

PERPUSTAKAAN UMP



0000044503

READ-O

G SYNCHRONIZATION

TRANSACATION SYSTEM (KOWA-MSTS)

TITO LAMAT ANAK JAYAN

**A report submitted in partial fulfilment
of the requirements for the award
of the degree of
Bachelor of Computer Science (Software Engineering)**

**Faculty of Computer Systems & Software Engineering
Universiti Malaysia Pahang**

NOVEMBER 2009

PERPUSTAKAAN UNIVERSITI MALAYSIA PAHANG	
No. Perolehan	No. Panggilan
044503	QA 76.9
Tarikh	D3 T58 2009 rs
12 MAR 2010	Bc.

ABSTRACT

In the world where internet does all the business nowadays, the demand of transmitted information over networks increase rapidly and the demand for steady bandwidth seems to be out of control. Particularly organizations need to provide updated data to users that might be geographically remote and handling a vast amount of requested data distributed in multiple sites. In distributed system environment, replicated data is the current trend to keep data updated in multiple sites. Replication in distributed environment has become increasingly popular due to its high degree of availability, fault tolerance and enhance the performance of a system. These advantages of replication are important because it enables organizations to provide users with access to current data anytime or anywhere even if the users are geographically remote. At certain time, access to the current data can be made anywhere in distributed systems. However, this way of data organization also introduces low data consistency among replicas when changes are made during transactions. The need to have a system to monitor this data replication arises. Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS) is developed to answer the issue of problem made by the ROWA replication technique. Rapid Application Development (RAD) is the software life cycle used to develop this project due to short period of development time. This project will help to monitor the replicated data distribution over multiple sites with a better performance.

ABSTRAK

Anjakan permintaan perkhidmatan jalur lebar yang melambung dalam menguruskan kehidupan sehari-harian telah menyebabkan permintaan untuk memiliki *bandwidth* yang mantap juga telah meningkat. Organisasi khususnya, mesti memastikan setiap data berada dalam keadaan yang sentiasa dikemaskini walaupun menghadapi kekangan seperti faktor geografi yang terpinggir. Di dalam persekitaran *Distributed System*, data replikasi ialah trend terbaru untuk memastikan data dikemaskini di setiap rangkaian. Pendekatan replikasi data menjadi semakin mendapat perhatian kerana ciri-cirinya seperti darjah kehadiran data yang tinggi, mesra kesilapan dan memantapkan persembahan sesuatu sistem. Terdapat keadaan di mana data yang sama diakses serentak oleh beberapa pengakses di beberapa lokasi. Keadaan ini menyebabkan data kurang konsisten apabila terdapat kemaskini terhadap data dibuat. Maka, timbullah satu keperluan dalam organisasi untuk mempunyai satu sistem yang boleh mengawal dan menyelia data replikasi ini. Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS) dibangunkan atas dasar untuk menyelesaikan kekangan yang berlaku sekiranya timbul masalah semasa data replikasi berlaku. Rapid Application Development (RAD) pula ialah pendekatan yang dipilih untuk membangunkan projek ini memandangkan cirinya yang sesuai untuk pembangunan sistem dalam masa yang singkat. Projek ini akan mengawal dan menyelia data replikasi dengan baik.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	TITLE PAGE	i
	DECLARATION	ii
	SUPERVISOR'S DECLARATION	iii
	DEDICATION	iv
	ACKNOWLEDGEMENT	v
	ABSTRACT	vi
	ABSTRAK	vii
	TABLE OF CONTENTS	viii
	LIST OF TABLES	xi
	LIST OF FIGURES	xii
	LIST OF ABBREVIATIONS	xiii
	LIST OF APPENDICES	xiv
1	INTRODUCTION	
	1.0 Introduction	1
	1.0 Problem Statement	3
	1.1.1 Case Study	4
	1.1 Project Objective	5
	1.2 Project Scope	6
	1.3 Thesis Organization	6
2	LITERATURE RIVIEW	
	2.0 Introduction	7
	2.1 Replication Technique	8
	2.1.1 Read-One-Write-All (ROWA)	8

2.1.2 Neighbor Replication on Grid (NRG)	14
2.1.3 Two Phase Commit (2PC)	14
2.1.4 CORel	15
2.1.5 Paxos	15
2.2 Existing Application Related to ROWA-MSTS	15
2.2.1 Mobile P2P environment	16
2.2.2 Postgres	17
2.3 ROWA-MSTS	19
2.3.1 Data Replication Strategies in Grid Environment	19
2.4 Software Approach for the development of ROWA-MSTS	20
2.4.1 Operating System	20
2.4.1.1 Operating System Security	21
2.4.1.2 Stability	22
2.4.1.3 Environmental Friendly	22
2.4.1.4 Update Availability	23
2.4.1.5 Freedom	24
2.4.1.6 Version of Linux	25
2.4.1.6.1 Debian-based	25
2.4.1.6.2 Knoppix-based	29
2.4.1.6.3 Ubuntu-based	30
2.4.1.6.3.1 Official Distributions	30
2.4.1.6.3.2 Unofficial Distributions	32
2.4.1.6.4 Gentoo-Based	37
2.4.1.6.5 RPM-Based	38
2.4.1.6.6 Fedora Based	40
2.4.1.6.7 RedHatLinux Linux-Based	42
2.4.1.6.8 Mandriva Linux-Based	43
2.4.1.6.9 Slackware Based	44

	2.4.1.6.10 SLAX-Based	44
	2.4.1.6.11 Others	45
2.5	Programming Language	49
2.5.1	Bash Programming	49
2.6	Server	50
2.7	Development Tools	51
2.7.1	File Transfer Protocol	51
2.7.1.2	Types of FTP	51
2.7.1.2.1	Active FTP	52
2.7.1.2.2	Passive FTP	53
2.7.1.2.3	Regular FTP	54
2.7.1.2.4	Anonymous FTP	54
2.7.1.3	Types of FTP Software	55
2.7.1.3.1	VSFTPD	55
2.7.1.3.2	PURE_FTPD	55
2.7.1.3.4	PRO FTP	56
2.7.1.4	Vi Editor	55
2.7.1.5	Terminal	57
2.8	General Hardware and Software Requirement	57
2.8.1	Hardware Requirement	58
2.8.2	Software Requirement	59
3	METHODOLOGY	
3.1	Introduction	61
3.2	Implement of ROWA method in ROWA-MSTS	61
3.3	Framework of ROWA-MSTS in Dist. Transaction Semantic	62
3.4	Proposed Algorithm for ROWA-MSTS	63
3.4.1	Flowchart for Algorithm of ROWA-MSTS	63
3.5	Hardware and Software tool	67
3.5.1	Hardware	67
3.5.2	Software	68

	3.5.3 Additional Hardware	68
4	IMPLEMENTATION	69
	4.1 Introduction	69
	4.2 Set up FTP Server	69
	4.2.1 Install VSFTPD Server	70
	4.2.2 VSFTPD Configuration	71
	4.3 Bash Script	72
5.	RESULT AND DISCUSSION	76
	5.1 Introduction	76
	5.2 Testing	78
	5.2.1 Test Connection	78
	5.2.2 Test the FTP Server	80
	5.3 Result	80
	5.3.1 Initiate Lock	82
	5.3.2 Propagate lock	83
	5.3.2 Obtain Lock	84
	5.3.4 Update lock	85
	5.3.5 Commit	86
	5.3.6 Unlock	87
	5.3.7 Release locks	88
	5.4 Discussion	89
	5.5 Project Limitation	91
	5.5.1 Development constraints	92
	5.5.2 Technical Knowledge	92
	5.5.3 Time Constraint	92
	5.5.4 System Constraint	93
	5.10 Future Enhancement	93
6.	CONCLUSION	94
	REFERENCES	96
	Appendix A	99

LIST OF TABLE

TABLE NO.	TITLE	PAGE
Table 2.1	Debian Based Distribution	26
Table 2.2	Knoppix-based Distribution	29
Table 2.3	Ubuntu-based Official Distribution	30
Table 2.4	Ubuntu-based Unofficial Distribution	33
Table 2.5	Gentoo-based Distribution	38
Table 2.6	RPM-based Distribution	39
Table 2.7	Fedora-based Distribution	41
Table 2.8	Red Hat Enterprise Linux-based	42
Table 2.9	Mandriva Linux-based	44
Table 2.10	SLAX-based	45
Table 2.11	Others	45
Table 2.12	Hardware Specifications	58
Table 2.13	Software Requirement	59
Table 3.1	Hardware Tools	67
Table 3.2	Software	68
Table 3.3	Additional Hardware	68
Table 5.0	Primary-Neighbor Coordination	81
Table 5.1	Status Lock for ROWA-MSTS	81
Table 5.2	Experiment result ROWA-MSTS	89
Table 5.3	Average time taken for each lock to complete	90

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
Figure 1.0	Replicated data stored in different servers	4
Figure 1.1	T _a request to update X=2 at server B	5
Figure 2.0	Sites contains the replica of data object	9
Figure 2.1	Initial value of data is U	10
Figure 2.2	Transaction T _i , is submitted to site S _i	10
Figure 2.3	Site S _i sends messages to participating sites	11
Figure 2.4	Sub transaction of T _i starts at participating sites	11
Figure 2.5	Replication Technique	12
Figure 2.6	DT T _i aborts if any of sub transaction aborts	13
Figure 2.7	None of the replica is updated	13
Figure 2.8	All replica updated	13
Figure 2.9	Transparent Postgres Replication Architectures	17
Figure 2.10	Hybrid Replica Connection Graph	19
Figure 2.11	Update Manager	24
Figure 2.12	Active and Passive FTP Illustrated	52
Figure 2.13	Terminals in Ubuntu Linux.	57
Figure 3.1	Locking Phase	62
Figure 3.2	ROWA-MSTS between 2 Servers	62
Figure 3.3	Framework of ROWA-MSTS	63
Figure 3.4	ROWA-MSTS Distribution Transaction Semantic	63
Figure 3.5	Flowchart of ROWA-MSTS	63
Figure 3.6	Transaction semantic in ROWA-MSTS	65
Figure 4.0	VSFTPD Server Configuration	70
Figure 4.1	VSFTPD Server Configuration	70
Figure 4.2	gnome-terminal	71

Figure 4.3	VSFTPD Configuration File	71
Figure 4.4	ROWA-MSTS Script	72
Figure 4.5	ROWA-MSTS Script	73
Figure 4.6	ROWA-MSTS Script	74
Figure 4.7	ROWA-MSTS Script	75
Figure 5.0	Configure the IP Address for Master Server.	78
Figure 5.1	Ping 172.21.140.137 success.	79
Figure 5.1	Ping 172.21.140.137 success.	79
Figure 5.2	Ping 172.21.140.192 success.	79
Figure 5.3	FTP connections successful	80
Figure 5.4	Initiate Lock Report	82
Figure 5.5	Propagate Lock Report	83
Figure 5.6	Obtain Lock Report	84
Figure 5.7	Update Lock Report	85
Figure 5.8	Commit Report	86
Figure 5.9	Unlock Lock Report	87
Figure 5.10	Release Lock Report	88
Figure 5.11	Timeline how ROWA-MSTS handle transaction	89
Figure 5.12	Graph locking phase over time	91

LIST OF ABBREVIATIONS

UMP	-	Universiti Malaysia Pahang
ROWA	-	Read-One-Write-All
ROWA-MSTS		Read-One-Write-All Monitoring Synchronization Transaction System

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Gantt Chart	97

CHAPTER 1

INTRODUCTION

1.0 Introduction

In the world where internet does all the business nowadays, the demand of transmitted information over networks increase rapidly and the demand for steady bandwidth seems to be out of control. Particularly, organizations need to provide updated data to users that might be geographically remote while handling vast amount of requested data distributed in multiple sites.

Distributed Systems keep the internet available anywhere regardless of the geography factor through Local Area Network (LAN), Wide Area Network (WAN) and Metropolitan Area Network (MAN). Distributed Systems is a system consists of two or more computers that coordinates their processing through the exchange of synchronous and asynchronous message passing [1]. Data can be accessed through synchronous or asynchronous replication and transaction in Distributed Systems.

The need of replicated data in Distributed Systems is to ensure that any data is backed up whenever problem occurs. In addition, replicated data is important as it will increase data availability and consistency [2]. The replicated data or replicas will be copied in every computer in the Distributed Systems. There are two types of data replication namely, synchronous replication and asynchronous replication.

Asynchronous replication usually transmitted inconsistently rather than a steady stream. Asynchronous replication also caused the receiver to have problems in receiving the data from the sender. Many vendors like Lotus Notes adopted asynchronous replication as a solution for managing replicated data because asynchronous replication works reasonably well for single object updates. However, asynchronous replication fails when involving multiple objects with single update.

Therefore, synchronous replication is the answer for constraints that the asynchronous brought. Synchronous replication will guarantee data consistency since it works based on quorum to execute the operations. Plus, synchronous replication provides a *'tight consistency'* between data stores [1]. For any copy that has been updated, the updates are applied immediately to all the copies within the same transaction [1]. This will ensure that all the copies in any site are the same and consistent. A consistent copy in all sites give advantages to the organization as it provides an updated data that is accessible anytime at any place in the distributed systems environment. However, synchronous replication require vast amount of storage capacity as multiple copies of replicated data stored in many sites and expensive synchronization mechanism are needed to maintain the consistency of data when changes are made. As a result, a proper strategy is needed to manage the replicated data in Distributed Systems.

Another problem that related to this system is regarding on the monitoring issue. Legacy system is the best example. Legacy system was developed few years back. Legacy system are often considered as problematic because it runs on outdated hardware or some even hard to find nowadays [5]. There are several possibilities that there might be no proper user manual or related documents about the antique system. This problem however, can be solved since the system is developed fully in LINUX environment. Since LINUX is an open source operating system, therefore, anyone from any part of the world can modify the system parallel to the current time.

In this project, Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS) will be proposed to solve the problems. ROWA-

MSTS is a system that will monitor the synchronous replicated data in distributed systems environment. A simple all-data-to-all sites replication technique called Read-One-Write-All (ROWA) technique is proposed to manage the synchronous data in Distributed Systems environment. By using ROWA, the read operation allowed to read any copies in the system. On the other hand, write operation required to write all copies of the data. Therefore, when updates of data performed, all value of data at all sites is the same. This technique provides read operations with high data availability and low communication cost. Besides that, in ROWA-MSTS system a usable interface will be developed as a result to ensure that this system is no alien to new network administrator. The system will also be able to generate the network report based on the transaction made on the particular time.

1.1 Problem Statement

Replication in distributed environment has become increasingly popular due to its high degree of availability, fault tolerance and enhance systems performance. These advantages of replication are important because it enables organizations to provide users with access to current data anytime or anywhere even if the users are geographically remote. In real working world, where data plays a crucial role like banking industry, this research is very important as the industry are dealing with consistent data like credit card numbers, amount of money transferred and other important transaction. At the same time access to the current data can be made anywhere in distributed systems. This way of data organization also introduces low data consistency among replicas when changes are made during transactions.

To understand the problem statement, consider a case of replicated data namely data x with 3 server in distributed environment. Each server is connected with one another. Define the following notations.

- PC A, PC B, PC C are the servers.
- X is transmitted data.
- α, β are the given locations.
- $T\alpha, T\beta$ are synchronous transaction at given locations.
- t is a specific time.

1.1.1 Case Study

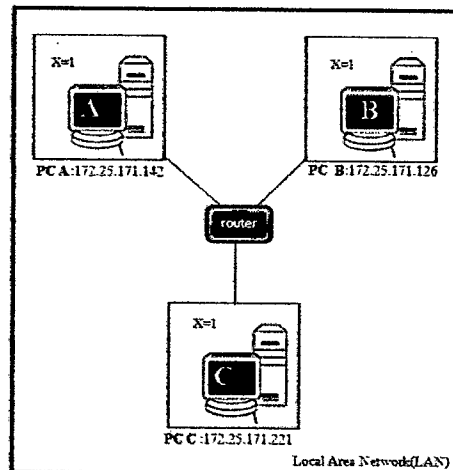


Figure 1.0 Replicated data stored in different servers

Figure 1.0 show that $X=1$ is replicated data that stored at server A, B and C at initial state.

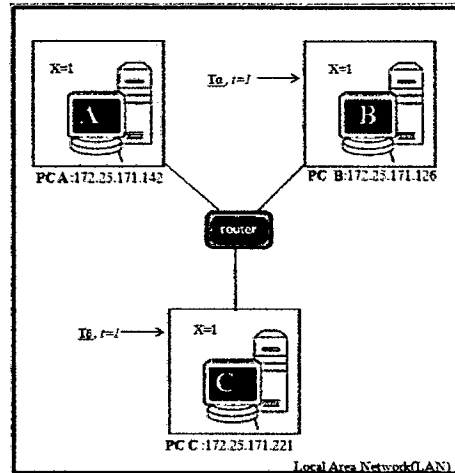


Figure 1.1 T_a request to update $X=2$ at server B

Figure 2.0 shows that at time $t=1$, T_a request to update $X=2$ at server B. At the same time where $t=1$, T_b also request to update $X=3$ at server C. These situation cause conflict in concurrency update requests for the particular data X . If there is no concurrency control for this situation, T_a will update data $X=2$ at server B and T_b will update the value of $X=3$ at server C at the same time, $t=1$. At time $t=2$, $X=2$ at server B, $X=3$ at server C and $X=1$ at server A. At time, $t=2$ data is not at consistent state. An issue arises, how do we maintain a steady and consistent transaction between all replicas in different servers at one time? How do we make sure that the synchronize replicated data can be transferred correctly?

1.2 Objectives

This system is developed to achieve objectives as follow:

- i. To develop a system name Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS).
- ii. To monitor the transaction synchronization in Distributed Systems.
- iii. To apply ROWA technique in order to maintain the data consistency in distributed system.

1.3 Scope

ROWA-MSTS system is developed especially for Network Administrator that using server, fixed and wireless network, Local Area Network (LAN) connection. This system only deals with synchronous replication and the research is focus only with data consistency issue.

1.4 Thesis Organization

This thesis will be organized as follows [13]: Chapter 2 explains about research that is done regarding to this project. This chapter divided into two major parts namely, research on the existing system and about techniques and technologies that is related to this project. The research is based on the previous paper or research that had done by other scientist or any current systems that implements the techniques related to this projects. This chapter also explains about techniques or technologies relevant to this project. In Chapter 3, the approach or overall framework about the development of project are discussed. This includes techniques, methods, or approaches that is used to develop and implemented throughout the project development. Chapter 3 also explains any justification of the techniques, hardware, software and methods that is used. Chapter 4 will document all the process that is involved in the development of the projects. Contents of this are depend on types of project developed. This chapter also exhibits the source code. Chapter 5 will discuss about the findings or result that is obtained and analysis of the data. Part of this chapter including the result analysis, project constraint and suggestion for improvement. Finally, in Chapter 6 will conclude overall projects that had developed. This includes the project summary, the summary of the data that is obtained and the effectiveness of data obtained with the objectives and problem statement. This chapter also discussed about the summary of methodology and implementation that is used throughout the project. Lastly, this chapter also includes any suggestion or new proposed approach regarding on the project for the research in the future.

CHAPTER 2

LITERATURE REVIEW

2.0 Introduction

This chapter will outline the general overview of any domain studies that is related to this project. This is purposely to increase the knowledge and understanding about the background of this project. This chapter also explains any research made that related to Read-One-Write-All Monitoring Transaction Synchronization (ROWA-MSTS) system regarding on the approach that is used throughout the development of this project. Chronologically, the first section is study on various replication techniques or methods that is currently used to deals with any data transaction in distributed systems environment. The second section will discuss on the already existed system that is related to this project. These existing systems can be a guide and giving some ideas on developing the project in the future. This subsection will discuss further and more specific about ROWA-MSTS system. This includes the modules involved and how the data are monitored. Next subsection explains about the software development life cycle that will be used throughout this project development along with its justification. Last section will discuss about software approach in developing the project. This includes any relevant software that might be needed.

2.1 Replication Techniques

Replication in distributed environment has become increasingly popular due to the high degree of data availability, fault tolerance and enhance the performance of a system. There are few types of replication techniques:

2.1.1 Read-One-Write-All (ROWA)

In ROWA techniques, replicas consistencies is guaranteed by the consistency of execution on one replica, but the client replicas are only updated and cannot provide accurate responses to queries. Synchronous replication methods guarantee that all replicas are maintained consistent at all times by executing each transaction locally only after all replicas have agreed on the execution order. Through this, a very strict level of consistency is maintained. However, because of the strict synchronization between replicas that is required for each transaction, synchronous replication methods have been deemed impractical and often times a centralized or client-server approach is preferred for systems that critically require strict consistency.

Any replicated database system needs to perform two main functions: execute transactions locally and make sure that the outcome of each transaction is re-erected at all replicas. The later task can be achieved in several ways. In the client-server approach, one server replica is in charge of executing/committing transactions locally before it disseminates the transactions to client replicas. This method is the closest to the single-server solution both in its performance. In ROWA-STD, the consistency of the replicas is guaranteed by the consistency of execution on one replica, but the client replicas are only lazily updated and cannot provide accurate responses to queries.

The asynchronous replication approach disseminates the transactions for parallel execution on all replicas. Each replica can answer queries and can initiate the execution of a transaction, but consistency is not always guaranteed, since connecting transactions

may be executed in different order at different replicas. These situations are detected and conflict resolution procedures are applied in order to restore consistency. If no conflicts arise, the transaction execution in the system can proceed very fast as each replica can locally execute the transactions as soon as they are received.

Synchronous replication methods guarantee that all replicas are maintained consistent at all times by executing each transaction locally only after all replicas have agreed on the execution order. This way a very strict level of consistency is maintained while providing high consistency response to clients. However, because of the strict synchronization between replicas that is required for each transaction, synchronous replication methods have been deemed non-practical and often times a centralized or client-server approach is preferred for systems that critically require strict consistency. It is believed that the trade between performance and client response time is too unbalanced for practical needs.

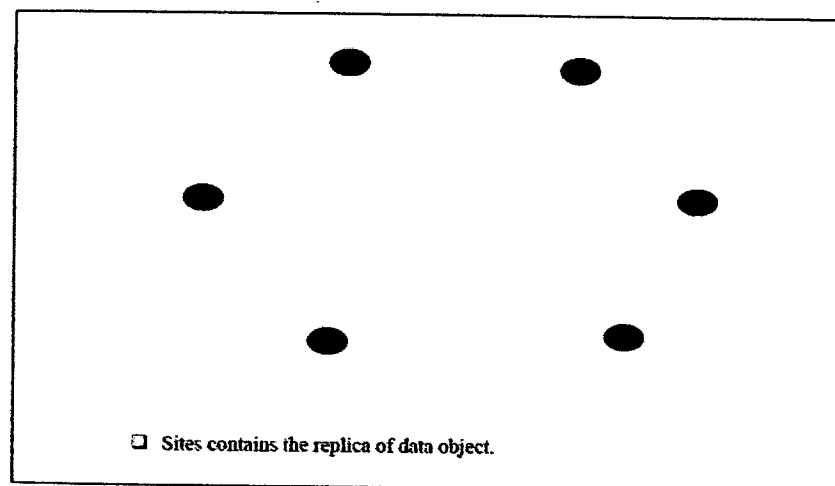


Figure 2.0: Sites contains the replica of data object

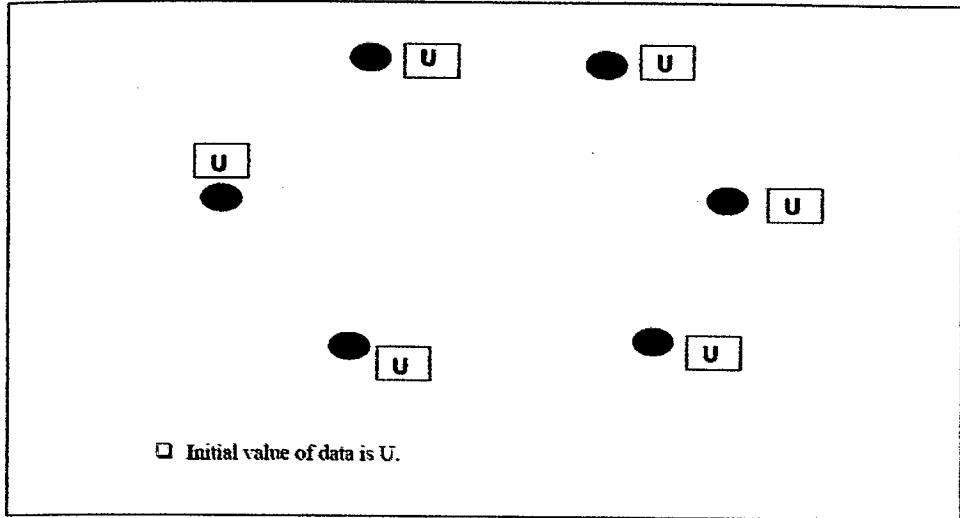


Figure 2.1 Initial value of data is U

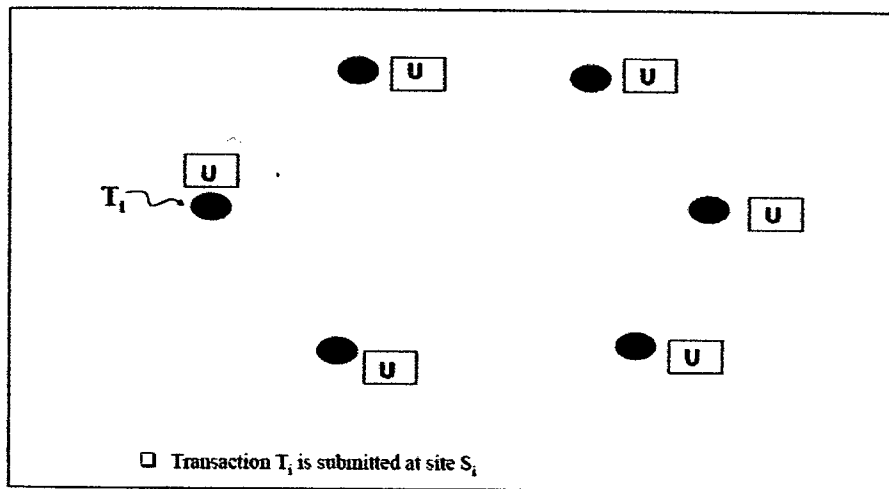


Figure 2.2 Transaction T_i is submitted to site S_i

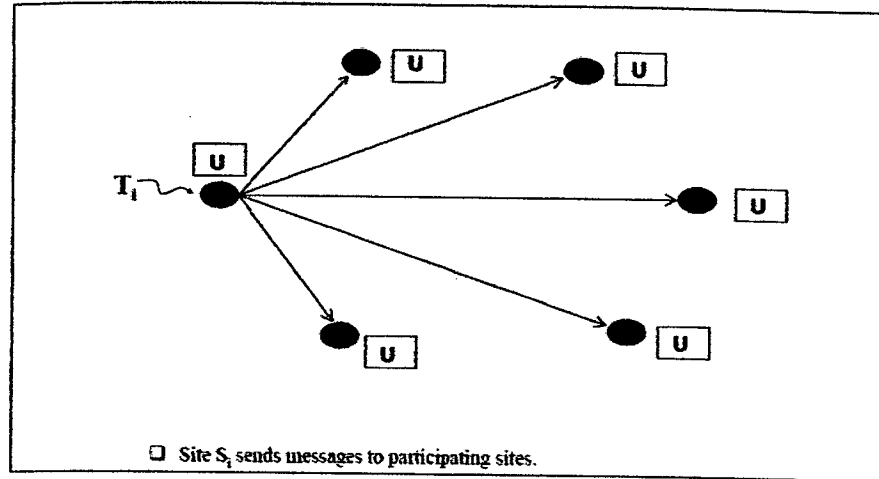


Figure 2.3 Site S_i sends messages to participating sites

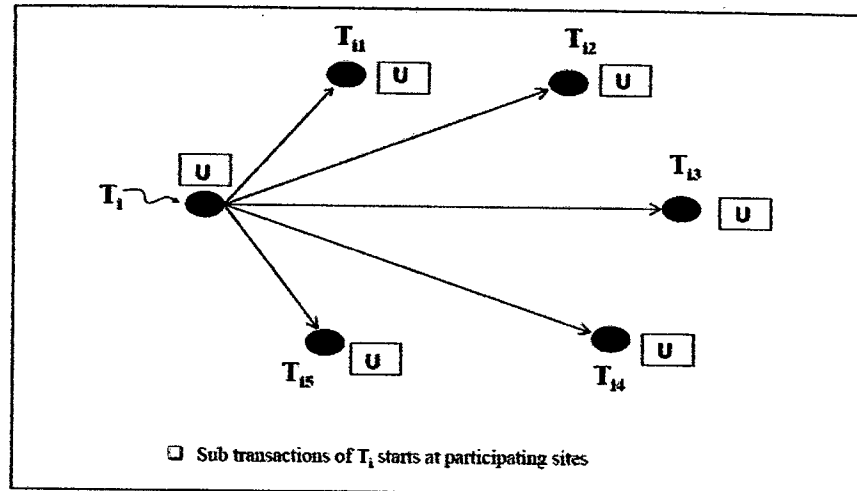


Figure 2.4 Sub transaction of T_i starts at participating sites.

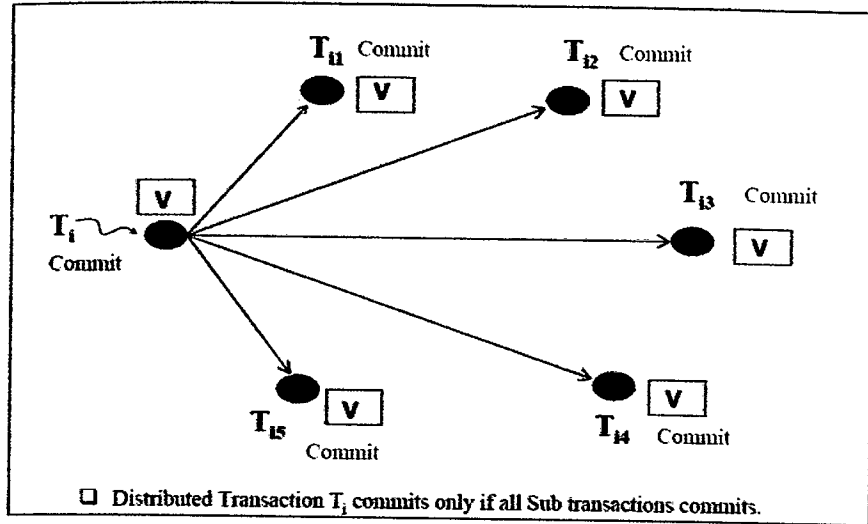


Figure 2.5 Replication Technique

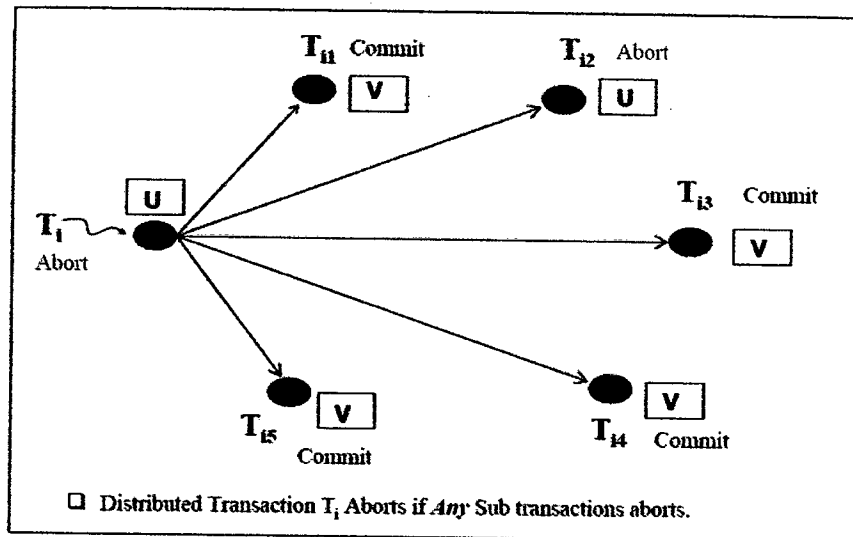


Figure 2.6 Distributed Transaction T_i aborts if any of sub transaction aborts.