to pursue highly innovative and speculative research. In fact, the "grand challenge" proposed by the group is "Make it easy for everyone to store organize access and analyze the majority of human information on-line". Although the research documented in this dissertation commenced long before the Asilomar meeting, its contributions are in fact, a direct response to this grand challenge.

In the Asilomar report the authors state that in the future, billions of web clients will be accessing millions of databases, and that the Web will be one large federated system. This research studies transaction management in a multidatabase environment that supports both static and mobile users. The multidatabase architecture defined in this research resembles the federated system of the Asilomar report. The primary objectives of this research are fourfold: first, to identify the issues related to transaction management in a multidatabase environment that supports both static and mobile users; second, to develop two transaction management techniques that addresses all identified issues; third, to develop analytical and simulation

models and to evaluate the performance of the proposed techniques and to compare their performance to that of techniques existing in the current literature; and fourth, to develop guidelines to help users and future researchers.

1.2 Organization

This dissertation is divided into seven chapters. The following paragraphs provide an overview of each chapter in the dissertation:

The problem statement is presented in the remainder of this chapter. First, the mobile computing environment and the mobile computing architecture will be discussed. Next, the issues related to transaction management in the mobile multidatabase environment will be identified. These issues fall into three categories: multidatabase design restrictions; the ACID properties; and disconnection and migration issues.

Chapter 2 presents the state-of-the-art of related work. Specifically it discusses semantic atomicity and serializability theory, advanced transaction models and existing transaction management techniques that are applicable to the mobile multidatabase environment.

Two transaction management techniques that address all the issues will be developed in Chapters 3 and 4. The Pre-Serialization (PS) transaction management technique will be developed in Chapter 3. The shortcomings of the PS technique will be identified and a Semantic-PS technique that overcomes these shortcomings will be developed in Chapter 4.

The performance of these techniques will be evaluated in Chapters 5 and 6. Analytical models will be developed in Chapter 5 and used to study the performance of the PS and Semantic-PS techniques. Simulation models will be developed in Chapter 6 and used to validate the evaluation in Chapter 5. The evaluation will also develop guidelines to assist future researches.

This dissertation will be concluded in Chapter 7. First, concluding remarks of the author will be presented. This will be followed by a discussion of future research issues.

1.3 Motivation

The Asilomar Report [Bernstein et.al.98] predicts that in the future billions of users will access millions of databases in order to access and analyze the vast information available on-line. This multidatabase environment consists of a set of autonomous databases connected to a fixed (wired) network that cooperate with each other to provide extended services to users. For example, users will be able to verify entire travel itineraries that include round-trip airline tickets, hotel reservations, and rental car reservations, all in one transaction. Obviously, such a transaction will need to access multiple independent database systems. The rapid advances in wireless technology and the availability of mobile palmtops dictates that that the services available to the static user be made available to the mobile user. It is also expected that millions of users will be carrying mobile computers often called personal assistants, to carry out their day to day activities [IB94]. Each mobile computer will be equipped with a wireless connection to the information networks [IB94]. The mobile user will demand access to the information on the fixed system from anywhere and at any time. The multidatabase environment is no exception.

The distinguishing characteristic of mobile computing is the wireless communication medium that makes it possible for a mobile user to communicate with a static (wired) computer system through some wireless communication medium. In today's busy, technology dominated and communication intensive business environment, wireless computing offers numerous possibilities for the multidatabase environment.

For example assume the following scenario. A business traveler is commuting on a commercial airline from city A to city B. During the flight, the commuter decides to invest his

or her annual bonus in the Stock Market. This person will first access some information systems to determine the best investment opportunity. Once a determination is made, the person will need to execute a global transaction that accesses his or her personal bank account (or brokerage account) to obtain the funds, the NYSE database to execute the sale, and the sellers account to deposit the value of the stock. The person may also need to access some personal database to record information on the transaction.

The transaction manager is a vital component of any Database Management System (DBMS). It is responsible for providing reliable and consistent units of computing to users. The characteristics of the mobile computing environment affects the conventional responsibilities of the transaction manager. They introduce new issues that need to be addressed by the transaction manager, i.e., disconnection and migration. The wireless communication medium is characterized by frequent disconnections that occur during the execution of a user session. These disconnections cannot be treated as communication medium failures that result in aborted transactions as in conventional wired systems. The ability to migrate during the execution of a user session is unique to the mobile computing environment. In order to accommodate mobile users, the transaction manager of the Mobile MultiDataBase System (MMDBS) needs to view disconnection and migration as routine events that occur during the normal course of execution of a transaction. However, in some cases disconnection may represent unrecoverable failures. Upon disconnection the MMDBS needs to determine the status of the user. If the user is expected to reconnect, the transaction should be temporarily suspended. If the user is not expected to reconnect, the transaction may be aborted. Erroneous decisions about the status of the disconnected user are likely to be made as the actual status can only be predicted after disconnection. Thus, such transactions should not be aborted until they interfere with the execution of other transactions. Upon reconnection, suspended transactions should be allowed to resume execution from the point of suspension. Further, transactions should be allowed to

resume execution from a location different than the location at which the user was situated prior to the disconnection.

In addition, disconnection and migration affect the execution behavior of the system. Disconnection and migration prolong the execution time of transactions of mobile users. Also, mobile transactions are expected to be interactive by nature, i.e., pause for input from the user [Chry93]. For example, the stock broker will determine the quantity of shares to be purchased only after verifying the price per share. Thus, Long Lived Transactions (LLTs) [PB94] need to be supported. The length of execution affects concurrency control as well. The probability of a transaction conflicting with other transactions in the system is proportional to the length of execution of that transaction. As a result, transactions of mobile users are more likely to cause a consistency violation [DG98] and therefore, are more likely to be aborted. In order to maintain a notion of fairness, the transaction manager needs to minimize this victimization of mobile transactions due to their prolonged execution.

Existing Mobile multidatabase (MMDB) transaction management techniques that are found in the literature address some of the issues that have been identified. However, none of the techniques addresses all these issues. In fact, all reviewed techniques fail to address the unfair treatment of mobile transactions due to their prolonged execution, nor do they ensure the consistency of the transactions. In order to facilitate mobile users access to the information systems available on-line, it is necessary to re-visit transaction management issues in this new environment and to provide solutions that address all requirements.

1.4 The Mobile Computing Environment

The general mobile computing model consists of two distinct sets of entities: a fixed network system and a continuously changing set of mobile hosts (Figure 1-1). The fixed networking system consists of a collection of static computers connected by a wired network. Some units on the static network have the capability of communicating with the mobile units through a wireless medium. These units are called base stations or mobile support stations (MSS). The area covered by an MSS is called a cell [PB95]. The wireless communication medium between the MSS and the mobile user includes cellular architecture, radio transmission over FM, satellite services, and wireless LAN. Although current wireless communication technology is fairly reliable, it is not as robust as the communication mediums used in the static systems. It is also limited in bandwidth compared to wired networks. During the course of execution the mobile user is likely to migrate from cell to cell. The mobile user will be connected to no more than one MSS at any given time. The process involved in transferring a user from one MSS to another is called a hand-off.

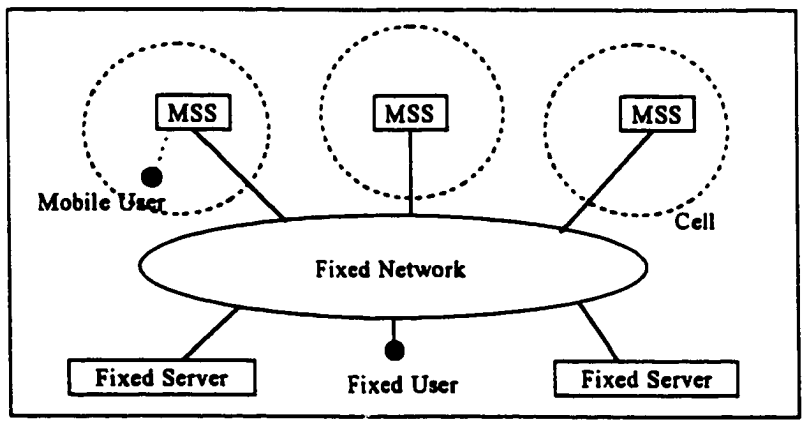


Figure 1-1: The Mobile Computing Environment

The mobile hosts are portable computers that vary in size, processing power, memory, etc. The typical mobile computer will have limited resources compared to its desktop counterparts [PB95-2]. These limitations include battery power, processing power, volatile memory, disk space, network bandwidth, etc. Due to the unreliability of the communication medium as well as limited resources available, the mobile user will be characterized by frequent disconnections and will operate in one of many modes ranging from highly connected to disconnected. However, a characteristic of these modes of operation is that they are foreseeable [PB93]. For example, the MSS will be able to predict that the user is going out of range by monitoring the strength of the signal. On the other hand, if the user decides to disconnect in order to conserve scarce resources, the MSS could easily be informed of this decision prior to disconnection.

1.5 The Mobile Multidatabase Architecture

A multidatabase system (MDBS) is a collection of autonomous database systems (called local database systems, or LDBSs) that are connected to a fixed network (Figure 1-2). In many cases, an MDBS is the result of shifting priorities, and the need of an organization to be part of a larger information system [ERS98]. The need to be part of a larger information system arises primarily for two reasons: one, organizations may acquire or merge with other organizations creating the need for a new global information system; and two, competition forces organizations to take advantage of the Internet to provide cooperating information systems that cater to the growing information needs of users.

In the MDBS, the respective LDBSs retain complete control over their databases. Each autonomous database may be viewed as an independent site in the network. These databases operate in different environments, and may use different data models, data manipulation facilities, transaction management and concurrency control mechanisms, etc. [GR93]. Existing

users - referred to as local users - will continue to access these databases through their respective LDBSs. The execution of local transactions submitted by local users will be transparent to any external process. Users who simultaneously access multiple databases - referred to as global users - do so by submitting global transactions to the multidatabase Management System (MDBMS).

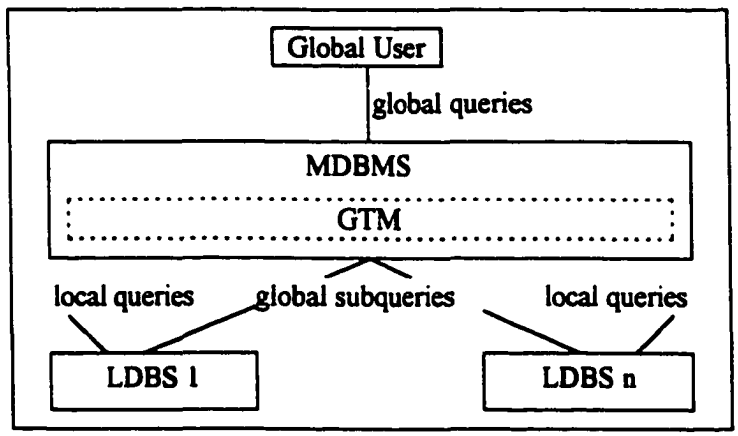


Figure 1-2: Multidatabase System

The MDBMS is a set of software modules existing on the fixed network that cooperates with the local LDBSs in order to project an illusion of a single database to the global user. Global users are allowed only limited access to the individual databases. For example, although global users will be allowed to make reservations on a commercial airline database system, they will not be allowed to execute ad-hoc queries that could compromise sensitive information. Each local database provides a service interface that specifies the operations accepted by the LDBS and the services provided to the MDBMS. The Mobile Multidatabase system (MMDBS) is simply an MDBS that supports both static and mobile users. The database management system or DBMS of an MMDBS is referred to as a Mobile Multidatabase Management System (MMDBMS).

The Global Transaction Manager (GTM) is a software component of the MMDBMS that manages the execution of global transactions. A global transaction consists of a set of queries, each of which is a legal operation accepted by some service interface of an LDBS in the system. Queries of a global transaction may be grouped together to form logical units of execution called sub-transactions. Any subset of queries of a global transaction that access the same LDBS may be executed as a single transaction with respect to that site and will form a logical unit called a site-transaction. As users migrate from one MSS to another, queries of a global transaction may be submitted from different MSSs (Figure 1-3). Such transactions will be referred to as migrating transactions. The notation Q^k_j is used to represent the j^{th} query of global transaction k.

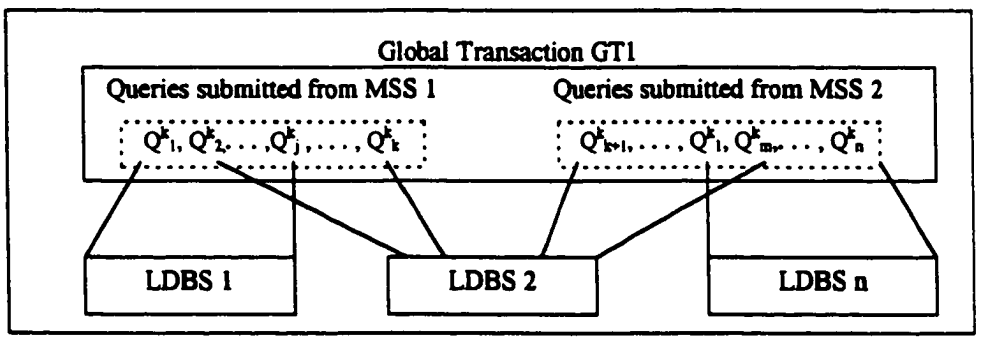


Figure 1-3: Migrating Global Transaction

1.6 Transaction Management Issues

In database theory, a transaction is defined as an independent, consistent and reliable unit of computing [PB95]. The definition of a transaction gives a strong indication of the primary responsibilities of the transaction manager, i.e., to provide consistent and reliable access to the data within its domain. Generally, this can be achieved by enforcing the ACID (Atomicity, Consistency, Isolation, and Durability) properties [GR93]. *Atomicity* requires that

either all operations of a transaction execute successfully or none at all. *Consistency* requires that a successful transaction not violate any consistency constraints defined on the database. *Isolation* requires that the effect of executing a set of transactions concurrently be equivalent to that of executing the same set of transactions in some serial order. *Durability* requires that all changes made by a successful transaction be permanently reflected in the database. However, the applicability of ACID in the MMDB environment has been questioned. Not that ACID is unenforceable, but because it is expected that ACID will be enforced using too many aborts, resulting in a system that is perfectly consistent, but gets only a small fraction of useful work done [DHB97]. In addition to providing consistent and reliable access to the data, the Mobile MultiDataBase System (MMDBS) needs to address disconnection and migrating transactions. These reasons provide the motivation to revisit the requirements of the GTM in the MMDB environment. In the remainder of this section, a more detailed discussion of these issues will be provided.

1.6.1 Multidatabase Design Restrictions

Local autonomy is the main feature that distinguishes multidatabase systems from conventional distributed database systems [BMS92]. Local autonomy dictates that no changes can be made to the local DBMSs in order to support the multidatabase system. A distinction can be made between structural design and execution aspects of local autonomy [ERS98]: Structural design autonomy refers to the ability of an LDBS to choose its own design with respect to issues such as data model, query language, etc.; Execution autonomy refers to the ability of an LDBS to execute transactions without interference. The MMDBMS cannot violate the structural design and execution autonomy of the local LDBSs. Local autonomy can be violated in four different ways:

- Preservation Infringement (PI) - The GTM requires that modifications be made to LDBSs and existing (local) software.
- Execution Infringement (EI) - The GTM infringes upon the execution freedom of the LDBS. For example, an LDBS may be prevented from aborting a site-transaction that executed at that site.
- Security Infringement (SI) - An LDBS is not allowed to control access to one or more data items within its domain.
- Transparency Infringement (TI) - The GTM requires the LDBSs to furnish control information such as serialization graphs.

The multidatabase environment give rise to other issues that need to be addressed as well. The vast number of LDBSs that could potentially be part of a MMDBS, the autonomy of the LDBSs, and the geographic distance that may separate the LDBSs make centralized algorithms or even distributed algorithms that require the cooperation of all sites, practically unacceptable. The global transaction management schemes that provide consistent and reliable units of computing to global users need to be de-centralized in nature, and need to minimize the cooperation required to perform its tasks.

1.6.2 The ACID Properties

As mentioned before, if a transaction is guaranteed to satisfy the ACID properties it is then a consistent and reliable unit of computing. Enforcing the ACID properties in the MMDB environment is compounded by three factors:

1. Enforcing the ACID properties in a distributed environment requires the cooperation of each site. For example, to enforce the atomicity property, the sites at which a global transactions executes site-transactions need to cooperate in order to ensure a consistent

final outcome i.e., global abort or commit. The autonomy requirement of the MMDBS limits the level of cooperation that can be achieved between the LDBSs.

2. Disconnection and migration of the mobile user alters the structure of (mobile) global transactions. For example, it prolongs the execution of a global transaction. This prolonged execution affects the behavior of the system, i.e., the transaction is likely to retain resources for longer periods of time, thus limiting concurrency.
3. The vast number of potential LDBSs that form a MMDBS and their geographic dispersion further limits the level of cooperation that can be achieved. It makes it practically impossible to implement any centralized algorithms or even distributed algorithms that require the cooperation of all sites.

As it is difficult to enforce the ACID properties in the MMDB environment, the applicability of ACID to this environment has been argued. Further, in [DH95] the authors make a compelling argument for providing unrestricted access to data in this environment: "Returning dirty data tagged with appropriate warnings is much more useful than returning an ABORT message ...". Thus, it is necessary for the transaction management process of the MMDBS to support a spectrum of correctness criterion with respect to the ACID properties. Next, each of the ACID properties will be discussed individually.

1.6.2.1 Atomicity

The atomicity property requires that either all operations of a transaction execute successfully, or that they are all aborted (and all changes made by the transaction are erased from the system). In the MMDB environment, atomicity requires that either all site-transactions of a global transaction execute successfully, or that they are all aborted. Thus, all sites at which a global transaction executes site-transactions need to corporate in order to ensure that the same outcome is recorded at all sites. It has been debated whether, in the MMDB environment, strict

atomicity can be enforced without violating local autonomy [BHS92]. One side of the argument is that the prepared-to-commit operation will become standard in most DBMSs and therefore provide the necessary cooperation to enforce strict atomicity. In essence, the local LDBS relinquishes its right to unilaterally abort the (site) transaction after the transaction enters the prepared-to-commit state. However as the transaction is not yet committed, it may be aborted if required to do so by the MMDBMS. The other side to the argument is that the prepared-to-commit operation causes an execution infringement upon the LDBS and that there will always be databases whose autonomy is critical and will not export the prepared-to-commit operation.

1.6.2.2 Isolation

The isolation property requires that the concurrent execution of any set of transactions be equivalent to some serial execution of the same set of transactions. That is, intermediate results of a transaction must not be visible to other concurrently executing transactions. Once again, enforcing the isolation property in the MMDB environment is difficult for two reasons:

1. Local transactions executed by the LDBSs are transparent to the MMDB system and therefore cannot be considered by any global algorithm.
2. The execution order of concurrent site-transactions of global transactions is not visible to the MMDB system.

In addition, to ensure that the local (LDBS) isolation property is not violated by a global transaction, it is necessary to execute all queries of a global transaction that access the same site as a single ACID transaction with respect to that LDBS. In other words, it is necessary to limit each global transaction to no more than one site-transaction per site. Else, the LDBS may execute local transactions between the separate site-transactions of a single global transaction. This violates the local isolation property as the local transactions are able to view intermediate results of the global transaction. This violation cannot be detected by the LDBS as

it views each site-transactions as a separate logical unit of computing. From the perspective of the MMDBs, these separate site-transactions belong to the same (global) logical unit of computing whose intermediate results should not be made visible.

As argued before, in order to provide unrestricted access to data in the MMDB environment, it is necessary to support a wide range of correctness criterion with respect to the isolation property. Note that global transactions with unrestricted access that write data back to the databases may compromise the accuracy of the databases as the transaction may have read inconsistent data. If the system demands that the correctness of the databases not be compromised, unrestricted access should be limited to read-only transactions. Thus, only the data returned to the user will be compromised. As this requirement is application specific, any proposed technique should provide a wide range of correctness criterion and let the designers of the individual MMDBSs define the level of correctness that needs to be enforced.

Disconnection and migration affect the execution time of a transaction which, in turn, affects the enforcement of the isolation property. As the duration of execution of a transaction increases, the possibility of the transaction conflicting with other transactions increases. If an optimistic concurrency control protocol - a protocol that checks for violations of the isolation property at the time of the transaction's commit - is used, transactions of mobile users are more likely to be aborted due to their prolonged execution time. If a pessimistic concurrency control protocol - a protocol that does not allow a transaction to violate the isolation property during its execution - is used, the average response time and throughput of the system will deteriorate as transactions will be blocked for extended periods of time by transactions of mobile users. In order to maintain a notion of fairness, the MMDB system needs to minimize the ill-effects caused by disconnection and migration.

1.6.2.3 Consistency and Durability

As a consequence of autonomy, we can assume that no data integrity constraints are defined on data items residing at different LDBSs [DE89]. As each LDBS will ensure that the site-transactions executed at its respective site do not violate any local integrity constraints, global transactions will, by default, satisfy the consistency property. Thus, the MMDB system is relieved of this responsibility. Similarly, the MMDB system can rely on the durability property of the LDBSs to ensure durability of committed global transactions.

1.6.3 Disconnection and Migration

Unlike disconnection in the static environment, disconnection in the mobile environment cannot be treated as failures that result in aborted transactions. The transaction manager of the MMDBS should allow transactions to be halted at arbitrary points during its execution. Upon re-connection, halted transactions should be allowed to resume execution from where they left off. In order to support migration, the transaction management process should allow halted transactions to resume execution from a location different from the location at which the previous execution was halted. All responses that cannot be delivered due to disconnection need to be logged by the MMDBS and be delivered to the user upon re-connection.

To fully support disconnection, it is not sufficient to simply allow disconnection to occur at arbitrary points during the execution of a transaction. In some cases, disconnection will be due to catastrophic failures, or catastrophic failures may occur during the period of disconnection. Halted transactions are not resumed after catastrophic failures. Therefore, the MMDBS needs to determine the status of its disconnected users periodically. When a catastrophic failure is deemed to have occurred, the MMDBS may terminate any associated transactions. Although disconnection is foreseeable, erroneous decisions are bound to be made

as the true status of the user cannot be verified after disconnection. Thus, not only should the MMDBS accommodate disconnection but should also minimize the affects of such erroneous decisions. These affects can be minimized by postponing the abort of a transaction until it obstructs the execution of other transactions.

Frequent disconnection and migration will prolong the execution time of global transactions. In addition, global transactions are expected to be interactive by nature, i.e., pause for input by the user [Chry93]. To support mobile users, the GTM will need to support long-lived Transactions (LLT) [PB94]. As concurrent transactions compete for resources, prolonged execution limits concurrency if resources obtained by transactions - such as locks - are not released in a timely fashion. The blocking of a transaction's execution must be minimized in order to increase concurrency [MB98]. Therefore, site-transactions should be allowed to commit early so that resources can be released immediately after the site-transaction has completed its execution.

1.7 Summary of Transaction Management Issues

The transaction management process of the MMDBS needs to enforce the atomicity and isolation properties with respect to global transactions. In fact, it is necessary to provide the functionality to enforce a range of correctness criterion with respect to atomicity and isolation. Disconnection needs to be viewed as an event that occurs during the normal execution sequence of a transaction. To support migration, the disconnected user should be allowed to resume execution from a different location. Untimely abortions caused by erroneous decisions on catastrophic failures to mobile users needs to be at least minimized, if not eliminated. Any ill-affects due to the prolonged execution caused by disconnection and migration needs to be minimized. In addition, the autonomy of the LDBSs must not be compromised. Also, any

algorithms used in the MMDB environment need to be de-centralized in nature and need to minimize the level of cooperation/coordination required by the sites.

Note that, as the local databases are autonomous, site-transactions are executed independent of each other by the respective LDBSs. As a result, each site-transaction can be considered as a consistent and reliable unit of computing with respect to that LDBS. That is, each site-transaction is guaranteed to be an ACID unit of computing with respect to the LDBS at which it executed. This does not guarantee that global transactions will be ACID with respect to the global database. However, the local ACID nature of the site-transactions can be exploited by the MMDBS in order to provide globally ACID (global) transactions.

1.8 Contributions

This section summarizes the major contributions of this dissertation. This research proposes two new transaction management techniques for the mobile multidatabase environment. These techniques are based on the multi-level transaction model. As the multi-level transaction model requires all sub-transactions be compensatable, these techniques require sub-transactions of a global transaction to be compensatable.

The proposed transaction management techniques introduce three new concepts. First, it introduces the notion of suspended execution of transactions. Suspended transactions are not aborted until they interfere with the execution of other transactions. As the status of a disconnected user can only be predicted, suspending the execution of global transactions instead of aborting their execution minimizes erroneous aborts. Second, it introduces pre-serialization which is used to minimize the unfair treatment of mobile transactions. Pre-serialization allows mobile transactions to establish their serialization order prior to completing their execution. Third, it introduces the Partial Global Serialization Graph (PGSG) algorithm that enforces the atomicity and isolation properties of global transactions. This algorithm is unique in that it

verifies the serializability of a global transaction by constructing a partial global serialization graph - a serialization graph that represents *only* a subset of the global serialization scheme. The PGSG algorithm relies on serialization information propagation in order to ensure that the all isolation property violations are detected.

As new algorithms and concepts are proposed, extensive analysis and simulation is carried out. This research develops analytical and simulation models that can be used to study transaction management in the mobile multidatabase environment.

The analytical and simulation experiments establish that the cost of enforcing the isolation property is minimal. The simulation results also indicate that the concept of pre-serialization minimizes the unfair treatment of mobile transactions due to their prolonged execution.

Chapter 2

LITERATURE REVIEW

In this section, relevant work will be discussed. As stated in Section 1.6.2.1, it has been argued whether strict atomicity can be enforced in the multidatabase environment without violating local autonomy. Without taking sides in that argument, the techniques proposed in this research will base its correctness criterion on semantic atomicity [LKS91] - an alternate criterion that is more suitable for the MMDB environment. Semantic atomicity will be discussed in Section 2.1. Serializability will be the correctness criterion used to enforce the isolation property in the proposed techniques. Serializability will be discussed in Section 2.2. A (global) transaction in the MMDB environment is a collection of (site) transactions that are executed as independent (local) transactions by the LDBSs. However, the global transaction needs to be executed as a reliable and consistent unit of computing by the MMDBS. The flat transaction model used in traditional databases is not suitable for the MMDB environment as it provides only one level of control. The transaction model proposed in this research is based on the Nested transaction model [Moss81] which will be introduced in Section 2.3. Finally, seven transaction management techniques that are applicable to the MMDB environment will be summarized in Section 2.4.

2.1 Semantic Atomicity

In order to enforce conventional atomicity, the GTM must ensure that either all sub-transactions of a global transaction are committed or that they are all aborted. Enforcing atomicity is difficult as each (autonomous) site retains the right to abort a (site) transaction executed under it's supervision at any time prior to a successful commit at that site. An alternate

criterion to conventional atomicity is *semantic atomicity* [LKS91]. Semantic atomicity requires one of the following conditions to be satisfied by each (global) transaction:

1. Either all sub-transactions are aborted or each sub-transaction is committed or retried;
2. Either all sub-transactions are committed or each sub-transactions is aborted or compensated for.

A compensatable transaction is a transaction whose effects can be undone after it has committed by executing a compensating transaction. For example, a sub-transaction that reserves a seat in an airline reservation system is compensatable as reservation can be canceled which, in effect, undoes the reservation. A re-triable sub-transaction is one which is guaranteed to succeed if retried a sufficient number of times. For example, a sub-transaction that credits a bank account is a re-triable sub-transaction as money can always be credited to a bank account provided that the account exists.

Semantic atomicity is more suitable than conventional atomicity for the multidatabase environment for two reasons: First, semantic atomicity is easier to implement in this environment as it does not require the cooperation of the autonomous sites in order to ensure that either all sub-transactions commit or that they all abort. The following cases will illustrate this point:

- Let us take the case where all sub-transactions are re-triable. Then, if an LDBS decides to abort a sub-transaction, then this sub-transaction can be retired until it executes successfully, satisfying condition 1 of semantic atomicity.
- Let us take the case where all sub-transactions are compensatable. Then, if an LDBS decides to abort a sub-transaction, then all committed sub-transactions can be compensated and all active sub-transactions are aborted, satisfying condition 2 of semantic atomicity. This, in effect, erases the entire execution of the transaction from the global system.

Second, semantic atomicity is ideally suited for the MMDB environment as it allows site-transactions to commit independently, releasing (local) resources held by that site-transaction in a timely fashion. This increases local as well as global concurrency.

The transaction management techniques proposed in this research implement condition 2 of semantic atomicity. Condition 2 is chosen for the following reasons:

- Most transactions executed on the Internet, a prevalent multidatabase environment, are inherently compensatable. For example, any reservation type transactions are compensatable as the reservation can be cancelled; most purchase type transactions are compensatable as most purchases are not final as most purchases can be cancelled within a certain time period.
- In a concurrent environment, strict isolation cannot be enforced under condition 1 of semantic atomicity. The atomicity and isolation property of a transactions cannot be verified until the transaction completes the execution of its last sub-transaction. However under condition 1, a transaction may commit at any point at which all outstanding sub-transactions are retriable. Although the transaction may not violate the isolation property at the point of commit, a consequent retriable sub-transaction may violate the isolation property. This violation can only be resolved by aborting one or more of the transactions involved in the isolation property violation. If all transactions involved in the violation have already committed, the violation cannot be resolved.

2.2 Serializability Theory

Serializability is the most frequently used correctness criterion to verify isolation [OV91]. To state it simply, the execution of a set of transactions is said to be serial if all operations of each transaction are executed consecutively [Ullm88]. The concurrent execution of a set of transactions is said to be *serializable* if its effect is equivalent to that of some serial

schedule of the same set of transactions. In this section, a formal definition of serializability will be provided. First, it is necessary to introduce some basic terminology. The terminology defined in this section is consistent with the definitions in [BHG87].

Definition 2.1: Let $T = \{T_1, T_2, ..., T_n\}$ be a set of transactions. A ***history*** H over T is a partial ordering with respect to ordering relation $\rightarrow$ such that:

1. H contains precisely the operations submitted by T, i.e., all operations of $T_1, T_2, ..., T_n$;
2. H honors all operation orderings specified by each transaction in T, that is, if operation Ok_i appears before Ok_j in transaction T_k, then Ok_i appears before Ok_j in any history that contains Tk; and
3. For every pair of conflicting operations O_i and O_j where O_i appears before O_j in the execution order of T, then $O_i \rightarrow O_j$ is in H.

To illustrate this definition, consider the following example.

Example 2.1: Let $T = \{T_i, T_j\}$ be a set of (two) transactions such that:

$T_i \Rightarrow R_i(a), R_i(b), W_i(a), W_i(b), C_i$

$T_j \Rightarrow R_j(a), W_j(a), C_j$

where R(x) represents a read operation on x, W(x) represents a read operation on x, and C represents a commit operation. Assume that the operations of T are executed in the following order:

$R_i(a), R_i(b), W_i(a), W_i(b), C_i, R_j(a), W_j(a), C_j$

Then, the history H over T is shown in Figure 2-1.

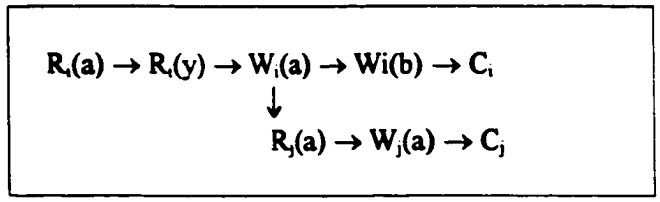


Figure 2-1: History over transaction set T

Definition 2.2: The execution of a set of transactions is said to be ***serial*** if the transactions are executed in some serial order.

Note that the execution of transactions T_i and T_j in Example 2-1 is serial.

Definition 2.3: Let O_i and O_j be two operations in transactions T_i and T_j respectively. O_i and O_j are said to ***conflict*** (directly) if O_i and O_j both access the same data object X and at least one is a write operation that modifies the value of X.

Note that operations $W_i(a)$ and $R_j(a)$ in T conflict as they access the same data object and one operation is a write. Also, if O_i and O_j conflict and O_j and O_k conflict, then O_i and O_k are said to conflict indirectly.

Definition 2.4: Two histories H and H^ are said to be ***conflict equivalent*** if they are defined over the same set of transactions and conflicting operations (of non aborted transactions) are ordered in the same way in both histories.

Definition 2.5: The concurrent execution of a set of transactions is said to be ***serializable*** if its history is conflict equivalent to some serial schedule of the same set of transactions.

The following example will be used to illustrate a non serial but serializable schedule.

Example 2.2: Let $T = \{T_i, T_j\}$ be a set of transactions as described in Example 2-1. However, in this case, assume that the operations of T are executed in the following order:

$R_i(a), R_i(b), W_i(a), R_j(a), W_j(a), C_j, W_i(b), C_i$

The history H^ is said to be serializable as it is defined over the same set of transactions T and H^ is conflict equivalent to the history H of the serial execution of T. In fact, the history H^ of T in Example 2-2 is the same as the history H in Example 2-1. Note that non conflicting operations in a history can be rearranged in any order without affecting the outcome of any of the transactions.

Definition 2.6: Two transactions T_i and T_j are said to ***conflict*** if they contain conflicting operations.

Definition 2.7: The serialization graph (SG) for a history H, denoted by SG(H), is a directed graph whose nodes represent committed transactions in H and whose edges $T_i \rightarrow T_j$ $(i \neq j)$ represent conflicting transactions T_i and T_j such that the conflicting operation of T_i precedes the conflicting operation of T_j.

Example 2.3: Let H be the history given in Figure 2-1. Then SG(H) is the graph $T_i \rightarrow T_j$.

Next, a fundamental theorem of serializability is presented. A rigorous proof of this theorem can be found in [BHG87].

Theorem 2.1: A history H is said to be serializable iff SG(H) is acyclic.

The serializability theory presented above cannot be applied directly to the multidatabase environment as this environment consists of two levels of execution. At the local level, each local serializability graph SG_l consists of local transactions and site-transactions of global transactions that have been submitted to that site. At the global level, the global serializability graph SG_g consists of global transactions. The global serializability graph does not contain any local transactions as the execution of local transactions at the LDBS is transparent to the MDBS. In order for the execution of a set of local and global transactions to be serializable in the multidatabase environment, the following conditions must be satisfied [MRKS92]:

1. The local serialization graphs must be acyclic, that is, the concurrent execution of local and global transactions within each local LDBS must be serializable; and
2. The global serialization graph must be acyclic, that is, conflicting global transactions must be serialized in the same order at all sites at which they conflict.

A simple example will be used to illustrate condition 2:

Example 2.4: Let S_1 and S_2 be two sites in a multidatabase system where S_1 contains data item a and S_2 contains data item b. Let GT = $\{GT_i, GT_j\}$ be two global transactions such that

$GT_i \Rightarrow R_i(a), W_i(b), C_i$ and

$GT_j \Rightarrow R_j(a), W_j(a), R_j(b)\ C_i$

Assume that the operations of GT_i and GT_j are executed by the respective LDBSs in the following order:

$S_1 \Rightarrow R_i(a), R_j(a), W_j(a), C_i, C_j$ and

$S_2 \Rightarrow W_i(b), R_j(b), C_j, C_i$

Then, GS_g over GT is given by the graph $GT_i \rightarrow GT_j$ as the execution of conflicting operations of GT_i precedes the execution of conflicting operations of GT_j at both sites. However, if the operations of GT_i and GT_j are executed in the following order:

$S_1 \Rightarrow R_i(a), R_j(a), W_j(a), C_i, C_j$ and

$S_2 \Rightarrow R_j(b), W_i(b), C_j, C_i$

then GT_i and GT_j are not serializable and SG_g contains a cycle.

Each LDBS guarantees that all transactions (local and site-transactions) executed at that site do not violate the ACID properties. Thus, it is assured that the concurrent execution of transactions within each LDBS is Serializable and, therefore, condition 1 is satisfied by all LDBSs of the MMDBS. However, since the GTM has no knowledge of the execution order of site-transactions within the local LDBSs, condition 2 cannot be verified without taking additional steps to determine the serialization order of site-transactions within each site.

2.3 Advanced Transaction Models

In the traditional flat transaction model, a transaction consists of a begin operation, a set of read and write operations that are executed sequentially, followed by a single commit operation or an abort operation that un-does the effect of the entire transaction. All committed transactions are required to satisfy the ACID properties. This model is called flat because there is only one level of control [GR93]. Either all operations succeed and the transaction is committed, or all operations are aborted. Flat transactions are the simplest type of transactions [GR93]. It does not provide the flexibility required for the multidatabase and mobile computing environments. It does not allow sets of operations to be executed as independent transactions under the supervision of autonomous DBMSs. As a transaction needs to be executed as a single atomic unit, atomicity cannot be enforced in the multidatabase environment without the cooperation of the constituent DBMSs. This model does not allow for a wide range of correctness criterion with respect to the atomicity and isolation properties to be supported.

To overcome these limitations, the nested transaction model has been proposed [Moss81, MMP83]. A nested transaction consists of a set of sub-transactions each of which is either a nested transaction or a flat transaction. Therefore, the whole transaction forms a tree and is called a transaction tree. The transaction at the root of the tree is called a *top-level transaction*; others are called sub-transactions. Sub-transactions at the leaf level are flat transactions. A sub-transaction's predecessor is called a *parent*; a sub-transaction at the next lower level is called a *child*. In [Moss81], primitive database operations can only be contained within the leaf level sub-transactions. In [GR93], this restriction is not observed. In this research, the model defined in [GR93] will be followed as it is less restrictive. The nested transaction model allows potential internal parallelism to be exploited. It contains a control structure that allows operations to be grouped together and executed independently.

Modifications to the commit protocol of the nested transaction model have been proposed. Each modification gives rise to a variation of the nested transaction model. These variations are detailed below:

2.3.1 The Basic Nested Transaction Model

In this model, each sub-transaction is committed or aborted independently [GR93]. However, its commit does not take effect unless its parent transaction commits. Therefore, by induction, a sub-transaction can finally commit only if the top-level transaction commits [GR93]. When a sub-transaction is committed, its results are made accessible only to its parent. If a sub-transaction is aborted, then all its (child) sub-transactions are aborted regardless of their local commit status. Note that a sub-transaction may commit even if one or more of its sub-transactions are aborted.

As the commit of sub-transactions do not take effect until the top-level transaction is committed, this model poses a major limitation with respect to the MMDB environment. That

is, access to the data items modified by the committed sub-transactions needs to be regulated by the GTM. This cannot be achieved without violating local database autonomy.

2.3.2 The Open Nested Transaction Model

The open nested transaction model [GR93] is a liberal version of the nested transaction model that allows each sub-transaction to commit early, independent of the outcome of its parent transaction. Thus, a sub-transaction may remain committed even if the parent transaction is aborted.

This model eliminates the limitations of the basic nested transaction model. That is, it allows sub-transactions' to commit independently in the multidatabase environment as the outcome of sub-transactions commit status is not influenced by the outcome of their parent. However, this model violates the atomicity property of transactions. The results of a sub-transaction may persist in the database even if the top-level transaction is aborted.

2.3.3 The Multi-Level Transaction Model

The multi-level transaction model is an extension of the nested transaction model [GR93]. This model allows sub-transactions to commit early as well. However, it assumes the existence of a compensating transaction that can semantically reverse the effects of committed sub-transactions in the event that the parent transaction is aborted. The compensating transaction guarantees that all updates made by committed sub-transactions can be revoked in the event that the top-level transaction fails.

This model is ideally suited for the multidatabase environment for three reasons:

1. It allows sub-transactions to commit early;
2. Atomicity can be enforced without the cooperation of the autonomous DBMSs as commits of sub-transactions can be reversed;

3. It allows a wide range of correctness criteria - with respect to the atomicity and isolation properties.

2.4 Transaction Management Techniques

In this section, a brief review of transaction management techniques applicable to the MMDB environment found in the current literature will be carried out. These techniques will be examined with respect to disconnection and migration support, LLT support, enforcement of the atomicity and isolation properties, and local database autonomy violations. The review will be summarized in Table 2-1.

In this section, the terms *Full* and *Partial* will be used to express the level of disconnection and migration support provided by each technique. Partial will be used to indicate that the technique allows for disconnection/migration but does not address all the related issues. The term Full will be used to indicate that the technique addresses all related issues. The issues related to disconnection are:

1. Arbitrary disconnection should be supported;
2. Disconnection that represent catastrophic failures need to be addressed;
3. Any ill-effect caused by the extended duration of transaction execution needs to be minimized.

The issues related to migration are:

1. Transactions should be allowed to halt their execution at one MSS and resume their execution from another MSS at arbitrary points;
2. Any ill-effect caused by the extended duration of transaction execution needs to be minimized.

VAR, *STR* and *None* will be used to express the level of atomicity/isolation provided by each technique. VAR states that a spectrum of correctness criteria is supported; STR states that

only strict atomicity/isolation is supported; and None states that atomicity/isolation is not enforced.

2.4.1 An Agent Based Approach

The technique presented in [PB95-2] is based on agent-based distributed computing. An agent is an object that encapsulates data and procedures that the receiving computer executes. Formally, an agent is a quadruple (D, M, SD, P), where D is a set of data, M is a set of methods, SD is a set of structural dependencies, and P is a set of break and relocation points. A global transaction can be visualized as an agent that consists of sub-agents. Agents may be submitted from various sites including mobile stations. Agent-based computation is decentralized as the agents themselves communicate with each other in order to provide consistent and reliable computing. A set of structural dependencies allows the user to define critical methods that, upon failure, cause the entire agent to fail. It can also be used to define compensating methods that are executed to compensate for already committed methods. Thus, this model supports early commits of its sub-agents and the spectrum of atomicity. In order to support migration, relocation points are pre-defined within the agent. This does not allow for arbitrary relocation of the mobile user. As a result, it does not fully support migration. The executions of Agents can be isolated from each other by ensuring that concurrent execution occurs within the pre-defined breakpoints. Yet, the isolation property cannot be enforced globally as the execution of local transactions is transparent to the external agents. This technique does not violate any local autonomy requirements.

2.4.2 The Cluster Model

Although [PB95] focuses on a distributed database with replication under central control, it introduces many interesting ideas applicable to the multidatabase environment as

well. In this technique, closely located data is grouped as a cluster. Clusters are defined dynamically as mobile users connect and disconnect at different locations. Full consistency is maintained within a cluster. However, different degrees of consistency are defined for replicated data located at different clusters. This will increase the availability of data and consequently performance at the cost of consistency. Transactions are defined as either strict or weak. Strict transactions are allowed to access only consistent copies of a data item and, thus, produce only consistent results. Weak transactions are allowed to access copies of data items that are inconsistent within an acceptable limit and, thus, may also produce inconsistent results. If only strict transactions are allowed, the isolation property is ensured and all copies of a data item are consistent. If weak transactions are allowed, multiple inconsistent copies of a data item are produced. Weak transactions provide the necessary flexibility to support a spectrum of atomicity and isolation criterion. Weak transactions are first committed at the cluster level and then at the global level. As weak transactions committed at the cluster level may be aborted during the global commit phase, weak transactions may be used only when compensating transactions are available; otherwise, cascading aborts of weak transactions could occur.

As a cluster maintains its own copy of data items, LLTs may be executed as weak transactions within that cluster without the undesirable effects of data contention. As this technique focuses on distributed databases under central control, the autonomy restriction does not apply. However, it must be noted that this technique causes an EI weak transactions committed at the cluster level may be aborted during the global commit, and PI as this technique requires the local DBMSs to be modified to support clusters.

2.4.3 The Semantic-Based Transaction Processing Scheme

The semantic based transaction processing scheme [WC94] addresses transaction processing for the general mobile database environment in which the constituent sites may or

may not be autonomous. Here, the authors exploit the semantics of data objects and operations defined on them to support autonomous mobile transactions that are executed on the mobile hosts and to increase concurrency. A 'master copy' of each object resides on a stationary database server. These objects are split into disjoint fragments that are handed out to the mobile hosts which manipulate the fragments within defined consistency conditions. Upon completing the required operations, the fragments are returned to the server and combined with the rest of the data objects using a merge operation. Not all data objects can be fragmented and operated upon independently. Sets, stacks, and queues are a few examples of fragmentable objects. This scheme is limited in its applicability as it works only in environments where those data can be fragmented and operated upon independently.

Disconnected operations and LLTs are supported by allowing the mobile user to cache data objects required for computation on the local machine. Communication cost is minimized as only the (fragmented) portion of the data objects required for the computation is obtained by the mobile host. As different consistency conditions may be specified for operating upon the fragmented objects, the full spectrum of atomicity can be supported. However, as the objects are split into disjoint fragments, only strict isolation can be supported. As this technique is not designed specifically for the multidatabase environment, it violates the autonomy of the local databases. Modifications need to be made to the local DBMSs in order to support fragmentation. In addition, this technique requires the cooperation of the local DBMSs - an EI violation.

2.4.4 Reporting Transactions and Co-Transactions

The technique proposed in [Chry93] is based on the open nested model and supports two additional types of transactions, namely, reporting transactions and co-transactions. These new types of transactions allow concurrent global transactions to share partial results improving

concurrency. In this model, sub-transactions can be committed or aborted independently. Each sub-transaction is either compensatable or non-compensatable. Non-compensatable sub-transactions are not allowed to commit their effects on objects when they commit. This is an EI with respect to the autonomy requirements. Sub-transactions are further categorized as either vital or non-vital. A transaction can commit only if all its vital sub-transactions commit and only after the statuses of non-vital sub-transactions are known. Thus, this model supports a spectrum of atomicity. The authors assume that sub-transactions of different global transactions can interleave their execution in any arbitrary order, eliminating the need to address the isolation property.

2.4.5 The Multidatabase Transaction Processing Manager Technique

The multidatabase Transaction Processing Manager (MDSTPM) technique proposed in [YZ94] is based on a Message and Queuing Facility (MQF) to manage global transactions submitted by mobile workstations. The site that a global transaction is initiated is designated as the coordinator site for that transaction and schedules and executes the transaction on behalf of the mobile unit. Transactions submitted by mobile users are placed in an Input Queue by the coordinator site. These transactions are then transferred to the Active Queue during execution. Once the transaction has been completed, it is placed in the Suspend Queue while the two-phase commit (2PC) protocol is executed [GR93]. Upon completion of the commit protocol, the transaction and its outcome are placed in the Output Queue. The user may disconnect at any time during the execution of the transaction. Upon re-connection, the user may query the status of the transaction. The outcome of the transaction and any results produced are kept in the Output Queue which could be delivered to the user.

The use of queues allows the MDSTPM model to explicitly handle disconnection. As the execution of a transaction is coordinated by the initial site, transactions cannot migrate with

the user. Instead, all communications with respect to a global transaction need to be forwarded to the coordinator site. As all operations of a transaction need to be submitted before execution may begin, LLTs that involve human interaction cannot be supported. This publication does not discuss enforcing the isolation property. Although the two-phase commit protocol is to be used, the authors do not discuss the implementation details of this commit protocol. Note that the two-phase commit protocol (2PC) provides only strict atomicity. This approach does not violate any autonomy requirements of the local DBMSs. However, as the implementation details of the 2PC are unknown, no definitive conclusion can be drawn.

2.4.6 The Kangaroo Model

The model presented in [DHB97] is based on the open nested model and is the first model to capture the movement behavior of the mobile user. A global transaction (referred to as Kangaroo transactions) consists of a set of Joey transactions, each consisting of all operations executed within the boundaries of one MSS. Each Joey transaction consists of one or more sub-transactions. As each Joey transaction contains all sub-transactions that are submitted from some MSS, the set of Joey transactions capture the migration of the global transaction. As a Joey consists of sub-transactions, this technique does not address arbitrary migration that may occur in the middle of a sub-transaction. A Joey transaction may be committed independently. Kangaroo transactions execute in two different modes: Compensating mode and Split mode. Under the Compensating mode, the failure of any Joey transaction of a Kangaroo transaction causes all its committed Joeys to be compensated and all its other active Joeys to be aborted. Under the Split mode, all committed Joeys will not be compensated and the decision to commit or abort any active Joeys is left up to the component DBMSs. These modes provide a full spectrum of atomicity. However, under the Split mode, component DBMSs may be left in an inconsistent state. Neither mode enforces the isolation property.

2.4.7 A Pre-Commit Model

The transaction model presented in [MB98] addresses transaction management in the mobile database environment in general. It does not specifically address the MMDB environment. This technique introduces a pre-read, pre-write, and a pre-commit operation to address the issues of mobile computing. Transactions of mobile users are initiated by the MH read or pre-read data values, manipulate the data that has been read and then pre-write modified values at the MH. Once all pre-write values have been declared, the transactions pre-commit at which point, all pre-write values are transmitted to the MSS. The MSS will then complete the transactions, i.e., write all values to the database and commit the transactions. A pre-write does not update the state of the physical data object but only declares its modified value. Once a transaction pre-commits, its pre-write values are written to a pre-write buffer maintained in the MSS and are made visible to other concurrent transactions executing at that MH and the respective MSS. A transactions read will return a pre-read value if the latest value available has not been written to the database as yet; otherwise, the value residing in the database (read value) will be returned. All pre-committed transactions are guaranteed to commit by the MSS.

This transaction model does not fully support disconnection as it does not address disconnection that represents catastrophic failures. It addresses the concurrency limitation caused by the extended duration of mobile transactions by maintaining a pre-write buffer and making the pre-write values visible upon pre-commit. However, the pre-write values are visible only to those transactions that are executing in that MH or MSS. Note that this limits concurrency at the LDBS level. As this technique is not specifically designed for the MMDB environment, it does not address the autonomy requirement. In fact, this technique violates EI as transactions are pre-committed by the MSS which guarantees that the pre-committed transaction will not be aborted. In the MMDB environment, this cannot be achieved without the

unilateral cooperation of the LDBSs. This technique enforces the strict atomicity and strict isolation. It does not provide the functionality to enforce a range of correctness criterion. As pre-committed transactions do not abort, no undo recovery or compensating transactions need to be performed.

2.5 Summary of Review

As shown in Table 2-1, none of the reviewed techniques enforces the isolation property without violating the autonomy of the local databases. In fact, four of the seven techniques reviewed do not enforce the isolation property at all. As a result, conflicts have no effect on the outcome of transactions and, therefore, lengthy executions do not incur any ill-effects. However, as the isolation property is not enforced, the consistency of the databases is compromised. In addition, all techniques do not address disconnections that represent catastrophic failures. It is assumed that a disconnection will always be followed by a subsequent re-connection. Moreover, performance analysis has not been conducted in any of these studies.

Technique	Disctn	Migrtn	Autonomy Violated	LLT Support	Atomicity Level	Isolation Level
Agent-Based Access [PB95-2]	Partial	Partial	No	Yes	VAR	None
The Cluster Model [PB95]	Partial	Partial	EI/PI	Yes	VAR	VAR
Semantic based TP [WC94]	Partial	Partial	EI/PI	Yes	VAR	STR
TP in Mobile Env [Chry93]	Partial	Full	EI	Yes	VAR	None
MDSTMP [YZ94]	Partial	Full	No	No	STR	None
Kangaroo Model [DHB97]	Partial	Partial	No	Yes	VAR	None
Pre-commit model [MB98]	Partial	Full	Yes	Yes	STR	STR

Table 2-1 : Summary of Mobile Database Transaction Models

Chapter 3

TRANSACTION MANAGEMENT IN THE MMDB ENVIRONMENT

In this chapter, the Pre-Serialization Transaction Management (PS) technique will be introduced. This technique fully addresses disconnection and migration, minimizes any prejudices against LLTs, provides the full range of correctness criterion with respect to the atomicity and isolation properties, and conforms to all multidatabase design restrictions. The Partial Global Serialization Graph (PGSG) algorithm which is used to verify the atomicity and isolation properties will be presented in Section 3.1.4.1. This algorithm is a graph-based algorithm that verifies isolation by analyzing serializability graphs of only a subset of the nodes in the system. In order to ensure that all isolation violations are detected, the algorithm propagates serializability information during the commit of global transactions.

3.1 Overview

The Global Transaction Manager (GTM) of the PS technique is divided into two layers: the Global Coordinator (GC) layer manages the overall execution of global transactions and disconnection and migration of mobile users, and the Site Manager (SM) layer supervises the execution of site-transactions. Global transactions are initiated at the GC layer. The GC layer will submit the site-transactions to the SM layer. The SM layer submits the site-transactions to the respective LDBS, and forwards the outcome of the site-transactions to the GC layer. Global transactions are based on the multi-level transaction model. Site-transactions are categorized as either vital or non-vital [Chry93]. All vital site-transactions must succeed for the global transaction to succeed. However, the failure of a non-vital site-transaction does not cause the global transaction to fail. The interval in which all vital site-transactions are executed

is referred to as the vital phase of the transaction. For simplicity, all LLTs will be considered to be mobile global transactions, and all non-LLTs will be considered as static global transactions.

Two new states - Disconnected and Suspended - are introduced to address disconnection and uncertainty about reconnection. Upon disconnection, global transactions are placed in the Disconnected state by the GC layer. Whenever a catastrophic failure is deemed to have occurred, global transactions associated with that connection are placed in the Suspended state. Suspended global transactions are not aborted until they interfere with the execution of other global transactions, thus minimizing erroneous termination.

The Partial Global Serialization Graph (PGSG) algorithm is used to verify the A/I properties of a global transaction. This algorithm is based on the optimistic approach and enforces the range of correctness criterion. If the A/I properties have not been violated the algorithm establishes the transaction's serialization order in the global serialization scheme. Note that the algorithm does not maintain a global serialization graph. Each site maintains a Site Serialization Graph (SSG) that contains partial global serialization information. The global serialization scheme can be obtained through the union of all SSGs - a very costly operation. However, the PGSG algorithm does not construct the entire global serialization scheme in order to verify isolation; it only constructs a partial global serialization scheme - hence its name.

A static global transaction initiates the PGSG algorithm at the end of its execution. If the A/I properties can be verified, the transaction's serialization order is registered in the global serialization scheme - henceforth referred to as being toggled - and the transaction is committed; otherwise, it is aborted. However, a mobile global transaction initiates the PGSG algorithm at the end of its vital phase. If the A/I properties can be verified, the transaction is toggled and execution continues; otherwise it is aborted. A toggled mobile global transaction is allowed to initiate only non-vital site transactions. At the end of execution of a toggled mobile global transaction, the transaction executes the PGSG algorithm a second time to verify whether any of

the non-vital site-transactions executed after being toggled violate the established serialization order. Any (non-vital) site-transaction that violates this order is aborted. As only non-vital site-transactions are initiated after being toggled, the global transaction is guaranteed to commit. The toggle operation minimizes the ill-effects of extended executions of mobile global transactions as they are allowed to establish their serialization order prior to completing their execution.

3.2 The Model

Global transactions are based on the Multi-Level transaction model. This model is ideally suited for the MMDB environment for three reasons:

1. Atomicity can be enforced without the cooperation of the LDBSs as sub-transactions are compensatable;
2. It allows sub-transactions to commit early, independent of the global transaction;
3. it provides the flexibility to accommodate a wide range of A/I criteria.

In the proposed model, all operations of a global transaction accessing the same LDBS constitute a site-transaction (analogous to sub-transaction) that is compensatable and will be executed as a single transaction with respect to that site. This will ensure that the global transaction does not execute more than one ACID transaction at any LDBS. In addition, all site-transactions will be categorized as either vital or non-vital [Chry93]. Vital site-transactions are site-transactions that must succeed in order for the global transaction to succeed. The abort of non-vital site-transactions does not force the global transaction to be aborted.

3.3 The Global Transaction Manager

The Global Transaction Manager (GTM) consists of two layers: a Global Coordinator (GC) layer and a Site Manager (SM) layer (Figure 3-1). The GC layer consists of a set of Global

Transaction Coordinators (GTCs) such that there exists a GTC at each MSS and any other static node to which global users may connect. All external users connect to the MMDBS via some GTC. The GTC is responsible for the overall execution of all global transactions of users currently connected to it. The GTC will submit site-transactions to the respective sites, handle disconnection and migration, log responses that cannot be delivered to the disconnected user, enforce the A/I, etc. The SM layer consists of a set of Site Transaction Managers (STMs) such that there exists an STM at each participating LDBS. The STMs receive site-transactions from the GTCs, submit the site-transactions to the respective LDBSs and oversee their execution.

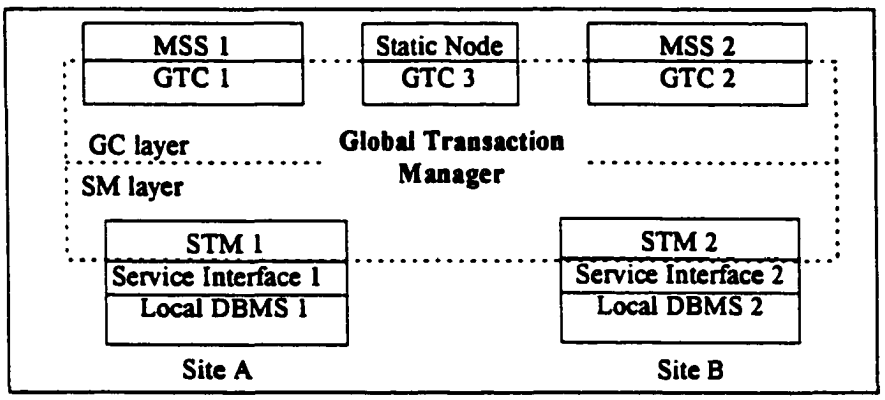


Figure 3-1: Global Transaction Manager

Each global transaction can be in one of five states: 1) Active - the user is connected and execution continues; 2) Disconnected - the user is disconnected, but the disconnection was predicted and re-connection is expected; 3) Suspended - the user is disconnected and is deemed to have encountered a catastrophic failure; 4) Committed - the transaction committed successfully; and 5) Aborted - the transaction is aborted. Note that the states Disconnected and Suspended do not apply to global transactions of static users.

When a global transaction is initiated by a user, the respective GTC creates a global data structure to keep track of the information required to supervise its overall execution. The Global Data Structure is given in Table 3-1.

GTID	global transaction identifier
GT_Type	Mobile (LLT) or static (non-LLTs)
GT_Status	current state of global transaction
Isolation_Verified	specifies whether isolation has been verified
Site_List	respective site of each site-transaction
STID_List	respective STID of each site-transaction
Critical_List	specifies vital/non-vital for each site-transaction
STID_Status_List	respective status of each site-transaction
Response_List	list of undelivered responses, if any

Table 3-1: Global Data Structure

When a user migrates to a new cell, the user will supply the current MSS with the identity of the previous MSS. The GTC at the current MSS will obtain the associated Global Data Structure from the previous GTC and assume the responsibility of the overall execution.

The STM at each site supervises the execution of site-transactions submitted to that site. Each LDBS defines the set of operations accepted by that LDBS. Each site-transaction can be in one of four states: 1) Active - the site-transaction is active; 2) Completed - the site-transaction has committed at the local database but the global transaction has not committed; 3) Aborted - the site-transaction is aborted; or 4) Committed - the site-transaction and the respective global transaction have committed. Each STM will maintain a Site Table containing information on all site-transactions submitted to it. For each site-transaction, the following information will be collected:

GTID	respective GTID
STID	assigned STID
MSS_ID	current MSS to which user is connected
STID_Status	current state of site-transaction
Compensating_Transaction	compensating transaction of site-transaction

Table 3-2: Site Table Structure

The GTC submits all site-transactions and their compensating (site) transactions to the respective STMs. Upon completion of each site-transaction, the STM will submit a commit operation to the LDBS and consequently update the STID_Status to reflect the outcome of the local commit operation, i.e., marked Completed or Aborted. The outcome will then be conveyed to the GTC to be recorded in the Global Data Structure. Site-transactions are committed locally independent of the future outcome of the global transaction in order to ensure that local resources are released in a timely manner.

Whenever a user disconnects, the respective GT_Status is marked as Disconnected. The execution of Disconnected transactions are not halted. All responses received after disconnection are placed in the Response_List. Upon reconnection, the GT_Status of Disconnected transactions will be set to Active, all responses in the Response_List are delivered to the user, and execution proceeds. At any time during a period of disconnection if the MMDBS determines that a catastrophic failure has occurred, the respective GT_Status is marked as Suspended and the execution is halted, i.e., no new site-transactions are initiated. Suspended global transactions are not aborted until they obstruct the execution of other global transactions. This will minimize the number of unnecessary aborts caused by erroneous decisions.

To verify the A/I properties of a global transaction, the respective GTC will execute the Partial Global Serialization Graph (PGSG) algorithm. This algorithm verifies the A/I properties with respect to all successful site-transactions of a global transaction. A static global transactions initiates the PGSG algorithm at the end of its execution. If the A/I properties have not been violated, the transaction's serialization order is registered in the global serialization scheme (i.e., toggled) and it is committed; otherwise it is aborted. In the case of a mobile global transaction, the PGSG algorithm is initiated at the end of its vital phase. If either if the A/I

properties have been violated, the mobile global transaction is aborted; otherwise the mobile global transaction is toggled, the Isolation_Verified field of the respective Global Structure is set to true, and execution is allowed to continue. After being toggled, a mobile global transaction may initiate only non-vital site-transactions. As a toggled mobile global transaction establishes its serialization order in the global serialization scheme, it is guaranteed to commit. At the end of its execution, each toggled mobile global transaction initiates the PGSG algorithm the second time to verify that the (non-vital) site-transactions executed after being toggled do not violate the already established serialization order. If any site-transaction violates this order, it is aborted. However, as it is non-vital, its abort does not cause the global transaction to be aborted. A toggled mobile global transaction is aborted only if it obstructs the execution of another global transaction while it is in the Suspended state. As mobile global transactions are allowed to establish their serialization order prior to completing their execution, the prejudicial treatment of mobile global transactions is minimized.

3.4 the Atomicity and Isolation Properties

In this Section, the PGSG algorithm used to enforce the A/I properties will be introduced. First, an overview of the algorithm is presented. Details of the algorithm will be provided in subsequent sub-sections.

As stated previously, it has been argued that strict atomicity cannot be implemented in the multidatabase environment without violating local autonomy. Without taking sides in that argument, the atomicity property of the PS technique will be based on condition 2 of semantic atomicity. That is, either all site-transactions are committed, or all site-transactions are aborted or compensated for. Condition 2 of semantic atomicity is chosen as it allows (compensatable) site-transactions to be committed even before the decision to commit the global transaction is reached.

The isolation property is based on serializability. In Chapter 2 it was stated that in order for a set of local and global transactions to be serializable in the multidatabase environment, all local serialization graphs must be acyclic and the global serialization graph which contains only global transactions must be acyclic. In Chapter 1 it was stated that all transactions executed by the local LDBSs will satisfy the ACID properties. This guarantees that all local serialization graphs will be acyclic. Therefore, the MMDBMS needs only to ensure that the global serialization graph that represents global transaction serialization order at the local sites is acyclic.

To capture the local serialization scheme, each STM maintains a Site Serialization Graph (SSG). The SSG is an ordered graph that reflects the execution order of site-transactions at that site. The combination of all SSGs represents the global serialization order of all global transactions. The nodes in the SSG represent global transactions and are categorized as either Accessed or Propagated. Accessed nodes represent transactions that execute vital site-transactions at that site. Propagated nodes are nodes that get copied to the SSG whenever the STM participates in the PGSG algorithm. The edges in the SSG represent the serialization order of the global transactions.

The PGSG algorithm will first verify semantic atomicity of the transaction to be toggled or committed, say T_i. That is, the PGSG algorithm verifies that all vital site-transactions of T_i have successfully committed at the respective LDBSs, i.e., marked Completed. If all vital site-transactions are marked Completed, the atomicity property is satisfied; otherwise, all Completed site-transactions of T_i (vital and non-vital) are compensated and T_i is aborted.

The fact that the abort of a non-vital site-transaction does not cause the global transaction to be aborted is actually a violation of condition 2 of semantic atomicity. However, the need for this can be argued as follows. In Chapter 2, it was argued that the GTM needs to support the full range of A/I properties. If all site-transactions are classified as vital, then all

site-transactions must succeed in order for the global transaction to succeed and condition 2 is strictly adhered to. If on the other hand all site-transactions are classified as non-vital, then unrestricted access is supported. By allowing a global transaction to consist of any combination of vital and non-vital site-transactions, the PS technique supports the full range of the A/I properties.

To verify serializability of T_i, the PGSG algorithm constructs the Partial Global Serialization (PGS) graph. The nodes in the PSG graph represent a subset of global transactions and the edges represent their serialization order. The PGS graph is constructed by combining Predecessor graphs obtained from the SSGs at all STMs at which T_i executed site-transactions successfully - henceforth referred to as Primary sites. Each Predecessor graph contains T_i, all nodes that precede T_i in that SSG, and additional serialization information obtained from other STMs with respect to propagated nodes in that SSG that are Active (henceforth referred to as Candidate nodes). These sites are referred to as Secondary sites.

After constructing the PGS graph, the algorithm verifies whether T_i violates the established serialization order of all committed and toggled transactions. Violations are represented as cycles that consist of T_i and other toggled or committed transactions. If cycles are detected, the algorithm will attempt to break these cycles by aborting Suspended (toggled) mobile global transactions as they are obstructing the execution of another global transaction. Note that all transactions marked as Suspended have not been committed and therefore can be aborted. If one or more cycles cannot be broken by aborting Suspended transactions, all Completed site-transactions of T_i are compensated and the global transaction is aborted. If there are no cycles or all cycles can be resolved, the Isolation_Verified field is set to True, the global transaction is committed or toggled, and the PGS graph is sent to all participating STMs , i.e., all Primary and Secondary sites. At each site, serialization information contained in the PSG graph is copied to its SSG - henceforth referred to as propagation.

Propagation is a fundamental part of the PGSG algorithm and ensures that all cycles are detected even though the global serialization scheme is only partially represented in each PSG graph. Whenever a global transaction T_i establishes its serialization order in the global serialization scheme, propagation is designed to copy all active nodes T_j that appear before T_i in the PGS graph (i.e., conflict with and precede T_i in the serialization order) to other participating SSGs. The motivation behind propagation is the following. Some active transaction T_j that precedes T_i in the serialization order when T_i was toggled may initiate other conflicting site-transactions in the future such that it now appears after T_i in the global serialization scheme, thus causing a cycle. However, propagation will ensure that when the last transaction in the cycle attempts to establish its serialization order, the cycle will be in the PGS graph as:

1. Propagation copies preceding serialization information to all participating SSGs.
2. In a cycle, any node precedes all other nodes.

Therefore, the cycle is detected and the global transaction is aborted. Note that the key to the algorithm is to identify those nodes to which the "preceding" serialization information needs to be copied. The nodes to which the PGSG needs to propagate information are all Active nodes that precede T_i in the PGS graph.

3.4.1 The PGSG Algorithm

As the execution order of site-transactions within the local databases are transparent to all external processes, the STM cannot determine the local serialization order by any direct means. However, the execution order of site-transactions within the local LDBS may be obtained implicitly by forcing conflicts among the site-transactions by using a data item called a ticket [GRS91] maintained at each site. Each site-transaction is required to read the ticket at that LDBS, increment its value and write the new value back as part of its execution. The ticket value read by the site-transaction indicates its serialization order at that site [BMS92] with

respect to other site-transactions and will be used to construct the SSG. Note that, any site-transaction that violates the serialization order represented by its ticket will be aborted by the LDBS as all sites enforce the ACID properties on all local transactions.

The SSG at each site is a directed graph whose nodes represent the respective GTIDs of site-transactions and edges represent (forced) conflicts between the respective site-transactions executed at that site (i.e., ticket values). For example, there exists $T_1 \rightarrow T_2$ in some SSG if and only if global transactions T_1 and T_2 access at least one common site and the ticket obtained by the site-transaction of T_1 is less than the ticket obtained by the site-transaction of T_2. The information contained within each node is given is Table 3-3. Each node in the SSG is categorized as either an Accessed node or a Propagated node. An Accessed node represents a global transaction that executed a site-transaction at that site. A Propagated node represents a global transaction whose serialization order was copied to the SSG during the execution of the PGSG algorithm. Next, certain terms used in the algorithm are defined.

GTID	Respective global transaction ID
GT_Status	status of global transaction
Isolation_Veri	commit intent of global transaction
Node_Categor	Accessed or Propagated
Site_ID	If Access, then this Site_ID; Else the Propagated Site_ID

Table 3-3: SSG Node

Definition 1: We say that T_j is *reachable* from T_i in graph G if there is a path from T_i to T_j in G, i.e., $T_i \rightarrow ... \rightarrow T_j$.

Definition 2: Reachable(T_j) is a (directed) sub-graph of an SSG that contains node T_j and all nodes T_i such that T_j is reachable from T_i in the SSG.

Definition 3: a *Candidate* node is any Propagated node whose GT_Status is not Committed.

Definition 4: Let G_i and G_j be two graphs with node sets N_i and N_j and edge sets E_i and E_j respectively. The operation $G = Merge(G_i, G_j)$ results in a new graph G(N, E) such that $N = N_i \cup N_j$ and $E = E_i \cup E_j$ where $\cup$ is the union operator. (G does not contain duplicate edges).

Definition 5: The graph $Predecessor(T_j, S_m)$ is the sub-graph Reachable(T_j) of the SSG at site S_m Merged with all Predecessor(T_k, S_n) graphs where T_k is a Candidate node in Reachable(T_j) and S_n is the respective propagated site of T_k. Formally,

Predecessor(T_j, S_m) = { G = Reachable(T_j) | Merge (G, Predecessor(T_k, S_n)) $\forall$ Candidate nodes T_k in Reachable(T_j) where S_n is the Site_ID of T_k }

In the graph Predecessor(T_x, S_y), T_x is referred to as the requested root node.

Definition 6: The list $PList(T_j, S_m)$ is a list (maintained at site S_m) whose elements represent sites from which Predecessor graphs were obtained in order to construct Predecessor(T_j, S_m).

The PGSG algorithm consists of two modules: the GlobalCoordinator module constructs the PGS graph from the Predecessor graphs and verifies the A/I properties, and the RequestPredecessor module constructs the Predecessor graphs. The GlobalCoordinator module is executed by the GTC that supervised the execution of the global transactions at the time that the commit or toggle operation was initiated. T_i represents the global transaction to be committed or toggled and the Request argument specifies it is to be committed or toggled. First, this algorithm verifies that all vital site-transactions have been Completed. Next, the algorithm obtains the Predecessor graphs from all Primary sites at which T_i successfully executed site-transactions. Each Primary site executes the RequestPredecessor algorithm to construct the

Predecessor graph and submits it to the GTC. The PGSG algorithm will then verify serializability and either toggle, commit, or abort the global transaction. If the global transaction is to be committed or toggled, the PGS graph is sent to all participating sites so that the required serialization information is propagated.

Algorithm 3-1: GlobalCoordinator (T_j, Request)

```
/* Verifies the A/I properties */
/* first, verify atomicity */
If any critical site-transaction has been aborted
  Send ABORT (Tj) to all sites in Site_List  /* Abort all site-transactions */
Else
  /* next, verify isolation */
  for all site Sm in Site_List  where Ti is marked Completed, obtain Predecessor(Ti, Sm)
    by executing the Request Predecessor(Ti, Sm) algorithm
  Generate PGSG by Merging all Predecessor(Tj, Sm)
  Check for cycles w.r.t. Ti , Committed nodes and Togged nodes
  If cycles are detected
    If cycles can be broken by aborting Suspended global transactions or
     non-vital site-transactions of Tj
       Mark GT_Status of Suspended nodes as Aborted in PGSG
    Else  /* isolation violated */
       Send ABORT (Tj) to all sites in Site_List  /* Abort all site-transactions */
       Exit Algorithm
    End If
  End If
  /* A/I properties verified */
  Mark Isolation_Verified  in Global Structure and node Tj in PGS graph as True
  /* Propagate  success and serialization information */
  Send "SUCCESS" and PGS graph  to sites in Site_List where Ti is marked Completed
End If
End {PGSG Algorithm}
```

For Transaction T_i, The GlobalCoordinator module initiates RequestPredecessor(T_i, S_m) at all primary sites S_m. In turn, each site S_m will initiate RequestPredecessor(T_j, S_n) for all candidate nodes T_j (propagated from Site S_n) in Predecessor(T_i, S_m). If Predecessor(T_j, S_n)

contains any Candidate nodes T_k, then the Secondary site initiates RequestPredecessor(T_k, S_p) for all Candidate nodes T_k. All S_ps are also categorized as Secondary sites. Finally, Secondary sites S_n submit their graphs to the Primary sites which submit Predecessor(T_i, S_n) to the GlobalCoordinator module. Each site then awaits the outcome of the Commit of Toggle operation. If T_i is to be committed or toggled, each site (Primary and Secondary) will copy propagation information by merging Reachable(T_x) of the returned PGS graph with Reachable(T_x) of its SSG where T_x represents the requested root node for the Predecessor graph submitted by that site. For example, T_x represents global transaction T_i in all Primary sites.

Algorithm 3-2: Request Predecessor(T_j, S_m)

```
/* Construct Predecessor graph */
Construct Predecessor(Tj, Sm), PList(Tj, Sm)
Submit Predecessor(Tj, Sm) to requester
Wait for Reply from requester
If Reply is ABORT (Tj)  /* site-transaction is to be aborted */
  If Ti is Accessed node in SSG  /* this is a Primary site */
    Abort Tj if Active or compensate Tj if Completed
  End If
  Send ABORT (Tj) to all sites in PList(Tj, Sm)  /* inform all Secondary sites */
Else  /* global transaction is to be toggled */
  If Ti is Accessed node in SSG  /* this is a Primary site */
    Mark Isolation_Verified as True
  End If
  /* Primary and Secondary copy serialization information */
  SSG = Merge(SSG, Reachable(Tj) of received PGS graph)
  Update status of Candidate nodes in Reachable(Tj)
  Send 'SUCCESS' and PGS graph to all sites in PList(Tj, Sm)
End If
End { Request Predecessor}
```

3.4.2 A Sample Execution of the PGSG Algorithm

In this example, the MMDBS consists of 3 sites labeled S_1 through S_3. There are 3 active global transactions labeled T_1 through T_3 in the system. For simplicity, we assume that each transaction accesses two sites, all site-transactions at each site conflict with each other, and that all transactions have completed their execution but have not yet committed. The algorithm is illustrated in Table 3-4. The initial SSG at each site is given in row one. For example, at site S_3, $T_3 \rightarrow T_1$ indicates that T_3 and T_1 conflict and that T_3 is serialized before T_1. Initially, all nodes are Accessed nodes. Rows two through four reflects the SSGs after the completion of the PGSG algorithm of the transaction given in column one. If a site does not participate in the PGSG algorithm, it will not have an entry in the corresponding row as the SSG does not change. The Site_ID of Propagated nodes is given in brackets below the respective node. The last column reflects the PGS graph that is constructed at each stage. A [C] below the respective node in the PGS graph states that the node is to be committed and a [A] states that the node is to be aborted.

In this example T_1 executes the PGSG algorithm first, T_3 second, and T_2 third. For the commit of T_1, S_1 and S_2 participate as Primary sites. The Predecessor graph submitted by S_1 is $T_3 \rightarrow T_1$, and the Predecessor graph submitted by S_2 is T_1. As there are no cycles in the PSG graph that is constructed by combining the two Predecessor graphs, T_1 commits successfully. After the commit, the PGS graph is sent to all participating sites, i.e., S_1 and S_2. At each site, the nodes in the PSG graph that are not in Predecessor(T_1, S_x) are copied (propagated) to the SSG. Next, for the commit of T_3, S_1 and S_3 participate as Primary sites. Once again, as there are no cycles in the PSG graph, T_3 is committed and node T_2 is propagated to S_1. Next, for the commit of T_2, S_2 and S_3 participate as Primary sites. As Predecessor(T_3, S_2) S_2 has an Active propagated node in its SSG with Site_ID S_1 (i.e., T_3 which when propagated to S_2 was Active and therefore, still deemed to be active), S_1 will participate as a Secondary site. Here, the PSG graph will

contain a cycle involving T_2 and therefore, T_2 will be aborted and removed form the SSGs at all sites as part of the Propagation process. Note that the cycle will not be detected if this example is carried out without Propagation.

	S_1	S_2	S_3	PGS graph
Initial SSGs	$T_3 \rightarrow T_1$	$T_1 \rightarrow T_2$	$T_2 \rightarrow T_3$	
Commit of T_1	$T_3 \rightarrow T_1$	$T_3 \rightarrow T_1 \rightarrow T_2$ [S_1]		$T_3 \rightarrow T_1$ [C]
Commit of T_3	$T_2 \rightarrow T_3 \rightarrow T_1$ [S_3]		$T_2 \rightarrow T_3$	$T_2 \rightarrow T_3$ [C]
Commit of T_2	$T_3 \rightarrow T_1$	$T_3 \rightarrow T_1$ [S_1]	T_3	$T_2 \rightarrow T_3 \rightarrow T_1 \rightarrow T_2$ [A]

Table 3-4: Sample execution of PGSG algorithm

3.4.3 Proof of Correctness

Lemma 1: Let $T_i \rightarrow T_j$ be in the SSG at some site S_j. Then, T_i began its execution at S_j prior to the completion of T_j's execution at S_j and therefore, prior to the (global) commit of S_j.

Proof: In order for $T_i \rightarrow T_j$ to exist, T_i must have obtained a ticket that is less than the ticket obtained by T_j. Therefore, T_i began its execution at S_j prior to T_j completing its execution at S_j.

Theorem 1: Let $T = \{T_1, T_2, ..., T_n\}$ be a set of transactions that cause a cycle. Assume that T_2 through T_n commit successfully and that T_1 is the last transaction in T to attempt to commit. Then T_1 will be a Candidate node in some Predecessor($T_{1,}$, S_x) used to construct the PGS graph. Thus, the cycle will be detected.

Proof: For simplicity, let us assume that each transaction executes at exactly two sites such that the cycle

$C \equiv T_1 \rightarrow T_2 \rightarrow ... \rightarrow T_n \rightarrow T_1$ is produced.

By Theorem 1, for all T_q that have completed their execution, there exists $T_p \rightarrow T_q$ for some T_p in T in the SSG at some site S_q at which T_q executed. When T_q executes the PGSG algorithm, T_p is in Predecessor(T_q, S_q) used to construct the PGS graph. Now, either T_p is committed, or not committed.

If T_p is not committed then T_p will be added as a Candidate node to all the SSGs at which T_q executed.

If T_p is committed, then, by Theorem 1, there exists a T_m in S such that $T_m \rightarrow T_p$ at some site S_m at which T_p executed. Once again, either T_m was committed or not committed at the time of T_p's commit. If T_m was not committed, then T_m was propagated to S_m at the time of T_p's commit and, as a result, will be in Predecessor(T_q, S_q) at the time of T_q's commit and will be added as a Candidate node to all SSGs at which T_q executed. If T_m was committed, we may repeat this argument. As the conflicts are cyclic, Predecessor(T_q, S_q) used to construct the PGSG when T_q attempts to commit will always contain a non-committed node from T which will then be added as a Candidate node to all SSGs at which T_q executed.

Now let T_1 attempt to commit at site S_1 and Sn where $T_1 \rightarrow T_2$ and $T_n \rightarrow T_1$ exist, respectively. Then, as T_n is committed, the SSG at S_n will contain a Candidate node - say T_x with respective site S_x - in its Predecessor(T_1, S_n). If T_x committed after its propagation to site S_n, then the SSG at S_x would, in turn, contain a Candidate node. Finally, as the only node in the cycle that is currently active is T_1, the Predecessor(T_1, S_n) constructed at site S_n will contain T_1 as an Accessed node as well as a Candidate node. Therefore, Predecessor(T_1, S_n) will contain the entire cycle. Thus, the PGS graph will contain the cycle. As all nodes except T_1 are committed, the cycle will be detected.

3.4.4 Concurrent Executions of the PGSG algorithm

In the preceding sections it was assumed that the PGSG algorithm always executed in isolation, that is, each instance of the GlobalCoordinator algorithm (and its respective RequestPredecessor algorithms) executed without interference from other instances of the algorithm. This section removes this assumption, studies its effects, and extends the algorithm to prevent the loss of information during propagation that may result from concurrent executions.

In the MMDBS environment, it is possible that multiple GTCs may execute the PGSG algorithm on behalf of different global transactions at the same time. The concurrent execution of the PGSG algorithm may cause conflict information being propagated to be lost or ignored resulting in cycles going undetected. This will be illustrated using the sample execution presented in Section 3.1.4.2 and altering its execution as follows: Let us assume that global transaction T_1 has committed and that T_2 and T_3 have completed their execution but have not committed. See Table 3-5 for the SSGs at each site after T_1 has committed.

	S_1	S_2	S_3	PGS graph
Initial SSGs	$T_3 \rightarrow T_1$	$T_1 \rightarrow T_2$	$T_2 \rightarrow T_3$	
After Commit of T_1	$T_3 \rightarrow T_1$	$T_3 \rightarrow T_1 \rightarrow T_2$ [S_1]	$T_2 \rightarrow T_3$	$T_3 \rightarrow T_1$

Table 3-5: SSGs after T_1 has committed

Next, let us assume that T_2 and T_3 initiate the PGSG algorithm concurrently. Table 3-6 illustrates the execution of the algorithm. Row 3 and Row 4 contain the PGS graphs for T_3 and T_4 respectively. A "*" in a cell indicates that that site participates in the commit of the respective transaction. Assuming that propagation from the commit of one transaction does not

overwrite the information propagated from the commit of the other transaction, row 5 contains the SSGs after all transactions have committed.

Note that, neither PGS graph contains the cycle as the PGS for T_2 does not contain the information that would have been propagated had T_3 committed before T_2 and vise versa. Therefore, both T_2 and T_3 will be allowed to commit.

	S_1	S_2	S_3	PGS graph
After Commit of T_1	$T_3 \rightarrow T_1$	$T_3 \rightarrow T_1 \rightarrow T_2$ [S_1]	$T_2 \rightarrow T_3$	
Commit of T_3	*		*	$T_2 \rightarrow T_3$ [C]
Commit of T_2	*	*	*	$T_3 \rightarrow T_1 \rightarrow T_2$ [C]
	$T_2 \rightarrow T_3 \rightarrow T_1$ [S_3]	$T_3 \rightarrow T_1 \rightarrow T_2$ [S_1]	$T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_3$ [S_1] [S_2]	

Table 3-6: Concurrent commit of T_2 and T_3

To address this issue, the algorithm needs to ensure that global transactions whose PGS graphs contain at least one common node (i.e., the same node from the same SSG) do not execute the PGSG algorithm at the same time. Global transactions whose PGS graphs are disjoint may execute the PGSG algorithm concurrently as established by the next theorem and the following discussion.

Theorem 2: If the PGS graphs of global transactions T_i and T_j are disjoint (i.e., do not ~~do not~~ contain any common node), then the node set S_i containing all nodes in Reachable(T_i') where T_i' is modified by the Propagation phase of T_i and the node set S_j containing all nodes in Reachable(T_j') where T_j' is modified by the Propagation phase of T_j are also disjoint.

Proof: Let us assume the contrary. Assume that S_i and S_j contain a common node representing global transaction T_x. As propagation copies only those nodes contained in the PGS (and as it copies information only to nodes contained in the PGS), T_x must exist in the PGS of both T_i and T_j. Clearly, this is a contradiction and therefore, S_i and S_j must be disjoint.

This theorem states that if the PSG graphs of any two global transactions are disjoint, then the nodes modified during the propagation phase are also disjoint. Therefore, if the PGS graphs of two or more transactions that execute the PGSG algorithm concurrently are disjoint, then the SSG graphs after a concurrent execution would not be any different had the PGSG algorithms executed in some serial order.

To ensure that global transactions whose PGS graphs contain at least one common node do not execute the PGSG concurrently, a simple lock mechanism is employed. Each node in the SSG is associated with an exclusive lock. Each primary site S_m that executes Request Predecessor(T_j, S_m) needs to obtain a lock on T_j and all nodes in all Predecessor graphs used to construct Predecessor(T_j, S_m). This simple locking mechanism will ensure that transactions whose PGS graphs are not disjoint will not be allowed to execute the PGSG algorithm concurrently as stated in the next theorem.

Theorem 3: If global transaction T_i obtains all locks necessary to construct its PGS graph G_i, then no other transaction T_j whose PGS graph G_j has at least one common node with G_i can obtain all locks necessary to construct its PGS graph.

Proof: Let us assume the contrary. Assume that Global Coordinators GTC_i and GTC_j execute the PGSG algorithm concurrently for T_i and T_j respectively, and that G_i and G_j have at least one

common node N_k. This implies that both GTC_i and GTC_j have obtained an exclusive lock on N_k. This contradicts the definition of "exclusive" lock and therefore cannot occur.

The updated GlobalCoordinator algorithm and RequestPredecessor algorithm are given below. The statements that have been added are in bold text.

Algorithm 3-3: GlobalCoordinator (T_j, Request)

/* Verifies the A/I properties */
/* first, verify atomicity */
If any critical site-transaction has been aborted
  Send ABORT (T_j) to all sites in Site_List /* Abort all site-transactions */
Else
  /* next, verify isolation */
  Loop
    For all site S_m in Site_List where T_j is marked Completed, obtain Predecessor(T_j, S_m) by executing the Request Predecessor(T_j, S_m) algorithm
    If any site returns SHARE-VIOLATION send SHARE-VIOLATION to all sites
  While some site returns SHARE-VIOLATION
  Generate PGSG by Merging all Predecessor(T_j, S_m)
  Check for cycles w.r.t. T_j, Committed nodes and Togged nodes
  If cycles are detected
    If cycles can be broken by aborting Suspended global transactions or non-vital site-transactions of T_j
      Mark GT_Status of Suspended nodes as Aborted in PGSG
    Else /* isolation violated */
      Send Abort (T_j) to all sites in Site_List /* Abort all site-transactions */
      Exit Algorithm
    End If
  End If
  /* A/I properties verified */
  Mark Isolation_Verified in Global Structure and node T_j in PGS graph as True
  /* Propagate success and serialization information */
  Send "Success" and PGS graph to sites in Site_List where T_j is marked Completed
End If
End {PGSG Algorithm}

Algorithm 3-4: RequestPredecessor(T_j, S_m)

```
/* Construct Predecessor graph */
Obtain Exclusive locks on Reachable(Tj)
If any lock cannot be obtained
   Release all locks and return SHARE-VIOLATION
   Exit
Else
   Construct Predecessor(Tj, Sm), PList(Tj, Sm)
   If any site returns SHARE-VIOLATION
      Release all locks and return SHARE-VIOLATION
      Exit
   End If
   Submit Predecessor(Tj, Sm) to requester
   Wait for Reply from requester
  If reply is SHARE-VIOLATION
      Release all locks and send SHARE-VIOLATION to all sites in PList(Tj, Sm)
      Exit
  Else  If Reply is ABORT(Tj)  /* site-transaction is to be aborted */
      If Ti is Accessed node in SSG  /* this is a Primary site */
         Abort Tj if Active or compensate Tj if Completed
      End If
      Release all locks
      Send ABORT (Tj) to all sites in PList(Tj, Sm)  /* inform all Secondary sites */
   Else  /* global transaction is to be toggled */
      If Ti is Accessed node in SSG  /* this is a Primary site */
         Mark Isolation_Verified as True
      End If
      /* Propagate serialization information */
      SSG = Merge(SSG, Reachable(Tj) of received PGS graph)
      Update status of Candidate nodes in Reachable(Tj)
      Release all locks
      Send 'SUCCESS' and PGS graph to all sites in PList(Tj, Sm)
   End If
 End If
 End { Request Predecessor}
```

Note that if at any time a necessary lock cannot be obtained, the PGSG algorithm releases all locks that have been obtained and starts over. This will ensure that the algorithm does not cause any deadlocks in the system.

3.4.5 Restricting the Growth of the SSGs

The difference in philosophies in pessimistic vs. optimistic concurrency control approaches can be generalized as preventive vs. cure. Pessimistic concurrency control algorithms are based on the assumption that it is more cost-effective to prevent isolation property violations. On the other hand, optimistic concurrency control algorithms are based on the assumption that it is more cost-effective to detect isolation property violations and take the necessary corrective measures. The PGSG algorithm introduced in this research is based on the optimistic philosophy.

In general, optimistic approaches can be categorized as either forward-examination or backward-examination. Forward-examination algorithms verify the serializability of a transaction T_i by looking at the serialization order of T_i and all active transactions in the system. Forward-examination algorithms detect a potential violation and resolve the violation by aborting some transaction involved in this potential violation. For example, if active transactions T_i and T_j conflict at some site where T_j is serialized before T_i and T_i and T_j do not conflict at any other site, then it may seem that T_i can be committed as there is no serializability violation. Yet if T_i is committed, a serializability violation could occur in the future if T_j executes at some other site at which T_i executed and T_j is serialized after T_i. This violation will not be detected when T_j attempts to commit as the algorithm is based on forward-examination and T_i has already committed. Therefore, when T_i attempts to commit, if T_i conflict with another transaction T_j at some site where T_j is serialized before T_i, forward-examination

protocols will abort either T_i or T_j even though there is no violation at this point in order to prevent any serializability violation occurring in the future.

On the other hand, backward-examination algorithms verify the serializability of a transaction T_i by looking at the serialization order of T_i and all committed transactions in the system. The advantage of backward-examination algorithms is that a transaction is aborted only if a real violation exists (as opposed to a potential violation). However, the disadvantage of the backward-examination algorithms is that, as the number of committed transactions in the system increases, the overhead required to verify serializability increases.

The PGSG algorithm verifies serializability based on the backward-examination approach. Therefore, the algorithm needs to address the growth in overhead over time. This is achieved by "trimming" the SSGs during the execution of the PGSG algorithm, thereby limiting the set of committed transactions that need to be considered. The basis behind this trimming is presented in the next theorem:

Theorem 4: Let node n_i representing committed transaction T_i be a node in some SSG. n_i can be a node in a cycle in the PGS of some global transaction T_j only if there exists a Candidate node in Predecessor(n_i).

Proof: Let us assume the contrary. Assume that T_i is committed, there is no Candidate node in Predecessor(n_i), that n_i is a node in a cycle in the PSG of T_j, and that T_j is the last node in that cycle that attempts to commit. Then by Theorem 1, T_j will be a Candidate node in some Predecessor($T_{j,}$, S_x) used to construct the PGS graph. As n_i is a node in the cycle, n_i must be in the Predecessor($T_{j,}$, S_x) that contains the cycle. Therefore, T_j must be in Predecessor(n_i). Clearly, this is a contradiction. Therefore, if T_i is committed and there is no Candidate node in its Predecessor(T_i) graph, it cannot be a node in a cycle.

As stated in the previous theorem, any node in an SSG representing a committed global transaction that does not contain a Candidate node in its Predecessor graph cannot be a node in a cycle. Therefore, it (and all nodes in its Predecessor) can be removed from that SSG. The PGSG algorithm trims its graphs as follows: Whenever an STM S_x executes Request Predecessor(T_j, S_x), the Request Predecessor algorithm trims the Predecessor(T_j) by removing any committed node T_i in Predecessor(T_j) such that $T_i \neq T_j$ and Reachable(T_i) contains no Candidate Nodes. Thus, any node that, by Theorem 2, cannot be a node in any cycle is removed from the SSG.

The updated RequestPredecessor algorithm is given below. Once again, the statements that have been added are in bold text.

Algorithm 3-5: RequestPredecessor(T_j, S_m)

/* Construct Predecessor graph */
Obtain Exclusive lock on Reachable(Tj)
If any lock cannot be obtained
  Release all locks and return SHARE-VIOLATION
  Exit
Else
  Construct Predecessor(T_j, S_m), PList(T_j, S_m)
  If any site returns SHARE-VIOLATION
    Release all locks and return SHARE-VIOLATION
    Exit
  End If
  Submit Predecessor(T_j, S_m) to requester
  Wait for Reply from requester
  If reply is SHARE-VIOLATION
    Release all locks and send SHARE-VIOLATION to all sites in PList(T_j, S_m)
    Exit
  Else If Reply is ABORT(T_j) /* site-transaction is to be aborted */
    If T_i is Accessed node in SSG /* this is a Primary site */
      Abort T_j if Active or compensate T_j if Completed

```
    End If
    Release all locks
    Send ABORT (T_j) to all sites in PList(T_j, S_m)   /* inform all Secondary sites */
  Else   /* global transaction is to be toggled */
    If T_i is Accessed node in SSG   /* this is a Primary site */
      Mark Isolation_Verified as True
    End If
    /* Propagate serialization information */
    SSG = Merge(SSG, Reachable(T_j) of received PGS graph)
    Update status of Candidate nodes in Reachable(T_j)
    Remove all T_i in Reachable(T_j) such that T_i ≠ T_j, T_i is committed and
      Reachable(T_i) contains no Candidate Nodes
    Release all locks
    Send 'SUCCESS' and PGS graph to all sites in PList(T_j, S_m)
  End If
End If
End { Request Predecessor}
```

3.5 Summary and Conclusion

This chapter proposes a new transaction management technique called Pre-Serialization (PS) for the mobile multidatabase environment The global transaction model of the PS technique is based on the multi-level transaction model, which requires site-transactions to be compensatable. The multi-level transaction model allows site-transactions to be committed prior to the decision to commit their global transaction, releasing local resources in a timely manner. Site-transactions are categorized as either vital or non-vital. The vital phase of a global transaction contains the entire execution between the first and last vital site-transaction of that global transaction. This categorization gives the PS technique the flexibility to enforce the full range of atomicity and isolation correctness criteria.

This technique introduces two new states to address the disconnectivity of the mobile user. Whenever a disconnection occurs, all global transactions of that user are placed in the Disconnected state. If at any stage it is deemed that the disconnected user has encountered a

catastrophic failure, these transactions are placed in the Suspended state. As catastrophic failures can only be predicted, Suspended transactions are not aborted until they interfere with the execution of other transactions. This minimizes unnecessary aborts caused by erroneous predictions.

This chapter proposes a new algorithm called PGSG that enforces the atomicity and isolation properties of global transactions in the MMDB environment. This algorithm verifies serializability by constructing a partial global serialization graph. This graph does not contain the complete serialization scheme of the MMDBS. Instead, it contains all the serialization information with respect to the transaction whose isolation property is being verified. This algorithm ensures that all cycles will be detected even though the complete serialization scheme in not reflected in the PGS through propagation which is the dissemination of serializability information. In order to minimize mobile global transactions being penalized due to their prolonged execution, the PS technique allows mobile transactions to establish their serialization order in the global serialization scheme at the end of their vital phase.

Chapter 4

THE SEMANTIC PRE-SERIALIZATION TRANSACTION MANAGEMENT TECHNIQUE

The PS technique has two major limitations. First, mobile global transactions incur additional overhead as opposed to static global transactions as mobile global transactions need to execute the PGSG algorithm twice. Second, this technique provides limited concurrency as each site employs a single ticket to serialize all site-transactions that execute at that site. In this section a Semantic Pre-Serialization (Semantic-PS) transaction management technique is proposed. The Semantic-PS technique is a modified version of the PS technique that overcomes the noted limitations of the PS technique.

4.1 Overview

The Semantic-PS differs from the PS technique in two areas - both relate to the enforcement of the A/I properties. First, in order to address the additional execution overhead incurred by mobile global transactions, the Semantic-PS technique further relaxes the A/I properties. That is, the Semantic-PS technique enforces the A/I properties only on the set of vital site-transactions of a global transaction. Therefore, all global transactions (mobile and static) need to execute the PGSG algorithm only once. Mobile global transactions execute the PGSG algorithm at the end of their vital stage. Static global transactions execute the PGSG algorithm at the end of their execution. Mobile global transactions are allowed to initiate non-vital site-transactions after being toggled.

Second, the Semantic-PS technique employs a modified version of the ticket method to improve concurrency. In this version, each LDBS maintains a set of tickets and forces conflicts only between site-transactions that potentially conflict with each other. It does not force

conflicts between all site-transactions that execute at that LDBS, thereby increasing concurrency.

Next, a detailed description of the enforcement of the A/I properties is provided. The transaction model and the GTM architecture is identical to that of the PS techniques.

4.2 The Atomicity and Isolation Properties

The A/I properties of the Semantic-PS technique are enforced on the set of vital site-transactions only. This does not limit the scope of the technique. It still provides the full range of correctness criteria as well. That is, if all site-transactions of a global transaction are categorized as vital, then strict A/I is enforced. On the other hand if all site-transactions of a global transaction are categorized as non-vital, then the A/I properties will not be enforced. This technique differs from the PS technique as follows: The PS technique enforces the A/I properties on all site-transactions that are completed successfully, i.e., all vital site-transactions and all non-vital site-transactions that complete execution successfully. Thus, the entire global transaction is executed as a consistent unit of computing. The Semantic-PS technique enforces the A/I properties on the set of vital site-transactions only. Therefore, only the set of vital site-transactions is executed as a consistent unit of computing. Although the Semantic-PS technique reduces the execution overhead and increases concurrency, its application is limited. It can only be used in environments where non-vital site-transactions do not cause any inconsistencies or, where the inconsistencies caused by the non-vital site-transactions can be tolerated.

As in the PS technique, the atomicity property of the Semantic-PS technique is based on condition 2 of semantic atomicity, and the isolation property is based on (global) serializability of global transactions. In the Semantic-PS technique, each site maintains an SSG graph as well. However, in Semantic-PS, only the execution of vital site-transactions is recorded in each SSG. The execution of non-vital site transactions is not recorded in the SSG. The Semantic-PS

technique enforces the isolation property by executing the modified PGSG algorithm described below.

In Semantic-PS, all global transactions execute the PGSG algorithm only once. Mobile global transactions execute the PGSG algorithm at the end of their vital stage while static global transactions execute the PGSG algorithm at the end of their execution. The PGSG algorithm will first verify semantic atomicity of the transaction to be toggled - say T_i. That is, the PGSG algorithm verifies that all vital site-transactions of T_i have successfully committed at the LDBSs, i.e., marked Completed. If all vital site-transactions are marked Completed, the atomicity property is satisfied; else, all Completed site-transactions of T_i (vital and non-vital) are compensated and T_i is aborted.

Next, the PGSG algorithm will construct the Partial Global Serialization (PGS) graph to verify serializability of T_i. Note that only the execution of vital site-transactions are represented in the PSG graph. After constructing the PGS graph, the PGSG algorithm will look for serializability violation in the PSG graph. Violations are represented as cycles that consist of T_i and other toggled or committed transactions. If cycles are detected, the algorithm will attempt to break these cycles by aborting Suspended mobile global transactions as they are obstructing the execution of another global transaction. If the cycles cannot be broken, all Completed site-transactions are compensated and the global transaction is aborted. If there are no cycles or the cycles can be resolved, the Isolation_Verified field is set to True, the global transaction is toggled, and the PGS graph is sent to all participating sites so that serialization information is propagated. Toggled mobile transactions are committed at the end of their execution.

4.2.1 The PGSG Algorithm

In the Semantic-PS technique, the execution order of site-transactions within the local LDBSs is obtained by using an enhanced version of the ticket method used in the PS technique.

Along with the operations accepted by each LDBS, the service interface provides conflict information with respect to the exported operations. That is, each service interface groups the exported operations in to a set of groups G such that all operations in any group potentially conflict with other. For example, if operation o_1 and o_2 access some table t in the LDBS, then o_1 and o_2 could potentially access the same data item in t. Therefore, o_1 and o_2 need to appear in at least one group in G. Formally, the service interface specifies a set of operations $O = \{o_1, \ldots o_m\}$ accepted by that site and a set of groups $G = \{g_1, \ldots g_n\}$ such that for all g_i in G, $g_i = \{o^i_1, \ldots, o^i_p \mid o^i_x\ x = 1..p$ in O and o^i_x potentially conflicts with all operations in $g_i\}$. The LDBS maintains a set $T = \{t_1, \ldots, t_n\}$ of tickets such that ticket t_i is associated with group g_i in G. Note that an operation may belong to one or more groups and therefore, be associated with more than one ticket.

Each vital site-transaction is required to increment all tickets associated with each operation in that site-transaction. Note that non-vital site-transactions do not read any tickets. The ticket values read by the vital site-transaction indicates its serialization order with respect to all other (potentially conflicting) vital site-transactions that execute at that site and will be used to construct the SSG just as in the PS technique. However, as there are multiple tickets associated with each site, all site-transactions that execute at the same site do not conflict with each other. In affect, this multiple ticked method reduces the granularity of locking from the LDBS to data items within each LDBS.

The PGSG algorithm of the Semantic-PS technique is similar to that of the PS technique. The differences are:

1. In order to construct the PSG for a global transaction, only the sites at which the global transaction executed vital site-transactions need to submit Predecessor graphs

2. As non-vital site-transactions cannot cause isolation property violations, the PGSG algorithm of the Semantic-PS techniques does not attempt to resolve cycles in the PGS graph by aborting non-vital site-transaction
3. The GlobalCoordinator propagates the PSG only to those sites at which the global transaction executed vital site-transactions.

The PGSG algorithm of the Semantic-PS technique is given below. The statements that have been added or modified are in bold text.

Algorithm 4-1: GlobalCoordinator (T_j, Request)

/* Verifies the A/I properties */
/* first, verify atomicity */
If any critical site-transaction has been aborted
  Send ABORT (T_j) to all sites in Site_List /* Abort all site-transactions */
Else
  /* next, verify isolation */
  Loop
    for all site S_m in Site_List where T_j executed vital site-transactions, obtain
      Predecessor(T_j, S_m) by executing the Request Predecessor(T_j, S_m) algorithm
    If any site returns SHARE-VIOLATION send SHARE-VIOLATION to all sites
  While some site returns SHARE-VIOLATION
  Generate PGSG by Merging all Predecessor(T_j, S_m)
  Check for cycles w.r.t. T_j, Committed nodes and Togged nodes
  If cycles are detected
    If cycles can be broken by aborting Suspended global transactions
    /* Does not attempt to resolve cycles by aborting non-vital site-transactions */
      Mark GT_Status of Suspended nodes as Aborted in PGSG
    Else /* isolation violated */
      Send Abort (T_j) to all sites in Site_List /* Abort all site-transactions */
      Exit Algorithm
    End If
  End If
  /* A/I properties verified */
  Mark Isolation_Verified in Global Structure and node T_j in PGS graph as True
  /* Propagate success and serialization information */
  Send "SUCCESS" and PGS graph to sites in Site_List where T_j

executed vital site-transactions
End If
End {PGSG Algorithm}

The Request Predecessor code is the same as in the PS technique. However, unlike in the PS technique, the set of Primary sites in the Semantic-PS technique include only those sites at which the global transaction executed its vital site-transactions. (The set of Primary sites in the PS technique includes all sites at which the global transaction executed its site-transactions successfully.)

4.3 Summary and Conclusion

This chapter introduces the Semantic-PS transaction management technique. This technique proposes two changes to the PS technique in order to overcome its limitations. First, the Semantic-PS technique does not force conflicts between all site-transactions that execute at a given site (in order to obtain the local serialization order). Instead, it uses semantic information about the operations exposed by the local interfaces to increase concurrency. Second, the Semantic-PS technique enforces atomicity and isolation only on the set of vital site-transactions – a further relaxation of the A/I properties. As A/I is enforced only on the set of vital site-transactions, mobile global transactions do not have to execute the PGSG algorithm a second time as in the PS technique. Note that just as in the PS technique, the Semantic-PS technique enforces the full range of A/I correctness criteria.

Chapter 5

ANALYTICAL EVALUATION

This chapter provides an analytical evaluation of the three transaction management techniques: PS, PS-Semantic, and Kangaroo [DH95]. The Kangaroo technique is chosen as it supports unrestricted mobility and it does not violate local autonomy - vital requirements for the MMDB environment. Prior to conducting the evaluation, the following steps are carried out: First, a general MMDB transaction management evaluation model is developed; Second, the model parameters' values are determined; Third, the general MMDB transaction management evaluation model is modified to accurately reflect each individual technique. Once the tailored models have been developed, the performance of the three techniques is evaluated.

5.1 The General MMDB Transaction Management Evaluation Model

Analytical modeling allows one to abstract essential components of the system and to model these components without regard to surrounding detail that one determines as insignificant. Analytical models provide accurate estimations of performance of a system at a relatively low-cost. Once analytical models of some computational environment are presented, these models can easily be used to evaluate the performance of different algorithms. This reduces the time, effort, and cost of the initial evaluation.

As transaction management in the MMDB environment is relatively new, analytical models of this environment for evaluating the performance of transaction management algorithms have not been developed. In this section, an analytical model of the general MMDB transaction management environment will be developed. In this model the average service time (ST_{avg}) of a global transaction - the average time taken by the system to complete the execution of a global transaction - will be formulated with respect to the key components of the MMDB

environment that affect the execution of global transactions. These components are: communication time, execution time of site-transactions, the disconnection and relocation time, and the time taken to execute the commit algorithm. The model is presented next.

Let GT_{exe} be the average time taken to execute all site-transactions of a global transaction and GT_{commit} be the average time taken to commit a global transaction. Then, the average service time of a global transaction ST_{avg} is:

$$ST_{avg} = GT_{exe} + GT_{commit} \qquad (1)$$

Next, GT_{exe} and GT_{commit} need to be formulated. Let N_{st} be the number of site-transactions in a global transaction, EXE_{st} be the average time taken to execute a site-transaction and T_{think} be the average time interval between the completion of one site-transaction and the submission of the next site-transaction of the same global transaction. Then, in a static environment GT_{exe} is:

$$GT_{exe} = N_{st} * EXE_{st} + (Nst - 1) * T_{think}$$

That is, GT_{exe} is the number of site-transactions multiplied by the average execution time of a site-transaction plus the think time between site-transactions, if any. However, GT_{exe} in the mobile environment is affected by disconnection and migration and these need to be accounted for in the model. For simplicity, it is assumed that, upon re-connection, all outstanding messages will be exchanged before the next disconnection. This assumption simplifies the model as follows: Although multiple disconnections may occur during the execution of a site-transaction, at most only one disconnection will cause a delay to any EXE_{st} or T_{think}. Let DLY_{st} and DLY_{thk} be the delay caused by a disconnection to EXE_{st} and T_{think} ,

respectively, and P^{st}_{dcn} and P^{thk}_{dcn} be the probability of a disconnection occurring during EXE_{st} and T_{think}, respectively. Then GT_{exe} is given by:

$$GT_{exe} = N_{st} * (EXE_{st} + (P^{st}_{dcn} * DLY_{st})) + (Nst - 1) * (T_{think} + (P^{thk}_{dcn} * DLY_{thk})) \quad (2)$$

Here, the potential delay caused by disconnection has been factored into EXE_{st} and T_{think}. The potential delay is modeled as the probability of disconnection multiplied by the delay caused by disconnection.

Next, we calculate DLY_{st}, DLY_{thk}, P^{st}_{dcn}, and P^{thk}_{dcn}. First, it is necessary to calculate the average delay caused by a disconnection (DCN_{dly}). DCN_{dly} is the average time of a disconnection (DCN_{tm}) plus the time taken by the system to address reconnection and migration (if any). Let N_{dcn} be the average number of disconnection during the execution of a global transaction, N_{mgr}, such that $N_{mgr} <= N_{dcn}$ be the average migrations during the execution of a global transaction, and RL_{tm} be the average time to address relocation. Then, DCN_{dly} is:

$$DCN_{dly} = DCN_{tm} + (N_{mgr} / N_{dcn} * RL_{tm}) \quad (3)$$

DLY_{st} and DLY_{thk} are influenced by three factors (Figure 5-1): 1 - the total delay caused by disconnection (DCN_{dly}); 2 - the point within the current site-transaction at which the disconnection occurs (X); and 3 - the length of execution of the current site-transaction (EXEst). For example, in Figure 5-1 (A), DLY_{st} is 0 and in Figure 5-1 (B), DLY_{st} is $X + DCN_{dly} - EXE_{st}$.

Note that, a disconnection affects EXE_{st} only if $X + DCN_{dly} > EXE_{st}$. Therefore, DLY_{st} is calculated by taking the probability that $X + DCN_{dly} > EXE_{st}$ multiplied by the average delay to EXE_{st} given that $X + DCN_{dly} > EXE_{st}$. Let us consider the cases $DCN_{dly} <= EXE_{st}$ and $DCN_{dly} > EXE_{st}$ separately. When $DCN_{dly} <= EXE_{st}$, the probability that $X + DCN_{dly} > EXE_{st}$ is DCN_{dly} /

EXE_{st} and the average delay given that $X + DCN_{dly} > EXE_{st}$ is $DCN_{dly}/2$ - that is, the average of the minimum delay (i.e., 0) which occurs when $X + DCN_{dly} = EXE_{st}$ and the maximum delay (i.e., DCN_{dly}) which occurs when the disconnection occurs at the very end of the execution of the site-transaction, that is, $X = EXE_{st}$. Thus:

$$DLY_{st} = (DCN_{dly} / EXEst) * DCN_{dly} / 2 \qquad (4a)$$

When $DCN_{dly} > EXE_{st}$, the probability that $X + DCN_{dly} > EXE_{st}$ is 1 and the average delay given that $X + DCN_{dly} > EXE_{st}$ is $(DCN_{dly} - EXE_{st} + DCN_{dly})/2$ - that is, the average of the minimum delay which occurs when the disconnection is at the very beginning of the site-transaction, and the maximum delay which occurs when the disconnection occurs at the very end of the disconnection. Thus:

$$DLY_{st} = 1 * (DCN_{dly} - EXE_{st} + DCN_{dly})/ 2 \qquad (4b)$$

Similarly, to formulate DLY_{thk}, let us consider the case $DCN_{dly} <= T_{think}$ and the case $DCN_{dly} > T_{think}$ separately. When $DCN_{dly} <= T_{think}$ then DLY_{thk} is:

$$DLY_{thk} = (DCN_{dly} / T_{think}) * DCN_{dly} / 2 \qquad (5a)$$

When $DCN_{dly} > T_{think}$, DLY_{thk} is:

$$DLY_{thk} = 1 * (DCN_{dly} - T_{think} + DCN_{dly})/ 2 \qquad (5b)$$

Finally, the probability of disconnection during EXE_{st} and T_{think} is formulated. Assume that only one disconnection occurs during the execution of a global transaction. As disconnection is equally likely to occur at any time, the probability of that disconnection occurring during EXE_{st} (P^{st}_{dcn}) is simply:

$$P^{st}_{dcn} = EXE_{st} / (N_{st} * EXE_{st} + (N_{st} - 1) * T_{think})$$

Then, as N_{dcn} disconnection occur during the execution of EXEst, P^{st}_{dcn}) is:

$$P^{st}_{dcn} = N_{dcn} * EXE_{st} / (N_{st} * EXE_{st} + (N_{st} - 1) * T_{think}) \qquad (6a)$$

Similarly, the probability of a disconnection occurring during T_{think} (P^{thk}_{dcn}) is:

$$P^{st}_{thk} = N_{dcn} * T_{think} / (N_{st} * EXE_{st} + (N_{st} - 1) * T_{think}) \qquad (6b)$$

In (4a) and (4b) we have formulated DLY_{st}, in (5a) and (5b) we have formulated DLY_{thk}, and in (6a) and (6b) we have formulated P^{st}_{dcn}, and P^{thk}_{dcn}. GT_{exe} can now be obtained from choosing the appropriate formulas for DLY_{st} and DLY_{thk}. Given GT_{commit} RL_{tm} and GT_{exe} for any technique, ST_{avg} can be calculated from (1). Note that the values for RL_{tm}, EXE_{st}, and GT_{commit} will be modeled separately for each transaction management technique and then will be used to derive the service time for the specific transaction management technique as described in Section 5.3.

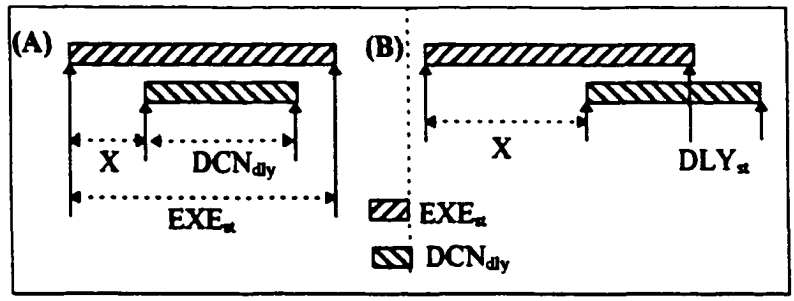


Figure 5-1: Relationship between DLY_{st} and X

5.2 Values of Model Parameters

In this section, the values of the model parameters for evaluating transaction management techniques in the MMDB environment are described. In order to simplify the evaluation, the following assumptions will be made about the environment:

- All sites in the MMDB environment are equally likely to be accessed
- All global transactions are mobile transactions and execute successfully at all sites
- All site-transactions are equivalent to those specified in TPC-C benchmark [TPC99]
- Site-transactions of a global transaction are executed consecutively. Each subsequent site-transaction is submitted to the MSS only after the results of the previous site-transaction is received and analyzed by the user (T_{think}).

The parameters used to construct the model and their default values are listed in the following table:

Parameter	Description	Default Values
ST_{avg}	avg. service time of a global transaction	Calculated
GT_{exe}	avg. time taken to execute a global transaction	Calculated
GT_{commit}	avg. time taken to commit a global transaction	Calculated
N_{st}	avg. number of site-transactions in a global transaction in the same global transaction	4
EXE_{st}	avg. time taken to execution a site-transaction (includes communication time between the user and MMDBS)	Calculated
T_{think}	avg. time between receiving the results of a site-transaction and submission of the next site-transaction	0
DCN_{tm}	avg. time between a disconnection and re-connection	0.1 second
DCN_{dly}	avg. processing delay caused by a disconnection (includes DCN_{tm} and processing time taken to address relocation etc.)	Calculated
RL_{tm}	avg. time to address relocation	Calculated
DLY_{st}	avg. delay added to EXE_{st} due to disconnection	Calculated
DLY_{thk}	avg. delay added to T_{think} due to disconnection	0
P^{st}_{dcn}	Probability of a disconnection occurring during EXE_{st}	Calculated
P^{thk}_{dcn}	Probability of a disconnection occurring during T_{think}	0
N_{dcn}	avg. Number of disconnection for a global transaction	$\lceil N_{st}/3 \rceil$
N_{mgr}	avg. Number of migrations for a global transaction	$\lceil N_{dcn}/3 \rceil$
T^{s}_{msg}	avg. time to transmit a message on the static (wired) network	0.0001 seconds
T^{w}_{msg}	avg. time to transmit a message over the wireless medium	0.07 Seconds
T^{s}_{pgn}	avg. time to transmit a Predecessor graph (or propagate a PGS graph) from site to site along the static network	0.001 seconds
EXE_{lcl}	avg. local execution time of a site-transaction	0.003 seconds
P_{cnf}	Probability of a site-transaction conflicting with another	0.05

Table 5.1: Model Parameters and Their Values

As mobile computing is a relatively new area, the values for many of the parameters used in the formulation of the model are not known. Here, educated guesses have been used to decide their default values. The rationale for choosing the stated values is given below.

The value of T_{think} has the same effect on all algorithms and therefore is set to 0 seconds. Similarly, the average delay due to disconnection that may be incurred between receiving the results of a site-transaction and the submission of the next site-transaction is set to 0 seconds. The average disconnection interval (DCN_{tm}) is arbitrarily set to 0.1 second. Picking an arbitrary value for DCN_{tm} does not favor any algorithm in any significant manner as it has a the same effect on all transaction management techniques. The local execution time of a site-transaction (EXE_{lcl}) is obtained from the TPC-C Benchmark [TPC99]. TPC-C is the Transaction Processing Performance Council's benchmark for Online Transaction Processing (OLTP) evaluation. EXElcl was obtained by calculating the average response time (obtained from the throughput from TPC-C) for five popular databases running on small to medium size servers (IBM DB2 on IBM AS400e, Informix OnLine 7.3 on Compaq ProLiant 5000, MS SQL Server 6.5 on Acer AcerAltos 19000Pro4, Oracle 7.3 on Sun UltraEnterprise 6000, and Sybase SQL Server 11.5 on Compaq ProLiant 6000). Message transmission time over the static (wired) network (T^{s}_{msg}) and wireless network (T^{w}_{msg}) have been calculated assuming that the average size of a message is 1 Kb and that the static network is a 10 Mbps Ethernet and the wireless communication medium is cellular telephony with a bandwidth of 14 Kbps [PS98]. The average time to transmit a Predecessor graph or to propagate a PGS graph from one site to another along the static network (10 Mbps Ethernet) is calculated assuming that the message is 10 Kb in size. The probability that a disconnection occurring during T_{think} (P^{thk}_{dcn}) is set to 0 as T_{think} is itself 0.

The default values for the average number of site-transactions in a global transaction (N_{st}), the average number of disconnection during the execution of a global transaction (N_{dcn}), the number of migrations during the execution of a global transaction (N_{mgr}), and the probability of a site-transaction conflicting with another site transaction (P_{cnf}) have been arbitrarily chosen. As these parameters have a significant effect on the service time of a global transaction, the analytical evaluation will study the performance for a range of values for each parameter.

5.3 Transaction Management Evaluation Models Tailored to Individual Techniques

The analytical model developed in Section 5.1 is a general model used to evaluate the performance of transaction management techniques in an MMDB environment. In this section, this model will be tailored to describe the PS, Semantic-PS and Kangaroo model. Specifically, EXEst, RL_{tm}, and GT_{commit} will be formulated separately for each technique.

5.3.1 The PS Technique

In this section, EXE_{st}, RL_{tm}, and GT_{commit}, for the PS technique will be formulated. The PS technique will incur two wireless messages to receive a site-transaction and submit its outcome to the user. Each site-transaction will require two additional wired messages to submit the site-transactions (and compensating transaction) and receive its outcome. Therefore, EXE_{st} is given by:

$$EXE_{st} = 2 * T^{s}_{msg} + 2 * T^{w}_{msg} + EXE_{lct}$$

In the PS technique, relocation incurs 2 wired messages - one message requesting the Global Structure and one to transfer this structure - and one wireless message to re-connect. Therefore,

$$RL_{tm} = 2 * T^{s}_{msg} + T^{w}_{msg}$$

Next, GT_{commit} is calculated. Let GT_{iso} and GT_{atm} be the average time taken to verify the isolation property and enforce the atomicity property respectively. Then the cost to execute the PGSG algorithm ($PGSG_{cost}$) is:

$$PGSG_{cost} = GT_{atm} + GT_{iso}$$

However, the PS technique executes the PGSG algorithm twice for each technique. Therefore GT_{commit} is given by:

$$GT_{commit} = 2^{*} (GT_{atm} + GT_{iso})$$

The atomicity property is enforced by sending two messages to all sites requesting the status of the site-transactions and sending either an abort or commit. As these messages are sent in parallel:

$$GT_{atm} = 2 * T^{s}_{msg}$$

Next, GT_{iso} is formulated. To verify serializability of global transaction T_j, the PGSG algorithm requests Predecessor(T_j) from all sites at which the global transaction executed its vital site-transactions. These sites will, in turn, request (in parallel) Predecessor(T_k) graph for all Candidate nodes T_k in Predecessor(T_j). This process continues until there is no candidate node in any Predecessor graph. At each (parallel) step of the algorithm, in order to have a candidate node T_k in Predecessor(T_j) three conditions must be satisfied:

1. T_k must conflict with T_j
2. T_k must have executed prior to T_j at the site

3. T_k must be must be active.

Note that, in the PS technique, all site-transactions that execute at a site are forced to conflict with each other. Therefore for all T_k that execute before T_j, the probability that T_k conflicts with T_j is 1. At each step of the algorithm, as T_k (of that step) executes prior to T_j (of that step) the probability that T_k is active decreases by a factor of $1/N_{st}$ (where N_{st} is the number of site-transactions in a global transaction) as the time interval since the initiation of that T_k has increased by EXE_{st}. As requests and submissions of Predecessor graphs are carried out in parallel for each Candidate node in any Predecessor graph, GT_{iso} is equivalent to the number of (parallel) steps multiplied by the time taken to execute a step. The number of parallel steps is determined by the probability that the Predecessor(T_j)'s of that step contains a Candidate node. Each step in the algorithm incurs 3 messages: one to request the Predecessor graph, one to submit the Predecessor graph to requesting site, and one to propagate the final outcome to that site. Therefore, GT_{iso} is given by:

$$GT_{iso} = 3 * T^{t}_{pgn} * 1 * \sum^{Nst}_{i=0} (N_{st} - i)/N_{st}$$

5.3.2 The Semantic-PS Technique

In this section, EXE_{st}, RL_{tm}, and GT_{commit}, for the Semantic-PS technique will be formulated. Note that, as the execution of local site-transactions and the steps taken to relocate a mobile user are the same in both the PS and Semantic-PS techniques, EXE_{st} and RL_{tm} for Semantic-PS are the same as those of the PS technique. Thus, EXE_{st} and RL_{tm} are given by:

$$EXE_{st} = 2 * T^{t}_{msg} + 2 * T^{w}_{msg} + EXE_{lcl}$$

$$RL_{tm} = 2 * T^{t}_{msg} + T^{w}_{msg}$$

However, as the Semantic-PS technique executes the PGSG algorithm only once for each technique, GT_{commit} for Semantic-PS is given by:

$$GT_{commit} = GT_{atm} + GT_{iso}$$

Again, as the enforcement of the atomicity property is identical to that of the PS technique, GT_{atm} is given by:

$$GT_{atm} = 2 * T^{s}_{msg}$$

Next, GT_{iso} for the Semantic-PS technique is formulated. Although the algorithm is identical to that of the PS technique, the ticket algorithm used to implicitly obtain the serialization order of site-transactions is different. The ticket algorithm of the Semantic-PS technique does not generate conflicts between all site-transactions that execute at a site. In the Semantic-PS technique, for all T_k that execute before T_j, the probability that T_k conflicts with T_j is determined by the operation (conflict) grouping defined by the service interfaces of the LDBSs. Ideally, the conflict grouping should result in only the operations that actually conflict at the local database being forced to conflict by the respective STM. Let us assume that the probability of conflicts between site-transactions at a site is given by P_{cnf}. Then GT_{iso} for the Semantic-PS technique is given by:

$$GT_{iso} = 3 * T^{s}_{pgn} * P_{cnf} * \sum^{N_{st}}_{i=0} (N_{st} - i)/ N_{st}$$

5.3.3 The Kangaroo Model

Here we formulate EXEst, RL_{tm} and GT_{commit} for the Kangaroo Model introduced in [DBH97] executing under the Compensating mode as this mode ensures atomicity. We assume that Joey transactions consist of sub-transactions that are analogous to site-transactions. First we calculate EXE_{st}. For each site-transaction the mobile user submits the site-transaction to the MSS which then submits it to the respective site, receives a response from that site, and submits the response to the user. Therefore,

$$EXE_{st} = 2 * T^{s}_{msg} + 2 * T^{w}_{msg} + EXE_{lct}$$

In this model, migration is handled by a hand-off process that requires a HandOff KT (HOKT) record be written to the originator's (MSS requesting handoff) MSS' log and a ConTinuing KT (CTKT) record be written to the destination MSS' log. These records register the transfer of control of a global transaction from one MSS to another in their respective log files and create a doubly linked list that describes the migration of the global transaction. The communication cost of writing the CTKT record is 0 as the global transactions is writing information to the current MSS. However, to write the HOKT record into the previous MSS' log the current MSS needs to send a message to the previous MSS along the static network. As relocation incurs one wireless message in order to contact the new MSS, RL_{tm} is given by:

$$RL_{tm} = T^{w}_{msg} + T^{s}_{msg}$$

To commit a global transaction, all log file entries for that global transaction need to be freed. This requires that the entire doubly linked list related to that global transaction be traversed. Therefore:

$$GT_{commit} = 2 * (T^{s}_{msg} * N_{msg})$$

5.4. Evaluation Results

In this section, the tailored analytical models will be used to examine the performance of the PS and Semantic-PS techniques and to compare their performance to that of the Kangaroo technique. The default values for the parameters used in the analytical model are taken from Table 5.1.

5.4.1 Service Time

First, the service time for the PS, Semantic-PS, and Kangaroo techniques will be calculated (Chart 5-1).

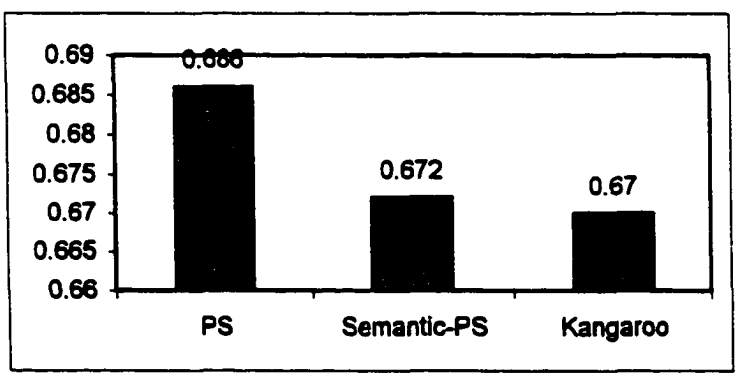


Chart 5-1 : ST_{avg} for Three Transaction Management Techniques

Although the Kangaroo technique has the best average service time, the PS and Semantic-PS techniques are only marginally greater, i.e., 2% and 0.3%, respectively. This result is somewhat counter-intuitive as one would expect the propagation of information during the execution of the PGSG algorithm to utilize noticeable overhead. To clarify this skepticism, the

average time taken to commit a global transaction is calculated (Chart 5-2). These results explain why there is only a small discrepancy between ST_{avg} for all techniques. That is, for all three techniques GT_{commit} is only a small fraction of the total execution time of a global transaction.

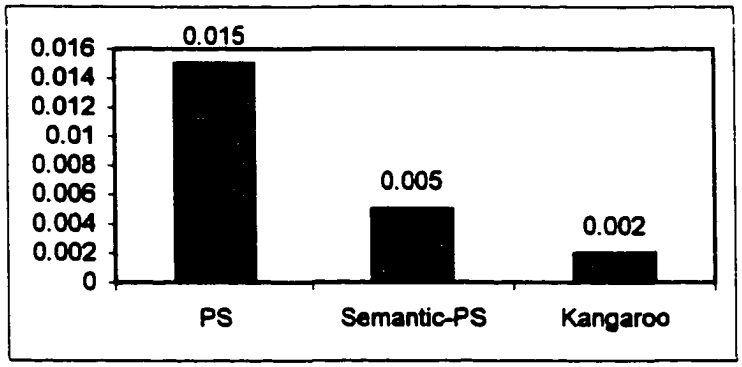


Chart 5-2 : GT_{commit} for Three Transaction Management Techniques

The execution of a global transaction is not very different in any technique. It is the execution of the commit protocol that differs from one technique to another. Here, the time taken to commit a global transaction by the PS technique is greater than the Kangaroo technique by a factor of 7.5, and the time taken by the Semantic-PS technique is greater than the Kangaroo technique by a factor of 2.5. This indicates considerable overhead. However, GT_{commit} accounts for only 2% of ST_{avg} for the PS technique and less than 1% for both the Semantic-PS and Kangaroo techniques and therefore, the effect of propagation is hardly noticeable. Next, the effect of the environmental parameters on ST_{avg} will be studied.

5.4.2 Varying The Number of Site Transactions in a Global Transactions

Here, the effect of the size of the global transaction (i.e., N_{st} - the number of site-transaction in a global transaction) upon ST_{avg} will be evaluated. Specifically, ST_{avg} will be calculated for N_{st} = 2, 4, 6, 8, 10. For this analysis, N_{dcm} N_{mgr} and P_{cnf} are set to the default values

in Table 5.1. The service time for each technique is given in Table 5-2. These results indicate that although the Kangaroo technique offers the best performance, ST_{avg} for the PS technique is only 1.6% greater than the Kangaroo technique and the PS technique is less than 1% greater than the Kangaroo technique.

N_{st}	2	4	6	8	10
PS	0.394	0.715	1.0	1.332	1.702
Semantic-PS	0.385	0.70	0.987	1.306	1.687
Kangaroo	0.385	0.70	0.986	1.305	1.686

Table 5-2: Service Time for varying N_{st}

5.4.3 Varying Number of Disconnections for a Global Transaction

Next, ST_{avg} will be calculated for different values of N_{dcn}. As at most only one disconnection can have any affect on the execution of a site-transaction, the default value for N_{st} will be set to 10 in order to accommodate sufficient test cases, i.e., N_{dcn} = 2, 4, 6, 8, 10. N_{mgr} is set to 1. The service time for each technique with respect to N_{dcn} is given in Table 5.3. Again, the difference in service time is insignificant - ST_{avg} for the PS technique is approximately 1% greater than that of the Kangaroo technique while the Semantic-PS techniques is less than 1% greater than that of the Kangaroo technique.

N_{st}	2	4	6	8	10
PS	1.592	1.658	1.726	1.796	1.865
Semantic-PS	1.56	1.626	1.694	1.763	1.833
Kangaroo	1.56	1.625	1.693	1.762	1.832

Table 5-3: Service Time for varying N_{dcn}

5.4.4 Varying Number of Migrations for a Global Transaction

Next, the effect of the number of migrations during the course of execution of a global transaction will be evaluated for each technique. As each migration also causes a disconnection, N_{dcn} will be set to the value of N_{mgr} (i.e., $N_{dcn} = N_{mgr}$). Once again, as at most only one disconnection (and therefore, at most one migration) can have any effect on the execution of a site-transaction, the default value for N_{st} will be set to 10 in order to accommodate sufficient test cases, i.e., N_{mgr} = 2, 4, 6, 8, 10. The service time for each technique with respect to N_{mgr} is given in Table 5.4. Again, the difference in service time is insignificant - ST_{avg} for the PS technique is approximately 1% greater than that of the Kangaroo technique while the Semantic-PS techniques is less than 1% greater than that of the Kangaroo technique.

N_{st}	2	4	6	8	10
PS	1.662	1.859	2.057	2.254	2.451
Semantic-PS	1.630	1.827	2.025	2.222	2.419
Kangaroo	1.629	1.826	2.024	2.221	2.418

Table 5-4: Service Time for varying N_{mgr}

From the evaluations carried out in Sections 5.4.2, 5.4.3, and 5.4.3, it is clear that the number of site transactions, the number of disconnections, and the number of migrations have a similar effect on ST_{avg} for all techniques. This is due to the fact that, as concluded in Section 5.4.1, the time taken to commit a global transaction accounts for only a small percentage (<= 2%) of the total execution time of a global transaction in all techniques. Although the PS and Semantic-PS techniques enforce the isolation property by executing the PGSG algorithm, the overhead with respect to propagation is not a dominant factor in ST_{avg}. Next, ST_{avg} will be evaluated with respect to the average communication time on the static network (T^{s}_{msg}) and the

probability of conflicts for site-transactions (P_{cnf}) to determine their effect on the average service time for the three techniques.

5.4.5 Varying Time to Transmit a Message on the Static Network

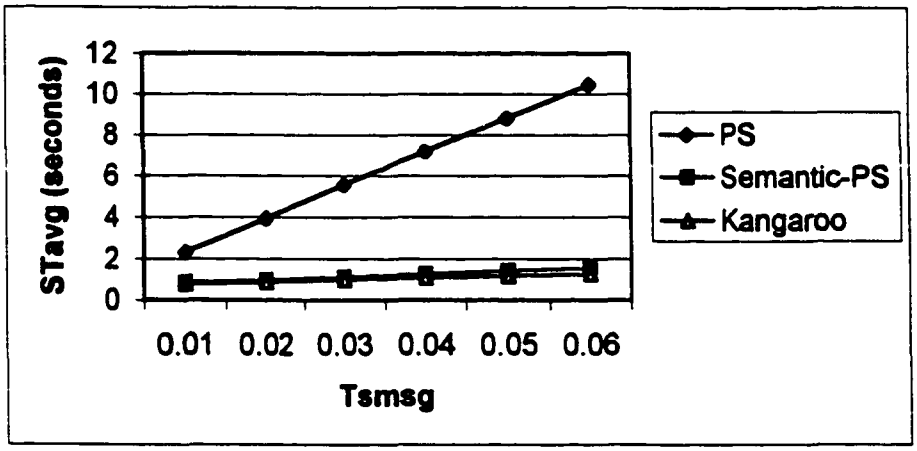
In this test the effect of the communication cost on the static network is evaluated for all techniques. Specifically, ST_{avg} will be calculated for T^s_{msg} = 0.01, 0.02, 0.03, 0.04, 0.05, 0.06. T^s_{msg} is the time taken to transmit a 1 Kb message on the static message. Along with T^s_{msg} it is also necessary to vary the value of T^s_{pgm} as it represents the time taken to transmit a 10Kb message on the static network. Accordingly T^s_{pgm} is set to 10 * T^s_{msg}, i.e., T^s_{pgm} = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6. The service time for each technique is given in Table 5-5.

T^s_{msg}	0.01	0.02	0.03	0.04	0.05	0.06
PS	2.3	3.93	5.56	7.19	8.82	10.45
Semantic-PS	0.82	0.96	1.11	1.26	1.41	1.55
Kangaroo	0.77	0.87	0.97	1.07	1.17	1.27

Table 5-5: Service Time for varying T^s_{msg}

This result indicates that the three techniques respond differently to changes to the communication time on the static network (Figure 5-2). The results show that the communication time on the static network has a much greater effect on the PS technique than the Semantic-PS and Kangaroo technique. For each increment of 0.01 seconds in the time taken to transmit 1 Kb on the static network, ST_{avg} of the PS technique increases by 1.63 seconds as opposed to 0.145 seconds for the Semantic-PS technique and 0.1 seconds for the Kangaroo technique. That is, the rate of growth of ST_{avg} for a 0.01 second increase per 1Kb message on the static network is approximately 16 times greater for the PS technique than the Kangaroo

technique. In comparison, the Semantic-PS techniques is only 0.45 times greater than the Kangaroo technique.



Graph 5-2 : Service Time for varying Tsmsg

Unlike in the previous experiments, the rapid growth of ST_{avg} of the PS technique with respect to T^{s}_{msg} (T^{s}_{pgn}) can be explained as follows: Unlike the Kangaroo the technique, the PS and Semantic-PS techniques enforce the isolation property by executing the PGSG algorithm. The PGSG algorithm utilizes information propagation to verify serializability. Serializability information is propagated by passing messages between STCs and MSSs residing on the static network. Therefore, any increases to T^{s}_{msg} relative to the rest of the environment variables will have an impact on ST_{avg} for the PS and Semantic-PS techniques.

The fact that this impact is more prominent for the ST_{avg} of the PS technique can be explained as follows: The volume of information being propagated by the PGSG algorithm is determined by the number of conflicts between site-transactions. In order to obtain local serialization information, the PS and Semantic-PS techniques force conflicts between site-transactions that execute at the same site. The Semantic-PS technique forces conflicts only

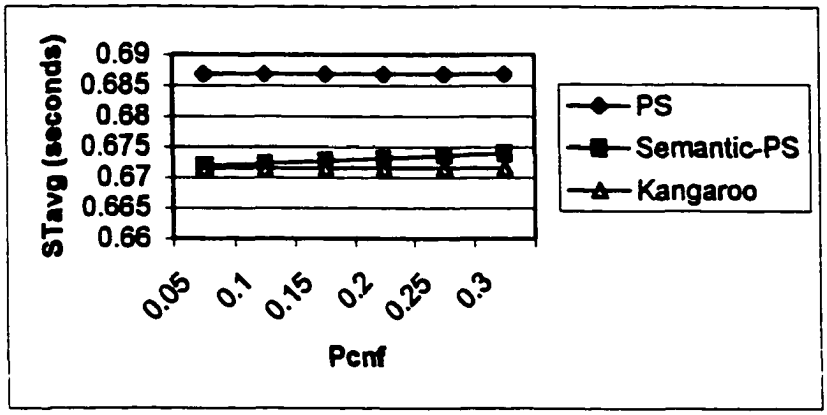
between site-transactions that potentially conflict with each other. On the other hand, the PS techniques forces conflicts between all site-transactions that execute at each site. In addition, the PS technique executes the PGSG algorithm twice, information is propagated a second time for every global transaction.

5.4.6. Varying Probability of Conflicts

Finally, the effect of the probability of conflicts between site-transactions is evaluated for all techniques. Specifically, P_{cnf} will be calculated for P_{cnf} = 0.05, 0.1, 0.15, 0.2. 0.25, 0.3. Here again, N_{dcn} N_{mgr} and T^{e}_{msg} are set to the default values in Table 5.1. The service time for each technique is given in Table 5-6. These results indicate that P_{cnf} has no effect on the PS technique and the Kangaroo technique (Graph 5-3). In the case of the PS technique, P_{cnf} has no effect on the service time because all site-transactions that execute at a given site are forced to conflict with each other as each LDBS maintains only one ticket. In the case of the Kangaroo model P_{cnf} has no effect on the service time as this technique does not enforce the isolation property. Thus, P_{cnf} affects only the Semantic-PS technique. However, the increase in ST_{avg} for the Semantic-PS technique is only marginal, i.e., an increase of 0.0004 seconds for every 5% increase in the probability of conflicts. Once again, this is due to the fact that the time taken to propagate information on the static network is relatively small compared to the time taken to transmit a message on the wireless network as well as the time taken to address migration.

P_{cnf}	0.05	0.1	0.15	0.2	0.25	0.3
PS	0.6868	0.6868	0.6868	0.6868	0.6868	0.6868
Semantic-PS	0.6719	0.6723	0.6727	0.6731	0.6735	0.6739
Kangaroo	0.6715	0.6715	0.6715	0.6715	0.6715	0.6715

Table 5-6: Service Time for varying P_{cnf}



Graph 5-3: Service Time Vs P_{cnf}

5.5 Summary and Conclusion

In conclusion this analysis suggests that in certain environments, the average service time of these techniques is comparable to the Kangaroo technique which does not enforce the isolation property nor does it attempt to minimize the ill effects of the prolonged execution of mobile global transactions. For the values of the model parameters developed in Section 5.2, ST_{avg} of the PS technique is only 2% greater than the Kangaroo technique and the Semantic-PS technique is only 0.3% greater than that of the Kangaroo technique. In addition, these experiments reveal that any changes to environment variables N_{st} N_{dcn} and N_{mgr} have a similar effect on all three techniques. In each case, ST_{avg} for the PS techniques was approximately 1% greater than that of the Kangaroo technique; ST_{avg} for the Semantic-PS technique was less than 1% greater than that of the Kangaroo technique.

However, ST_{avg} for the PS technique deteriorates rapidly with respect to the communication time on the static network. For every 0.01 second increase in the time taken to transmit 1 Kb over the static network, the rate of change of ST_{avg} for the PS technique is approximately 16 times greater (1.63 seconds as opposed to 0.1 second) than that of the Kangaroo technique; the rate of change of ST_{avg} for the Semantic-PS technique is approximately 1.45 times greater than that of the Kangaroo technique.

Chapter 6

SIMULATION

This research introduces two new concepts to transaction management in the MMDB environment. First, it introduces the notion of pre-serialization, that is, verifying the isolation property of mobile transactions prior to their completing their execution. Second, it introduces a new technique called the PGSG algorithm to verify the isolation property of global transactions in large heterogeneous environments based on partial global serialization graphs and information propagation. As new concepts are introduced it is important that, as part of this research, the PS and Semantic-PS techniques be simulated in order to observe (and learn about) the behavior of these techniques and to make recommendations for future researchers.

The primary goals of the simulation are twofold: First, the simulation models will be used to measure the service time of the PS, Semantic-PS, and Kangaroo techniques in order to validate the analytical models developed in Section 5.1. Second, the simulation models for the PS and Semantic-PS techniques will be used to study the effectiveness of pre-serialization in achieving its design goal, i.e., minimizing the unfair treatment of mobile transactions due to their extended execution time.

6.1 The Simulation Model

The ARENA [KSS98] simulation software is used to carry out the simulation experiments. ARENA is a high-level simulator that allows one to model discrete event-based simulation models. The execution of a global transaction in the MMDB environment can be defined by a sequence of discrete events that occur during its execution, i.e., its creation,

submission of a site-transaction, completion of a site-transaction, completion of the global transaction, potential disconnection and migration, etc.. Therefore, the ARENA software can be used to simulate the PS, Semantic-PS, and Kangaroo techniques. As ARENA is used for simulation, the simulation models will be described using simulation constructs similar to those available in the ARENA software. First, the basic ARENA constructs that are used to describe the models will be introduced.

a) The Simulate Module

The Simulate module is used to control the simulation. This module is used to specify the time of simulation, the number of runs in each simulation, the number of entities to be created, etc.

b) The Create Module

The Create module is used to create entities. Entities are dynamic objects in the simulation that are transferred from module to module in the simulation model. Each entity is associated with zero or more attributes that define the state of the entity at any given time.

c) The Dispose Module

The Dispose module is used to remove entities from the simulation and to dispose them. This module can also be used to collect statistics with respect to entities.

d) The Choose Module

When multiple simulation paths exist, the Choose module is used to determine the appropriate path that the entity needs to take based on some criteria. The criteria used to determine the path can be based on the current state of the entity or some distribution function.

e) The Assign Module

The Assign module is used to assign values to attributes of an entity.

f) The Delay Module

The Delay module is used to delay entities for some period of time before being sent to the next module in the simulation.

g) The Station Module

The Station module passes entities that arrive at that module to the next module in the simulation model. They perform no particular task and are used mainly to represent different simulation paths.

h) The While End-While Modules

The While End-While module is used to model while loops in the simulation. The entity remains in the While End-While loop as long as it satisfies the condition that is defined in the While module.

i) The If End-IF Modules

The If End-If module is used to represent conditional statements. An entity that arrives at an If module will pass through all modules encapsulated within the If End-IF module if it satisfies the condition set forth in the If.

j) The Resource Module

The Resource module is used to represent resources available to entities in the simulation.

k) The Seize and Release Modules

The Seize module is used to model entities seizing resources defined using the Resource module. The Release module is used to release resources that have been seized by an entity.

l) The Write Module

The Write Module is used to write information to a file. This module is used to record the intermediate state of entities in an external file.

m) The Tally Module

The Tally module is used to collect statistics. These statistics are reported at the end of the simulation.

6.2 The Common Simulation Model

In order to eliminate repetition, the general simulation model will be described in this section. The details of each transaction management technique will be described in subsequent sub-sections. The simulation process for all techniques can be broken into three steps: the creation of global transactions, the simulation of global transactions, and the final deletion of the global transactions from the simulation (Figure 6-1). In all simulation models, global transactions are modeled as entities that are created by the Create module. Each global transaction is associated with a set of attributes that are initialized by this module. They are

GTID, TransType, NumSites, SiteList, DcnDelay, and StartTime. The GTID attribute is assigned a unique (consecutive) identifier. The TransType is assigned either "Static" or "Mobile" indicating that the transaction is either a static or mobile transaction.

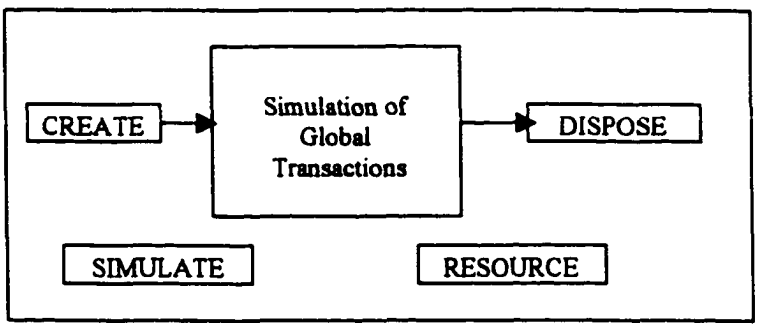


Figure 6-1: Overview of Simulation Model

NumSites is assigned the number of site-transactions in that global transaction. Once NumSites has been initialized the SiteList is assigned the list of sites that are to be accessed by that transaction. Each SiteList is generated such that it does not contain duplicate sites. This ensures that a global transaction does not access any site more than once during its execution as required by the PS and Semantic-PS techniques. Each site to be accessed is assigned as vital or non-vital indicating the type of site-transactions to be executed at that site. The attribute DcnDelay is used to record the time a site-transaction is to be delayed if a disconnection occurs. This attribute is initially set to 0. Finally, the time on the simulation clock at which the global transactions was created is assigned to the StartTime attribute.

After being created by the Create module, global transactions are transferred to the Simulation of Global Transactions module. This module is used here to represent the simulation of the execution of global transactions for a given transaction management technique.

After completing the simulated execution, global transactions are transferred to the Dispose module. In this module, statistics such as the time taken to execute the global

transaction are collected before being disposed from the system. The Simulate module is used to control each simulation process.

The Resource module is used to represent sites (i.e., STMs) in the MMDBS. In Section 5.1 it was determined that the average mid-size DBMS system is capable of executing an average of 333 transactions per second. Therefore each resource in this simulation is modeled as a resource with unbounded capacity. That is, each site is capable of executing multiple site-transactions concurrently without any significant performance deterioration.

6.3 Tailored Simulation Models

6.3.1 Disconnection and Migration

In all techniques disconnection and migration are modeled the same way. Therefore the details of disconnection/migration (D/M) model will be presented in this section and be represented as the D/M module in the tailored models.

During the execution of a site-transaction of a mobile global transaction, the user may be disconnected. This is modeled using the Choose module (Figure 6-2). The Choose module decides between a continuous execution and disconnected execution modeled using the Continuous and Disconnection Stations, respectively. If a resource is sent to the Disconnection module, then the subsequent AddDelay Assign module assigns the delay to be incurred to the DcnDelay attribute.

In addition, each disconnection may represent a migration which is modeled using a second Choose module. Once again entities will be sent to either a Migration or No Migration module. If the entity arrives at the Migration module, the DcnDelay is further incremented at the second AddDelay Assign module by the time taken to address migration. Finally, the entity

is delayed at the Delay module by the amount specified in the DcnDelay attribute, the DcnDelay is reset to zero, and the entity is transferred to the next module in the simulation.

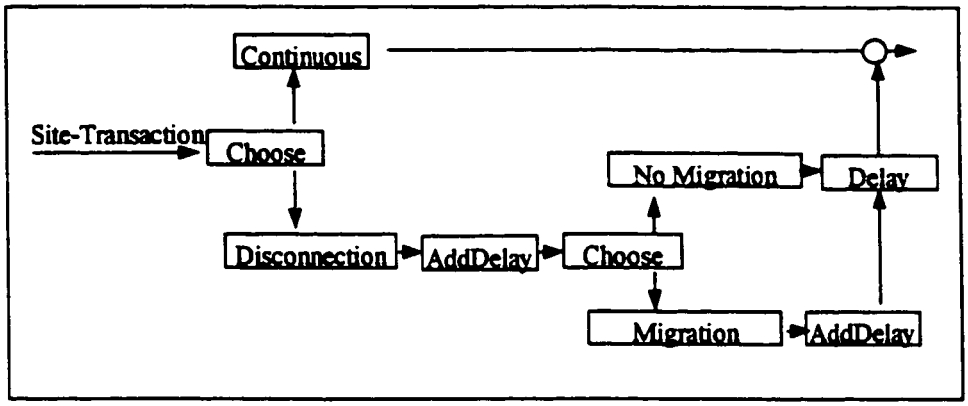


Figure 6-2: D/M module

6.3.2 Simulation Model for the PS Technique

This section details the "Simulation of Global Transactions" step of the common simulation model for the PS technique. Due to the complexity of the PGSG algorithm, especially propagation, the commit of global transactions cannot be simulated using ARENA constructs. Therefore, the simulation is carried out in two steps. First, an ARENA model is used to simulate the creation and execution of global transactions. This model simulates the entire life of the global transaction except the execution of the PGSG algorithm. The simulation records all relevant events - the creation of global transactions along with its type (i.e., Static or Mobile), the execution of each site-transaction, the occurrence of disconnection and migration, etc. - in an external file. Next, a Java application is used to simulate the PGSG algorithm by reconstructing the entire execution sequence recorded by the ARENA simulation. This program simulates the PGSG algorithm and adds the time taken to commit the global transaction to the service time recorded by ARENA.

6.3.2.1 The ARENA Model of the PS Technique

Global Transactions created by the Create module are transferred to the Record module labeled Rstart where their creation is recorded in an external text file labeled "PS.dat" (Figure 6-3). For each global transaction, the triple <"CREATE", GTID, TransType> is recorded. For GTID and TransType it is the attribute values that are recorded. Next, global transactions are transferred to the Choose module. The Choose module will transfer the global transaction to the appropriate path based on the value of the TransType attribute.

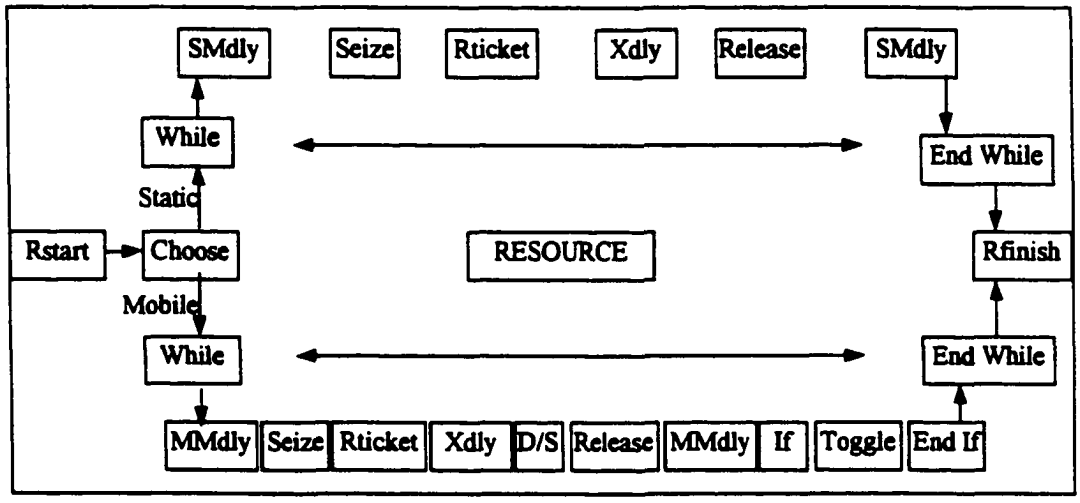


Figure 6-3: Simulation of Global Transactions – PS technique

The execution of static global transactions is modeled using a While End-While loop. Each loop simulates the execution of one site transaction. The simulation of a site-transaction consists of six steps. First, the site transaction is transferred to the Delay module labeled SMdly (Static Message delay) to simulate the time taken to submit the site-transaction to the STM. Next, the site-transaction will seize the resource representing that site at which it is supposed to execute. This is modeled using the Seize module. Next, it is transferred to the Record module labeled Rticket where its execution at that site is recorded in PS.dat. For each site-transaction

the quadruple <"EXECUTE", GTID, SiteId, Criticality> (where Criticality represents whether the site-transaction is vital or non-vital) is recorded. In essence, this step emulates the ticket value obtained by the site transactions at the respective site. Next, the site-transaction is delayed for some period of time to simulate its local execution. This is modeled using the Delay module labeled Xdly. Next, the Release module is used to release the seized site back to the system. Finally, the site-transaction is transferred to the Delay module labeled SMdly to simulate the time taken to return the outcome of the site-transaction to the user.

Similar to static global transactions, the execution of mobile global transactions are modeled using a While End-While loop as well. However, each loop consists of ten steps. First, the site-transaction is transferred to the Delay module labeled MMdly (Mobile Message delay) to simulate the time taken to submit the site-transaction to the user. The next Seize Record and Xdly modules perform the same functions as in the simulation of static transactions. The next D/S module (Section 6.1.3) simulates potential disconnection and migration that may occur during the execution of site-transactions. At the end of site-transaction execution, the Release module releases the site resource. The next three blocks model pre-serialization, i.e., the toggle operation. The toggle operation is modeled using an If End-If block. If the last site-transaction that was just simulated represents the last vital-site transaction of a global transaction, then the global transaction is toggled by the Record module labeled Toggle. This module records that the global transaction is to be toggled in PS.dat. For each mobile global transaction to be toggled the tuple <"TOGGLE", GTID> is recorded.

Upon completing their execution, global transactions (static and mobile) are transferred to the Record module labeled Rfinish which records the completion of the simulation of the global transaction in PS.dat. For each global transaction, the triple <"COMMIT", GTID, ServiceTime> where ServiceTime is the time taken to simulate the global transaction (i.e.,

current ARENA time - StartTime), is recorded. Each global transaction is then transferred to the Dispose module to be removed from the simulation.

6.3.2.2 The PGSG Java Application

Once the ARENA simulation is completed, the text file PS.dat contains the complete ordered sequence of events necessary to trace the execution of all global transactions. This sequence of events is used to simulate the PGSG algorithm using a Java application and to determine the following: 1 - whether each global transaction is to be toggled committed or aborted; 2 - what is the number of parallel steps (i.e., parallel message transmissions) executed by the PGSG algorithm; and 3 - reporting the results of the simulation. This application is described next.

First, the application creates a list of Site objects, each object represents a site in the simulation environment. Each Site object is associated with a Graph object that represents the Site Serialization Graph (i.e., SSG). Initially, each Graph object contains an empty set of nodes. The Java application then processes each entry in PS.dat file. The PS.dat file contains an ordered list of events where each event is one of the following types:

- <"CREATE", GTID, [Mobile or Static]> - global transaction of type Mobile or Static with identifier GTID was created.
- <"EXECUTE", GTID, SiteId, [vital or non-vital]> - global transaction GTID executed a site-transaction of type vital or non-vital at site SiteId.
- <"TOGGLE", GTID> - global transaction GTID executed the toggle operation.
- <"COMMIT", GTID, ServiceTime> - global transaction GTID completed its execution. The service time for the transaction is given by ServiceTime.

Each event is processed as follows. For each CREATE event, the application creates a Global Transaction object with the corresponding GTID and TransType. Each Global

Transaction object contains an additional attribute named Propagation Count. This attribute is used to keep a count of the number of parallel steps (i.e., parallel message transmissions) executed by the PGSG algorithm when the transaction is toggled and/or committed. For each EXECUTE event, the application adds a node labeled GTID in front of the SSG of the specified site. Essentially, this represents the serialization order of the site-transaction at that site.

For each TOGGLE and COMMIT event, the application executes the PGSG algorithm and determines whether the global transaction is to be toggled committed or aborted. The complete PGSG algorithm for the PS technique is given in Section 3.1.4.5. Each time the global transaction executes the PGSG algorithm, the Propagation Count is updated accordingly. If the global transaction is to be aborted, the corresponding Global Transaction object is marked as Aborted. If the event is a TOGGLE event and the PGSG algorithm succeeds, then the Global Transaction is marked as Toggled. If the event is COMMIT and the operation is successful, then the Global Transaction is marked as Committed and its execution time is set to the ARENA service time.

Once all events in PS.dat have been processed the application processes all Global Transaction objects and reports the final results of the simulation. For each simulation, the following results are reported:

- The ratio of static-global-transactions-aborted / static-global-transactions-simulated
- The ratio of mobile-global-transactions-aborted / mobile-global-transactions-simulated
- The average service time of all successful global transactions. Here the service time of a global transaction is the sum of the ARENA service time and the time taken to commit the global transaction, i.e., Propagation Count multiplied by the time taken to transmit one message on the static network.

6.3.3 Simulation Model for the Semantic-PS Technique

This section details the "Simulation of Global Transactions" step of the common simulation model for the Semantic-PS technique. Once again, the simulation is carried out in two steps. First, an ARENA model is used to simulate the creation and execution of global transactions. Here, the sequence of events that occur during the ARENA simulation is written to a file named SemPS.dat. Next, a Java application is used to simulate the PGSG algorithm by reconstructing the entire execution sequence recorded by the ARENA simulation. This is simulated as described in the following sub-sections.

6.3.3.1 The ARENA Model of the Semantic-PS Technique

The execution process of global transactions in the Semantic-PS technique is very similar to that of the PS technique. The differences between the PS and Semantic-PS techniques are: 1 – The Semantic-PS technique enforces atomicity and isolation properties only on the set of vital site-transactions; 2 – Mobile global transactions execute the PGSG algorithm only once – during the toggle phase; and 3- The ticket method used to obtain the local serialization order forces conflicts only between site-transactions (that execute at the same site and) potentially conflict with each other. Here, 2 and 3 are related to the execution of the PGSG algorithm. As the ARENA model does not simulate the execution of the PGSG algorithm, these do not have to be modeled in the ARENA model.

In order to ensure that the A/I properties are enforced only on the set of vital site-transactions, the ARENA model records only the execution of vital site-transactions in SemPS.dat file. This can be modeled by encapsulating the Rticket Record module within an If End-If block (Figure 6-4). If the site-transaction is a vital site-transaction, then it is recorded;

otherwise it is not. Note that the execution of non-vital site-transactions needs to be simulated in ARENA as it affects the service time of a global transaction.

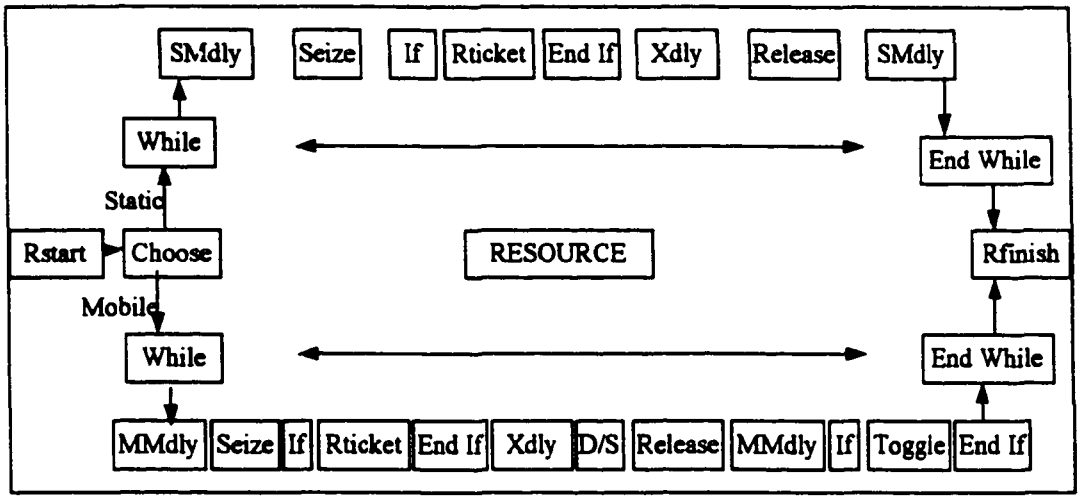


Figure 6-4: Simulation of Global Transactions – Semantic-PS technique

6.3.3.2 The PGSG Java Application

The Java application that simulates the PGSG algorithm for the Semantic-PS technique is very similar to that of the PS technique as well. The only differences are:

1. All site-transactions that execute at one site do not conflict with each other.
2. Mobile global transactions do not execute the PGSG algorithm during the commit phase.

These change are implemented as follows. The probability of a site-transaction conflicting with another site-transaction is defined by the environment variable P_{cnf}. As each site maintains multiple tickets that need to be distinguished, each Node object in the SSG graph is associated with an integer variable called ticket. Whenever the Java application encounters an EXECUTE event in the SemPS.dat file, this event is processed by adding a Node object to the SSG of that site with the respective GTID and the ticket is assigned a random integer value in the range $1..1/P_{cnf}$. For example, if $P_{cnf} = 0.1$, i.e., there is a one-in-ten chance that a site-

transaction conflicts with another site-transaction at that site, then the ticket will be assigned a random value in the range of 1..10. Unlike for the PS technique, nodes in the SSG are linked only if they have the same ticket value. Therefore, the probability of a site-transaction conflicting with another site-transaction that executes at the same site is 1/10, i.e., P_{cnf}.

The CREATE event and the TOGGLE event are processed identical to that of the PS technique. However, when a COMMIT is encountered, the application executes the PGSG algorithm only if the TransType is Static as in Semantic-PS, mobile transactions do not execute the PGSG during the commit. Note that the PGSG algorithm that is executed for each TOGGLE or COMMIT event is the one defined for the Semantic-PS technique in Section 4.2.1.

6.3.4 Simulation Model for the Kangaroo Technique

This section details the "Simulation of Global Transactions" step of the common simulation model for the Kangaroo technique. The entire simulation of the Kangaroo technique is carried out using an ARENA model which is described below.

The Kangaroo technique does not distinguish between static global transactions and mobile global transactions. Therefore, the simulation model contains only one path. In the Kangaroo model site-transactions are encapsulated in a Joey transaction. In the Kangaroo simulation model, in addition to the GTID, TransType, NumSites, SiteList, DcnDelay, and StartTime attributes, each global transaction entity contains two additional attributes labeled MigrationCount and MessageDelay. The MigrationCount attribute is used to keep track of the number of migrations that occur during the execution of a global transaction. The MessageDelay attribute is assigned the time taken to submit a site-transaction on the appropriate medium depending on the type of global transactions. That is, for mobile transactions MessageDelay is assigned the time taken to transmit a message on the wireless network; for

static transactions MessageDelay is assigned the time taken to transmit a message on the wired network.

The execution of a global transaction is simulated using a While End-While loop. Each loop simulates the execution of a single site-transaction. The execution of a site-transaction consists of seven steps (Figure 6-5). First, the site-transaction is delayed for a time period of MessageDelay to simulate the submission of a site-transaction to the MMDBMS. This is modeled using the Mdly Delay Module. Next, the global transaction seizes the Site which is modeled using the Seize module; executes the site-transaction which is modeled using the Delay module labeled Xdly; and releases the seized site which is modeled using the Release module.

Next, potential disconnection and migration need to be modeled for mobile global transactions. If the transaction is a mobile transaction (which is modeled using the If module), then the potential disconnection and migration are modeled using the D/M module described in Section 6.1.3. In this module if a migration does occur, the MigrationCount is incremented. In essence, the MigrationCount represents the number of Joey transactions that are created for the global transaction.

Once the global transaction has completed its execution, it is transferred to the Commit module. This module is a Delay module and delays the transaction to simulate the execution of the commit protocol. Each global transaction is delayed for a time period equivalent to the commit time of the global transaction, i.e., 2 * MigrationCount * time taken to transmit a message on the static network. Next, the global transaction is transferred to the Tally module which collects statistics on the simulation, i.e., the average service time for global transactions and the individual tallies for static and mobile transactions.

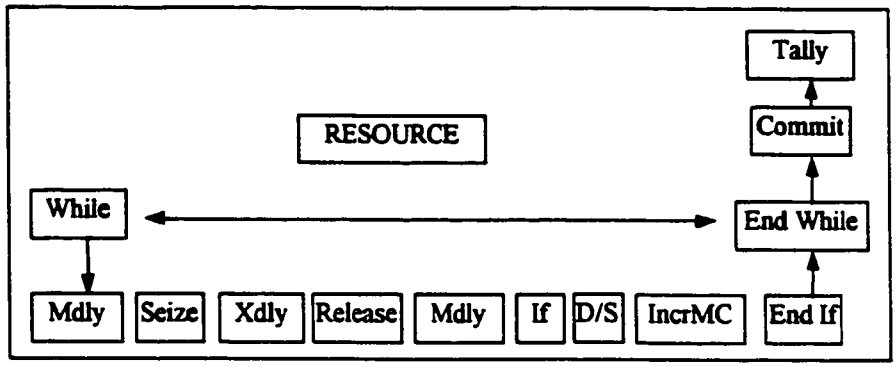


Figure 6-5: Simulation of Global Transactions – Kangaroo technique

6.4 The Simulation Environment

In this section the default values used to define the simulation environment will be documented. A summary of the parameters is presented in Table 6-1. As one of the primary goals of this simulation is to study the effectiveness of pre-serialization, some parameter values have been chosen especially to facilitate this goal. In essence, it was necessary to define a simulation environment that provided a sufficient number of isolation property violations so that the effectiveness of pre-serialization could be studied.

The number of sites in the simulation environment is one such parameter. The typical MMDB environment will consist of a very large number of sites. As the number of sites in the system increases, the probability of isolation property violations decreases especially as it is assumed that each site is equally likely to be accessed. Therefore it was necessary to perform the simulation over a small number of sites in order to magnify isolation property violations. The number of sites in this simulation (TotalSites) has been set to 10.

Global transactions are created based on an exponential distribution with a mean of 0.2 time units. The exponential distribution is chosen as it is often used to model inter-event times

in random arrival processes [KSS98]. A probability distribution is used to label transactions as either Static or Mobile transactions.

A triangular distribution (with a minimum of 3, average of 4, and maximum of 5) has been chosen to describe the number of site-transactions in a global transaction (NumSites). The triangular distribution is chosen for two reasons: 1 – it is commonly used in situations in which the exact form of the distribution is not known [KSS98]; and 2 – it is bounded by a minimum and maximum value. Note that on average 40 percent of the sites in the system are being accessed by a global transaction. Once again, a high value has been chosen in order to magnify the isolation property violations.

The execution time of a site-transaction is described using a triangular distribution. The mean time for execution has been derived from the values given in Table 5.1.

The length of the vital stage of a global transaction (VT_{stg}) is set to be 50% of the length of the global transaction. This value has been chosen as initial simulation results indicate that 50% is the value for VT_{stg} at which neither static nor global transactions will be penalized. The average time of disconnection has been taken from the values given in Table 5.1.

The probability of a site-transaction conflicting with another site-transaction at the same site (P_{cnf}) is set to 0.5. (Note that this parameter applies only to the Semantic-PS technique.) Once again a high value is chosen to magnify the isolation property violations. The parameters P_{dcn}, P_{mgr}, T^{s}_{msg}, T^{w}_{msg}, and T^{s}_{pgn}, have been taken from Table 5.1. The length of a simulation run (Len) has been set to 7200 time units. Assuming that a time unit represents one second, the length of a simulation run represents a 2-hour period.

Finally, as in the analytical evaluation, this simulation assumes that each site is equally likely to be accessed by any site-transaction such that a global transaction does submit two site-transactions at the same site.

Parameter	Description	Default Value
TotalSites	Number of sites in the MMDBS	10
GT_{int}	Inter-arrival time of global transactions	EXPO(0.2) time units
GT_{ratio}	The ration of mobile global transactions to static global transactions	0.3/0.7
NumSites	Number of site-transactions in a global transaction	TRIA(3, 4, 5)
EXE_{lcl}	avg. local execution time of a site-transaction	TRIA (0.001, 0.003, 0.005)
VT_{stg}	The size of the vital stage of a global transaction (as a fraction of the total length)	0.5
DCN_{tm}	Avg. time between disconnection and relocation	0.1 time units
P_{cnf}	Probability of a site-transaction conflicting with another site-transaction (at the same site)	0.5
P_{dcn}	The probability of a disconnection during the execution of a site-transaction	$\lceil NumSites/3 \rceil$/NumSites
P_{mgr}	The probability of a migration during the execution of a site-transaction	$\lceil P_{dcn}/3 \rceil$
T^{s}_{msg}	avg. time to transmit a message on the static (wired) network	0.0001 time units
T^{w}_{msg}	avg. time to transmit a message over the wireless medium	0.07 time units
T^{s}_{pgs}	avg. time to transmit a Predecessor graph (or propagate a PGS graph) from site to site along the static network	0.001 time units
Len	Length of simulation	7200 time units

Table 6-1: Environment parameters for simulation

6.5 Service Time for Global Transactions

In this section, the simulation models will be used to obtain the service time (i.e., ST_{avg}) for the PS Semantic-PS and Kangaroo techniques. As the primary purpose of this experiment is to validate the analytical model (and vice versa) by reproducing the experiment carried out in Section 5.4.1 - Service Times, the MMDB environment will need to duplicate the same environment. The default values will be taken from Table 6-1 for every parameter, except P_{cnf} and GT_{ratio}, as the values represent the same environment defined in Table 5.1. In this experiment P_{cnf} will be set to 0.05 - the value defined in Table 5.1. In the experiment carried out is Section 5.4.1, all global transactions are mobile global transactions. To duplicate this scenario, GT_{ratio} is set to 1/0.

The simulation is repeated 20 times for each technique. The average ST_{avg} is given in Figure 6-6. These results show that the simulated ST_{avg} for all techniques is less than the analytical ST_{avg}. However, this deviation is very minimal. That is, for the PS technique, ST_{avg} obtained from the analytical model is only 1.3% greater than that obtained from the simulation model. For the Semantic-PS and Kangaroo techniques, ST_{avg} obtained from the analytical models is less than 1% greater than that obtained from the simulation models. Once again, it can be concluded that ST_{avg} for the PS and Semantic-PS technique is not significantly higher than that of the Kangaroo technique.

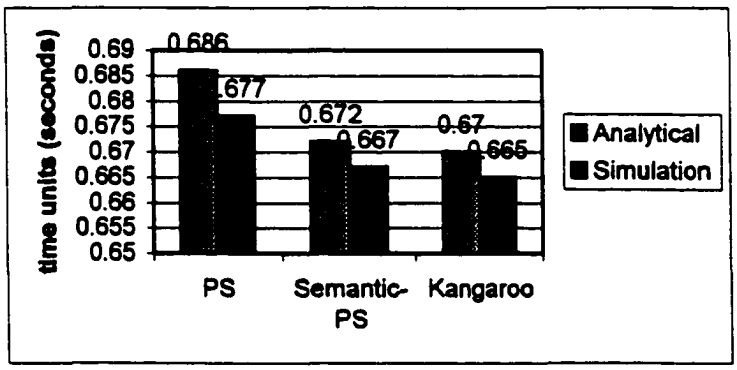


Figure 6-6: ST_{avg} for Three Transaction Management Techniques Using Analytical Models and Simulation Models

Next, the 95% confidence interval over the 20 runs is calculated. The 95% confidence interval for the PS technique is 0.677 ± 0.009. Thus 95% of the simulation results reside in the interval (0.668 - 0.686). The 95% confidence interval for the Semantic-PS technique is 0.667 ± 0.006, i.e., (0.661 - 0.673). The 95% confidence interval for the Kangaroo technique is 0.665 ± 0.006, i.e., (0.659 - 0.671). Note that the ST_{avg} obtained from the analytical model for each technique is within the 95% confidence interval obtained from the simulation model. Thus, it can be stated with confidence that the simulation model complements the analytical model.

6.6 Hypothesis Testing

In the previous section it was concluded that the service times for the PS and Semantic-PS techniques are not significantly higher than the service time of the Kangaroo technique. This conclusion was drawn from an informal evaluation of the simulation results. In this section, hypothesis testing will be used to examine whether ST_{avg} for the PS and Semantic-PS techniques are significantly greater than that of the Kangaroo technique. Specifically, the hypothesis test concerning means will be used to determine whether it can be established that ST_{avg} for the PS and Semantic-PS techniques are different from that of the Kangaroo technique at the 0.05 level of significance.

6.6.1 Hypothesis Test for the PS Technique

In this section, the claim that ST_{avg} for the PS technique is not significantly higher than that of the Kangaroo technique is tested. As the aim of this hypothesis test is to establish that ST_{avg} for PS is not significantly higher, the null hypothesis will claim the contrary. That is, the null hypothesis states that ST_{avg} for the PS technique is significantly different than that of the Kangaroo technique. The null hypothesis H_0, and alternate hypothesis H_1 are stated as:

H_0: ST_{avg} for the PS technique $\neq$ ST_{avg} for the Kangaroo technique (i.e., 0.665)

H_1: ST_{avg} for the PS technique = ST_{avg} for the Kangaroo technique

Let x be the sample mean of the PS technique, σ be the variance, and n be size of the population. Then the test statistic z is given by:

$$z = (x - ST_{avg} \text{ for Kangaroo}) / (\sigma / \sqrt{n})$$

A 0.05 level of significance defines a critical region for z such that for any $z < 1.645$ the null hypothesis must be rejected in favor of the alternate hypothesis. For the PS technique, x = 0.677 and σ = 0.0205. Consequently,

$$z = (0.0677 - 0.665) / (0.0205 / \sqrt{20}) = 0.13$$

As $z < 1.645$, the null hypothesis H_0 must be rejected in favor of the alternate hypothesis H_1. Thus it must be concluded that, at the 0.05 level of significance, ST_{avg} for the PS technique is not different from that of the Kangaroo technique.

6.6.2 Hypothesis Test for the Semantic-PS Technique

In this section, the claim that ST_{avg} for the Semantic-PS technique is not significantly higher than that of the Kangaroo technique is tested. Once again, the null hypothesis will claim the contrary. That is, the null hypothesis states that ST_{avg} for Semantic-PS is significantly different than that of the Kangaroo technique. The null hypothesis H_0 and alternate hypothesis H_1 are stated as:

H_0: ST_{avg} for the Semantic-PS technique $\neq$ ST_{avg} for the Kangaroo technique (0.665)

H_1: ST_{avg} for the Semantic-PS technique = ST_{avg} for the Kangaroo technique

For the Semantic-PS technique, x = 0.667 and σ = 0.0137. Consequently,

$$z = (0.0667 - 0.665) / (0.0137 / \sqrt{20}) = 0.2$$

Once again, as $z < 1.645$, the null hypothesis H_0 must be rejected in favor of the alternate hypothesis H_1. Thus, it must be concluded that at the 0.05 level of significance, the ST_{avg} for the Semantic-PS technique is not different from that of the Kangaroo technique.

6.7 Evaluation of Pre-Serialization

In this section the simulation model will be used to evaluate the effectiveness of pre-serialization in minimizing the unfair treatment of mobile global transactions due to their prolonged execution time. In order to measure its effectiveness, the ideal case needs to be established.

Simply stated, the ideal case is to ensure that mobile global transactions are not penalized in any manner due to their prolonged execution. With respect to the PS and Semantic-PS techniques, this requires that the percentage of mobile global transactions aborted due to isolation property violations be equal to the percentage of static global transactions aborted due to isolation property violations. Formally, let MT-Abort be the percentage of mobile global transactions aborted during some time interval t, and ST-Abort be the percentage of static global transactions aborted during the same time interval t. Then the penalty incurred by mobile global transactions due to their extended execution time (MT-Penalty) can be represented as:

MT-Penalty = (MT-Aborts - ST-Aborts)

Note that MT-Penalty > 0 represents that mobile global transactions are penalized by the concurrency control algorithm and MT-Penalty < 0 represents that mobile global transactions are being favored by the algorithm. The ideal case is MT-Penalty = 0.

6.7.1 The Ideal Length of the Vital Stage for Global Transactions

This simulation attempts to identify the length of the vital stage of a global transactions such that the ideal MT-Penalty is obtained. For this simulation, MT-Penalty will be calculated for VT_{stg} = (0.4, 0.5, 0.6, 0.7, 0.8). The default values are used for the rest of the parameters. The simulation is carried out 10 times for each value for each technique. The results of the simulation for the PS technique are presented in Figure 6-7. The simulation results indicate that, for the PS technique, the point at which mobile transactions are not penalized for their extended execution time is when VT_{stg} is 0.5. That is, MT-Penalty is approximately 0 when the length of the vital stage spans no more than 50% of its entire length. Note that when VT_{stg} is 40% of the length of a global transaction, static transactions are being penalized by the PGSG algorithm.

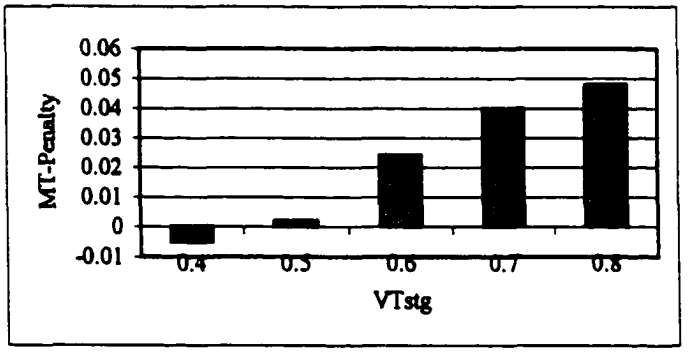


Figure 6-7: MT-Penalty Vs VT_{stg} (PS technique)

Next, the same experiment is carried out for the Semantic-PS technique. The results of this simulation are presented in Figure 6-8. Once again, the simulation results indicate that the point at which neither mobile transactions nor static transactions are penalized by the PGSG algorithm is when VT_{stg} is 0.5.

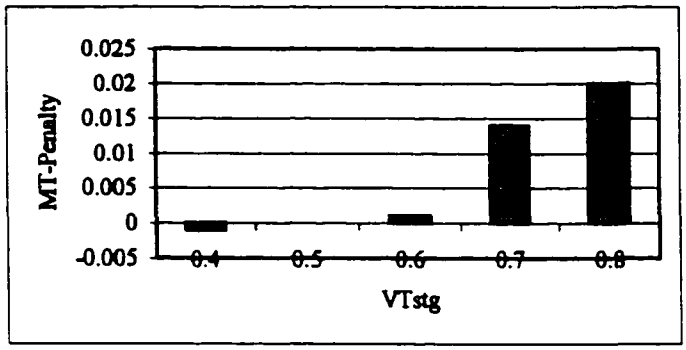


Figure 6-8: MT-Penalty Vs VT_{stg} (Semantic-PS technique)

Now that the value of VT_{stg} at which the ideal MT-Penalty is obtained has been established, the confidence interval for this value of VT_{stg} needs to be determined. The confidence interval is a level of confidence with respect to the simulation and specifies the probability that any given simulation run would produce a result within the confidence interval. This simulation will establish the 95% confidence interval. To establish this confidence interval, the simulation is carried out 20 times for each technique.

The 95% confidence interval for MT-Penalty for the PS technique when VTstg = 50% is 0.002 ± 0.0039. That is, the range for MT-Penalty that includes 95% of the simulation results for the PS technique is (-0.0019, 0.0059). The 95% confidence interval for MT-Penalty for the Semantic-PS technique when VT_{stg} = 50% is 0.000 ± 0.0015. That is, the range for MT-Penalty that includes 95% of the simulation results for the Semantic-PS technique is (-0.0015, 0.0015).

The above simulation establishes the length of the vital stage of a global transaction such that neither mobile transactions nor static global transactions are penalized by the PGSG algorithm and the 95% confidence interval for MT-Penalty. Next, simulations are carried out to determine whether environment parameters that were arbitrarily chosen would drastically affect the observed the ideal length of the vital stage, specifically GT_{int}, GT_{ratio} and P_{conf}.

6.7.2 Varying the Inter-Arrival Time

In the next experiment, the simulation is carried out for a range of global transaction inter-arrival times in order to determine its effect on VT_{stg}. Specifically, the simulation is carried out for GT_{int} = (0.2, 0.4, 0.6, 0.8, 1.0) for both techniques. Table 6-2 contains the MT-Penalty for both techniques for the different inter-arrival times. These results indicate that the inter-arrival time has no significant effect on the "fairness" of pre-serialization. Note that the results for each simulation falls within the 95% confidence interval established in the previous section.

GT_{int}	0.2	0.4	0.6	0.8	1.0
PS	0.003	0.001	0.003	0.002	0.001
Semantic-PS	0.00015	0.0004	0.001	0.0005	0.000

Table 6-1: MT-Penalty for varying inter-arrival times

6.7.3 Varying the Mobile to Static Transaction Ratio

In the next experiment, the simulation is carried out for different values of GT_{ratio} - the ratio of mobile global transactions to static global transactions in order to determine its effect on VT_{stg}. Specifically, the simulation is carried out for GT_{ratio} = (0.3/0.7, 0.4/0.6, 0.5/0.5, 0.6/0.4, 0.7/0.3) for both techniques. Table 6-3 contains the MT-Penalty for both techniques for the different inter-arrival times. Once again, the results fall within the 95% confidence interval obtained in the initial simulation, indicating that the ratio of mobile/static global transactions has no significant effect on the "fairness" of pre-serialization.

GT_{ratio}	0.3/0.7	0.4/0.6	0.5/0.5	0.6/0.4	0.7/0.3
PS	0.003	0.002	0.002	0.004	0.004
Semantic-PS	0.000	0.0001	0.001	0.000	0.000

Table 6-3: MT-Penalty for varying Mobile/Static ratios

6.7.4 Varying the Probability of Conflicts

In the next experiment, the simulation is carried out for different values of P_{cnf} - the probability of a site-transaction conflicting with another site-transaction at the same site. Specifically, the simulation is carried out for P_{cnf} = (0.1, 0.2, 0.3, 0.4, 0.5). As P_{cnf} has no influence on the execution of the PS technique, this experiment is carried out only for the Semantic-PS technique. Table 6-4 contains the MT-Penalty obtained from this experiment. For P_{cnf} = 0.1 this simulation produced no aborts of global transactions. For P_{cnf} = 0.2, 0.3, 0.4, 0.5, MT-Penalty falls within the 95% confidence interval indicating that VT_{sg} is not affected by P_{cnf}.

P_{cnf}	0.1	0.2	0.3	0.4	0.5
Semantic-PS	0.000	0.002	0.001	0.000	0.001

Table 6-3: MT-Penalty for varying Mobile/Static ratios

6.7.5 Varying the Disconnection Time

In the next experiment, the simulation is carried out for different values of DCN_{tm} - the average time of disconnection. Specifically, the simulation is carried out for DCN_{tm} = (0.2, 0.4, 0.8, 1.0, 1.2). These results indicate that the average time between disconnection and relocation affects MT-Penalty (Figure 6-9). As the average disconnection time increases, MT-Penalty increases as well.

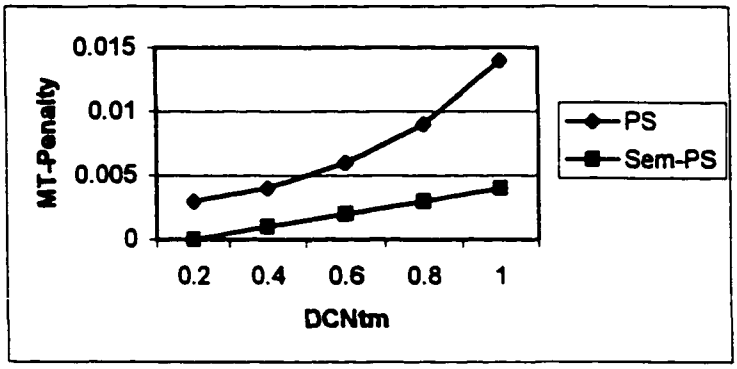


Figure 6-9: MT-Penalty for varying DCN_{tm}

6.7.6 Varying the Wireless Communication Time

In the next experiment, the simulation is carried out for different values of T^{w}_{msg} - the time taken to transmit a message over the wireless network. Specifically, the simulation is carried out for T^{w}_{msg} = (0.1, 0.2, 0.3, 1.4, 1.5). These results indicate that the time taken to transmit a message over the wireless communication network affects MT-Penalty (Figure 6-10). As T^{w}_{msg} increases, MT-Penalty increases as well.

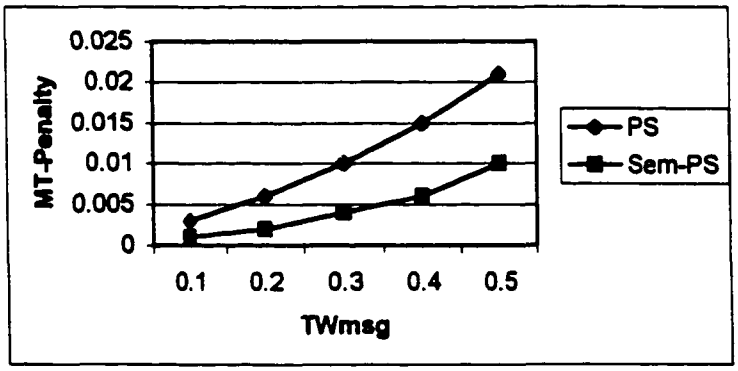


Figure 6-10: MT-Penalty for varying T^{w}_{msg}

6.8 Summary and Conclusion

The simulations carried out in this section establishes that pre-serialization can effectively reduce the unfair treatment of mobile global transactions due to their prolonged

execution. In fact, it was shown that the PS and Semantic-PS techniques can effectively eliminate all unfair treatment of mobile global transactions when the vital stage of a global transaction is 50% of the total length of the global transaction. It was also shown that the inter-arrival time, the ratio of mobile to static global transactions, and the probability of conflicts do not have any drastic effect on the effectiveness of pre-serialization for the default simulation environment. This is to be expected as the execution of mobile global transactions is identical to that of static global transactions in both techniques except for the time taken to complete the execution. The parameters GT_{int}, GT_{ratio} and P_{cnf} do not affect the execution time of mobile transactions. Therefore, they have no effect on MT-Penalty.

However, the simulation showed that DCN_{tm} and T^{w}_{msg} affect MT-Penalty. This can be explained as follows. Pre-serialization is introduced to address the unfair treatment of mobile transactions due to their prolonged executions. In effect, pre-serialization reduces the time period within which a mobile global transaction can cause conflict violations, i.e., VT_{stg}. The ideal MT-Penalty is achieved at some ratio (between the time taken to complete the execution of static global transactions and the VT_{stg} of mobile transactions) at which the concurrency control algorithm resolves conflict violations by aborting the same percentage of mobile and static global transactions. Unlike GT_{int} GT_{ratio} and P_{cnf}, DCN_{tm} and T^{w}_{msg} alter the interval VT_{stg} and therefore, change the point at which the ideal MT-Penalty occurs. In fact, it can be concluded that any parameter that changes the ratio between the time taken to complete the execution of static global transactions and the VT_{stg} of mobile transactions will affect the point at which MT-Penalty is zero.

Chapter 7

CONCLUSION AND FUTURE RESEARCH

Current advances in technology has changed the conventional computing environment. On the one hand the Internet has revolutionized connectivity and introduced the notion of cooperating but autonomous information systems. On the other hand rapid advances in wireless communication technology has introduced the notion of mobile computing. This dissertation research studies database transaction management in the mobile multidatabase environment.

The major contributions of this research are fourfold. First, this dissertation research studies the issues related to transaction management in the MMDB environment. Second, two new transaction management techniques that address all identified issues are proposed. These techniques introduce the following new concepts to transaction management:

1. It introduces two new states - Disconnected and Suspended - in order to address disconnectivity of the wireless communication medium.
2. It introduces the notion of pre-serialization to address the prolonged execution of mobile global transactions.
3. It introduces a new concurrency control algorithm based on partial global serialization graphs and information propagation.
4. An analytical model of the MMDB environment is developed. Simulation models of the proposed transaction management techniques are also developed. These models are used to evaluate the performance of the proposed techniques.

7.1 Transaction Management in the MMDB Environment

Transaction management is a core concept in the science of Database Management and has been studied extensively in traditional database environments. Transactions are defined to be consistent and reliable units of computing. In traditional systems, transactions satisfy this definition if they adhere to the ACID properties. However, the cooperating federated computing environment and the mobile computing environment introduce new issues that affect database transaction management. The multidatabase environment requires cooperating database systems to provide a single logical view of the information resources to the user without violating the autonomy of the constituent database systems. The mobile database environment requires that information available on the static network to wired users be made available to mobile users who connect from anywhere at any time. Wireless communications are frequently interrupted by disconnection and migration. These disconnection and migration violate underlying presumptions about user connectivity that exists in wired systems. The natures of these new environments have raised legitimate questions about the applicability of the ACID properties. Therefore, it is necessary to revisit the responsibilities of the global transaction management process in light of the new environments.

The GTM of the MMDBS is responsible for enforcing the reliability and consistency of global transactions. Unlike in traditional DBMSs, the GTM of the MMDBS does not have to enforce all the ACID properties for two reasons. As the constituent databases enforce the ACID properties on the site-transactions that execute under their control, global transactions, by default, satisfy the consistency and durability properties. Therefore, the GTM is responsible for only the atomicity and isolation properties. However, the requirements of the new environments dictate that the GTM enforces a range of correctness criteria with respect to the atomicity and isolation properties, ranging from strict A/I to unrestricted access.

In addition to the A/I properties, the GTM needs to address disconnection and migrating transactions. Unlike in the static environment, disconnection in the mobile environment cannot always be treated as failures that result in aborted transactions. In some cases, however, disconnection will be caused by a catastrophic failure. As the MMDBS can only predict catastrophic failures, aborting disconnected transactions is likely to result in some untimely terminations. The GTM needs to take appropriate steps to minimize such untimely terminations. Disconnection and migration of the mobile user prolong the execution time of mobile transactions as well. Consequently, this affects the enforcement of the isolation property. In order to maintain a notion of fairness, the concurrency control mechanism of the GTM must minimize any unfair treatment of transactions of mobile users.

Furthermore, the GTM must also conform to multidatabase design restrictions, i.e., the autonomy of the LDBSs cannot be violated.

7.2 The PS and Semantic-PS techniques

This dissertation proposes two transaction management techniques, PS and PS-Semantic, for the MMDB environment based on the Multi-Level transaction model. In this transaction model, global transactions consist of a set of site-transactions such that each site-transaction is executed as a single (consistent and reliable) transaction at some local DBMS. Each site-transaction is categorized as vital or non-vital. The time between the first and last vital site-transaction of a global transaction constitute its vital phase. All vital site-transactions of a global transaction must complete successfully in order for the global transaction to complete its execution successfully. As a global transaction can consist of any combination of vital and non-vital site transactions, these techniques can enforce a range of correctness criteria with respect to the A/I properties.

The GTM of each technique consists of two layers: the Global Coordinator layer is responsible for the overall execution and coordination of global transactions, and the Site Manager layer is responsible for managing the execution of site-transactions at each site. This research introduces two new states - Disconnected and Suspended - in order to address the disconnectivity of wireless communications. Global transactions of a disconnected user are placed in the Disconnected state until the user re-connects (at which time the global transaction is set back to active), or until such time that the MMDBS determines that a catastrophic failure has occurred. In the later case, the global transaction is placed in a Suspended state. In an effort to minimize untimely aborts caused by erroneous decisions about the users' connectivity status, Suspended transactions are not aborted until they interfere with the execution of other global transactions.

The PGSG algorithm verifies the atomicity and isolation properties of global transactions. The PGSG algorithm verifies the isolation property by constructing a partial global serialization graph and relies on information propagation to ensure that all violations are detected. The primary difference between the PS technique and the Semantic-PS technique lies in the enforcement policy of the A/I properties.

In the PS technique, static global transactions execute the PGSG algorithm at the end of their execution in order to verify the A/I properties. If A/I have not been violated the transactions commits; otherwise the transactions are aborted. Mobile global transactions execute the PGSG algorithm at the end of their vital stage. If A/I has not been violated, the global transactions are toggled; otherwise they are aborted. The toggle operation registers the global transactions' serialization order in the global transaction serialization scheme. Upon completing their execution, toggled mobile global transactions execute the PGSG algorithm a second time in order to rectify any isolation property violations that they may have caused after being toggled. In the PS technique, the PGSG algorithm enforces A/I on all site-transactions of a

global transaction. However, only vital site-transactions can cause a global transaction to be aborted. As the local serialization of site-transactions is transparent to the MMDBS, the PS technique forces conflicts between all site-transactions that execute at each site using a local ticket data item. The ticket value is then used to deduce the local serialization order.

The limitations of the PS technique are:

1. Mobile transactions utilize additional overhead as they execute the PGSG algorithm twice.
2. Concurrency is limited as all site-transactions that execute at each site are forced to conflict with each other.

The Semantic-PS technique overcomes these limitations as follows. The Semantic-PS technique enforces the A/I properties only on the set of vital site-transactions of a global transaction. As in the PS technique, the static global transactions execute the PGSG algorithm at the end of their execution and mobile global transactions execute the PGSG algorithm at the end of their vital stage. However unlike in the PS technique, mobile global transactions are not required to execute the PGSG algorithm a second time as the Semantic-PS technique enforces A/I only on the set of vital site-transactions. Toggled mobile global transactions are allowed to commit at the end of their execution. In order to improve concurrency, the Semantic-PS technique relies on semantic information about local data items to combine local operations (executed by site-transactions) into groups that potentially conflict with each other and assign a ticket to each group. Thus, the Semantic-PS technique enforces conflicts only between site-transactions that execute one or more operations from the same group.

As the PS and Semantic-PS techniques allow mobile global transactions to establish their serialization order prior to completing their execution, these techniques minimize the unfair treatment of mobile global transactions due to their prolonged execution.

7.3 Feature Comparison with Existing Techniques

The MMDB environment is a relatively new area of research which encompasses emerging technologies. Yet, it has received considerable attention from the research community. Many existing publications specifically address transaction management in the MMDB environment. However, the proposed techniques fall short of meeting all the requirements of the MMDB environment. To summarize their deficiencies, none of the techniques enforces the (global) isolation property. Thus, global transactions are not executed as consistent units of computing. In addition, disconnections that represent catastrophic failures are not addressed. It is assumed that a disconnection will always be followed by a subsequent re-connection. The PS and Semantic-PS techniques are compared to the existing techniques in Table-7-1.

Technique	Disctn. Support	Migrtn. Support	Autonomy Violated	Atomicity Level	Isolation Level
Agent-Based Access [PB95-2]	Partial	Partial	No	VAR	None
TP in Mobile Env [Chry93]	Partial	Full	Yes	VAR	None
MDSTMP [YZ94]	Partial	Full	No	STR	None
Kangaroo Model [DHB97]	Partial	Partial	No	VAR	None
PS Technique	Full	Full	No	VAR	VAR
Semantic-PS technique	Full	Full	No	VAR	VAR

Table 7-1: Summary of Mobile Multidatabase Transaction Models

7.4 Performance Analysis and Simulation

In this dissertation research, an analytical model of transaction management in an MMDB environment is developed in Chapter 5. This model is used to study the transaction service time of the PS and Semantic-PS techniques and to compare their performance to that of the Kangaroo model. A simulation model is developed in Chapter 6 and is also used to study the

performance of the three techniques. This simulation model is then used to study the behavior of the PS and Semantic-PS techniques.

The PGSG algorithm of the PS and Semantic-PS techniques requires serialization information to be transmitted on the static network in order to construct the partial global serialization graph and to propagate serialization information back to the participating sites. The primary goal of the analytical evaluation was to determine the cost (per global transaction) of this information propagation with respect to the total execution time of the global transaction. The initial analysis with respect to global transaction length, number of disconnections, and number of migrations indicated that the transaction service time and the rate of growth of the service time for the PS and Semantic-PS techniques are comparable to those of the Kangaroo technique. When the service time is evaluated with respect to the communication cost, the PS and Semantic-PS still remain comparable to the Kangaroo model. However, it is evident that the service time of the PS technique deteriorates more rapidly than the Semantic-PS and Kangaroo technique. This can be attributed to the following:

1. The PS technique executes the PGSG algorithm twice for all mobile global transactions.
2. All site-transactions that execute at the same site are forced to conflict with each other.

The primary goal of the simulation was to determine the length of the vital stage of global transactions such that mobile global transactions are not penalized for their extended execution time. The simulation model was also used to study the behavior of the PS and Semantic-PS techniques with respect to related environment parameters. The simulation results indicate that the length of the vital stage such that mobile global transactions are not penalized (i.e., MT-Penalty = 0) is 50% of the total length of the global transaction for both the PS and Semantic-PS technique. The 95% confidence interval for MT-Penalty when the vital stage is 50% of the total length of the global transaction is (-0.0019, 0.0059) for the PS technique and (-0.0015, 0.0015) for the Semantic-PS technique. The simulation demonstrated that the inter-

arrival time of global transactions, the ratio of static to mobile transactions, and the probability of conflicts do not alter the ideal MT-Penalty in any significant way. However, it is evident that MT-Penalty is affected by DCN_{m} and T^{w}_{msg}. This is due to the fact that these parameters affect the length of the vital stage of mobile transactions.

In summary, the analytical model demonstrates that the communication cost incurred by the respective PGSG algorithm of the PS and Semantic-PS techniques accounts for only a small portion of the service time of the global transaction in environments where the communication cost on the static network is relatively small. Thus, the PS and Semantic-PS techniques offer a substantial advantage over existing techniques – they enforce the isolation property without violating local DBMS autonomy – for little additional communication overhead. The simulation demonstrates that pre-serialization is an effective technique that can be used to minimize the unfair treatment of mobile global transactions due to their prolonged execution time.

7.5 Future Research

This research deals with transaction management in two rapidly changing computing environments, i.e., the vastly expanding Internet environment and the wireless computing environment. Like any technology in its early stages, the Internet and wireless computing environments are driven by innovations. In essence, these environments can be characterized as revolutionary environments rather than evolutionary environments. The volatile nature of these rapidly changing environments makes it difficult for researchers to design evaluate and optimize algorithms for the environments. It also makes it necessary for researchers to continuously re-evaluate design criteria and to either alter proposed solutions to meet the needs of the changed environments or propose new solutions. Accordingly, the future research of this dissertation will proceed in three directions: the requirements of the transaction manager need to be re-evaluated

with respect to emerging computing models; the PS and Semantic-PS techniques will be tailored for the cellular communication architecture; these techniques need to be re-evaluated for the new environment.

7.5.1 Emerging Computing Models

In today's Internet computing environment, electronic commerce (a.k.a., e-commerce) has become a predominant application domain. Just as mobile computing has affected transaction processing, e-commerce is beginning to influence transaction processing in the multidatabase environment. The e-commerce transaction processing is called Internet Transaction Processing (iTP). Although the transaction model is not yet fully understood, it is expected that this model will be different from traditional online transaction processing (OLAP). As iTP evolves, the PS and Semantic-PS techniques need to be tailored to satisfy the new requirements as iTP will be the predominant heterogeneous transaction processing environment.

7.5.2 Cellular Communication Architecture

Current trends indicate that the cellular communication medium will be the predominant communication medium of mobile computing applications. Thus, it is reasonable to optimize the PS and Semantic-PS techniques for the cellular communication architecture. Although there are many cellular networks (e.g., Sprint PCS and AT&T Digital Networks), their underlying architecture is very similar. The cellular network consists of cells – regions that are covered by a node that supports wireless communication. Cells are grouped to form clusters and in turn, these clusters are grouped to form a hierarchy of clusters that represents a tree of clusters.

In order to minimize the overhead of information propagation, the research needs to study a hybrid concurrency control algorithm that is based on information propagation (such as in the PGSG algorithm) between clusters and some concurrency control algorithm that does not require information propagation within a cluster of cells. The motivation of this algorithm is that it would eliminate the need for information propagation within a cluster of cells and yet provide all the advantages of the PGSG algorithm with respect to the global environment (i.e., the cluster tree). In essence, each cluster is treated as a site in the current environment. Information will need to be propagated only when transactions access sites covered by different clusters.

BIBLIOGRAPHY

[Adam94] N. R. Adam. "A New Dynamic Voting Algorithm for Distributed Database Systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 3, June 1994.

[ABI94] A. Acharya, B. R. Badrinath, T. Imielinski. "Checkpointing Distributed Applications on Mobile Computing", Proc. of the Third International Conference on Parallel and Distributed Systems, September 1994.

[AK93] R. Alonso, H. F. Korth. "Database System Issues in Nomadic Computing", SIGMOD Record, May 1993.

[BAI94] B. R. Badrinath, A. Acharya, T. Imielinski. "Structuring Distributed Algorithms for Mobile Hosts", Proc. Fourth International Conference on Distributed Computing Systems, May 1994.

[Bern et.al98] P. Bernstein, M. Brodie, S. Ceri, D. DeWitt, M. Franklin, H. Garcia-Molina, J. Gray, J. Held, J. Hellerstein, H. Jagadish, M. Lesk, D. Maier, J. Naughton, H. Pirahesh, M. Stonebraker, J. Ullman. "The Asilomar Report on Database Research". SIGMOD Record Vol. 27, No. 4 December 1998.

[BHG87] P. A. Bernstein, V. Hadzilacos, N. Goodman. "Concurrency Control and Recovery in Database Systems". Addison-Wesley Publishing Company, 1987.

[BI94] D. Barbara, T. Imielinski. "Sleepers and Workaholics: Caching Strategies in Mobile Environments", ACM SIGMOD, Vol. 23, No 2, June 1995.

[BMS92] Y. Breitbart H. Garcia-Molina, A. Silberschatz. "Overview of Multidatabase Transaction Management" , TR-92-21, University of Texas at Austin.

[BR92] S. Ben-Hassen, M. Rusinkiewicz. "On Serializability of Distributed Nested Transactions", Proc. 12th International Conference on Distributed Computing Systems, Japan 1992.

[BS88] Y. Breitbart, A. Silberschatz. "Multidatabase Update Issues", Proc. ACM SIGMOD Conference on Management of Data, Chicago 1988.

[Chry93] P. K. Chrysanthis. "Transaction Processing in Mobile Computing Environments", IEEE Workshop on Advances in Parallel and Distributed Systems, October 1993.

[DE89] W. Du, A. K. Elmagarmid. "Quasi Serializability: A Correctness Criterion for Global Concurrency Control in InterBase". Proceedings of the 15th International Conference on VLDB, Amsterdam, The Netherlands, August 1989

[DG98] R. Dirckze, L. Gruenwald. "Nomadic Transaction Management - Examining the Affects of Mobility upon the ACID Properties" IEEE Potentials , April 1998.

[DG98-2] R. Dirckze, L. Gruenwald. "Disconnection and Migration in Mobile Multidatabases", The World Conf. on Design and Process Technology, Germany, July 1998

[DG98-3] R. Dirckze, L. Gruenwald. "A Toggle Transaction Management Technique for Mobile Multidatabase", The 7th International Conference on Information and Knowledge Management, Washington DC, November 1998.

[DG2000] R. Dirckze, L. Gruenwald. "A Pre-serialization Transaction Management Technique for Mobile Multi-databases" To appear: Special issue on Software Architectures for Mobile Applications, MONET 2000.

[DH95] M. H. Dunham, A. Helal. "Mobile Computing and Databases: Anything New?", SIGMOD Record, Vol. 24, No. 4, December 1995.

[DHB97] M. Dunham, A. Helal, S. Balakrishnan. "A Mobile Transaction Model that Captures Both the Data and Movement Behavior ". Mobile Networks and Applications, Vol. 2, No. 2, October 1997.

[DI99] R. Dirckze, S. Iyengar. "Metadata Repositories and Mobile Computing". 3rd World Multiconference on Systemics, Cybernetics and Informatics and 5th Int. Conf. on Information Systems Analysis and Synthesis. Orlando FL, August 1999.

[DSW92] A Deacon, H. Schek, G. Weikum. "Semantic-based Multilevel Transaction Management in Federated Systems", Proc. 10th International Conference on Data Engineering, Texas, 1992

[EJB95] A. Elmagarmid, J. Jing, O. Bukhres. "An Efficient and Reliable Reservation Algorithm for Mobile Transactions", Proceedings of the 4th International Conference on Information and Knowledge Management, 1995.

[EJB95-2] A. Elmagarmid, J. Jing, O. Bukhres. "Distributed Lock Management for Mobile Transactions", Proc. of the 15th International Conference on Distributed Computing Systems, Canada, June 1995.

[ERS98] A. Elmargarmid, M. Rusinkiewicz, A. Seth. "Management of Heterogeneous and Autonomous Database Systems", Morgan Kaufmann Publishers, Inc. San Francisco CA, 1998.

[GR93] J. Gray, A. Reuter. "Transaction Processing: Concepts and Techniques". Morgan Kaufmann Publishers, Inc. 1993.

[GRS91] D. Georgakapolous, M. Rusinkeiwicz, and A. Sheth. "On Serializability of Multidatabase Transactions through Forced Local Conflicts", Proceedings of the 7th International Conference on Data Engineering, Kobe, Japan, 1991.

[HE95] A. Helal, M. Eich. "Supporting Mobile Transaction Processing in Databasse Systems", Technical Report: TR-CSE-95-003, University of Texas at Arlington, April 1995.

[IB92] T. Imielinski, R. B. Badrinath. "Querying in Highly Mobile Distributed Environments" Proc. Eighteenth VLDB Conference, Vancouver, Canada, 1992.

[IB94] T. Imielinski, R. B. Badrinath. " Wireless Computing: Challenges in Data Management", Communications of the ACM, Vol. 37, No. 10, October 1994.

[JNRS93] W. W. Jin, L. Ness, M. Rusinkiewicz, A Sheth. "Concurrency Control and Recovery of Multidatabase Work Flows in Telecommunication Applications", ACM SIGMOD, Vol. 22, No. 2 June 1993.

[KS93] P. Kumar, M. Satyanarayanan. "Log-Based Directory Resolution in the Coda File System", Proc. Second International Conference on Parallel and Distributed Information Systems, 1993.

[KSS98] Kelton, W. D., Sadowski, R. P., Sadowski, D. A. "Simulation with Arena", WCB/McGraw-Hill Publishers 1998.

[LKS91] E. Levy, H. F. Korth, A. Silberschatz. "An Optimistic Commit Protocol for Distributed Transaction Management". Proceedings of ACM-SGMOD International Conference on Management of Data, Colorado, May 1991.

[MB98] S. K. Madria, B. K. Bhargava. . "A Transaction Model for Mobile Computing". IDEAS, 1998

[MMP83] E. T. Mueller, J. D. Moore, G. J. Popek. "A Nested Transaction Mechanism for Locus". Proc. 9th Symposium on Operating Systems Principals, ACM/SIGOPS, 1983

[Moss81] J. E. B. Moss. "Nested Transactions: An Approach to Reliable Computing". MIT, LCS-TR-260, 1981.

[MRSK92] S. Mehrotra, R. Rastogi, A. Silberschatz, H. F. Korth. "A Transaction Model for Multidatabase Systems", Proc. 12th International Conference on Distributed Computing Systems, Japan 1992

[NS95] B. D. Noble, M. Satyanarayanan. "A Research Status Report on Adaptation for Mobile Data Access", SIGMOD Record, Vol. 24, No. 4, December 1995.

[OV91] M. T. Ozsu, P Valduriez. "Principles of Distributed Database Systems" Prentice Hall, Englewood Califf, N.J. 1991.

[PB93] E. Pitoura, B. Bhargava. "Dealing with Mobility: Issues and Research Challenges". Technical Report CSD-TR-93-070, Purdue University 1993.

[PB94] E. Pitoura, B. Bhargava. "Revising Transaction Concepts for Mobile Computing", Proceedings of the IEEE Workshop on Mobile Systems and Applications, Santa Cruz, CA, December 1994.

[PB95] E. Pitoura, and B Bhargava. "Maintaining Consistency of Data in Mobile Distributed Environments". 15th International Conference on Distributed Computing Systems, Canada, June 1995.

[PB95-2] E. Pitoura, and B Bhargava. "A Framework for Providing Consistent and Recoverable Agent-Based Access to Heterogeneous Mobile Databases". SIGMOD Record, September 1995.

[TPC99] Transaction Processing Performance Council. www.tpc.org

[Ullm88] J. D. Ullman. "Principles of Database and Knowledge-Base Systems". Computer Science Press, Inc.

[SS93] W. Schaad, H. Schek. "Federated Transaction Management using Open Nested Transactions", Proc. Workshop on Interoperability of Database Systems and Database Applications, Switzerland, 1993.

[WC94] G. D. Walborn and P. K. Chrysanthis. "Supporting Semantic-Based Transaction Processing in Mobile Database Applications", 14th IEEE Symposium on Reliable Distributed Systems, September 1994.

[WM85] R. E. Walpole, R. H. Mayers. "Probability and Statistics for Engineers and Scientists". Macmillian Publishing Company, New York, 1985.

[WSDNR95] O. Wolfson, P. Sistla, S. Dao, K. Narayanan, R. Raj. "View Maintenance in Mobile Computing". ACM SIGMOD Vol. 24, No. 4, December 1995.

[YZ94] L. H. Yeo, A. Zaslavsky. "Submission of Transactions from Mobile Workstations in a Cooperative MDB Processing Environment". 14th International Conference on Distributed Computing Systems, Poland, June 1994.

[ZE93] A. Zhang, A. K. Elmagarmid. "A Theory of Global Concurrency Control in Multidatabase Systems", The VLDB Journal, Vol. 2, No. 3, July 1993.

[ZNBB94] A. Zhang, M. Nodine, B. Bhargava, O. Bukhres. "Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems", ACM SIGMOD Vol./ 23, No. 2, June 1994.