# A NEW COMMIT PROCESSING UNDER DISTRIBUTED REAL TIME DATABASE SYSTEMS

**Y. Jayanta Singh**
**Vikas Humbe**
**S. C. Mehrotra**

## *Abstract*

*A real time database system (RTDBS) is a Transaction processing system that is designed to handle workloads where transactions have a complete deadline. Many Transaction complexities are in handling concurrency control and database recovery in distributed database systems. And real time applications made more complexes to these by placing deadlines on the response time of the database system. Objective of such RTDBS is to complete the processing of Transactions before expiry of these deadlines. The performances of such system depends on factors such as database system architecture, underlying processors, disks speeds, variety of operating conditions and traffic workloads. A preliminary report of Simulation of Commit processing in Distributed RTDBS is reported to bring up the performance of the system. The scheduling of data accesses is done in order to meet their applied deadlines and to minimize the number of Transactions that misses the deadlines. The performances of Transaction processing under different environments of distributed systems are reported. A new approach "Firm randomization" is introduced to distribute the work slot to sites uniformly.*

Keywords: Real time database system, distributed sites, deadlines of transactions.

## 1. INTRODUCTION

Distributed computing is a technique where heterogeneously networked computers are used to solve a single problem. If a transaction runs across two sites, it may commit at one site and fail at another site, leading to an inconsistent transaction. Two-phase commit protocol is most widely used to solve these problems [Silberschatz, Korth, Sudarshan(2002)]. A uniform commitment is guarantee by a commit protocol in a distributed transaction execution to ensure that all the participating sites agree on a final outcome. A result may be either a commit or an abort condition.

Transactions processing in any database system can have real time constraints. A real time database system, like Air traffic control, e-payments etc are the Transaction processing system that are designed to handle workloads where transactions have a complete deadlines. Many real time database applications in areas of communication systems and military systems are distributed in nature. To ensure transaction atomicity, commit protocols are implemented in distributed database system. This study investigated the performance implications of supporting transaction atomicity in a distributed real time database system (RTDBS) with the help of simulation. Mostly this studies concentrated on the management of deadline apply to a transaction and scheduling of arrival rates of the workload. Experimental performances of the system under variety of workloads and different system configuration are evaluated through this simulation.

## 2. Realtime database Concept

Database researchers have proposed varieties of commit protocols like Two phase commit [J. Gray (1978), C. Mohan, B. Lindsay and R. Obermark(1986)], Nested two phase commit, Presumed commit [B. Lampson and D. Lomet (1993)] and Presume abort, Broadcast Two phase commit [M.Oszu, P.Valduriez(1991)], Three phase commit [Minutesmansoftware, (2001(4E))] etc. These require exchanges of multiple messages, in multiple phases, between the participating sites where the distributed transaction executed. Several log records are generated to make permanent changes to the data disk, demanding some more transaction execution time [B. Lampson and D. Lomet (1993) P. Spiro, A Joshi and T. Rengarajan (1991), G. Samasras, K. Britton, A.Citon & C. Mohan (Feb.1993)]. Proper scheduling of transactions and management of its execution time are important factors in designing such systems.

Systems with the deadlines are called Real-time system [Silberschatz, Korth, Sudarshan(2002)]. Deadlines are categorized as follows: (i) Hard deadline: serious problem, such as a system crash, may occurs if a task is not completed within the deadline, (ii) Firm deadline: tasks has zero value if it is completed after the deadline and (iii) Soft deadline: the task diminishing (smaller) value if it is completed after the deadline. Designing Real-time system involved ensuring that there is enough processing power to meet the deadlines without requiring excessive Hardware resources.

Transactions processing in any database systems can have real time constraints. In [Robert Abbott & Hector Garcia-Molina (1988)] a model for scheduling transactions with deadlines on a single processor memory resident database system has been developed and evaluated the scheduling through simulation. A real time database system (RTDBS) is a Transaction processing system that is designed to handle workloads where transactions have complete deadlines. A survey in RTDBS is given in [O. Ulusoy (1.1994), O. Ulusoy (2.1994)]. And a detail study of deadline was discussed in [Abbott Robert and Hector Garcia-Molina (Sept.1992)]. Real-Time commit protocol designation has been investigated in [Jayant H, 25 Jayant H. M. Carey and M. Livney (Dec 1990), Jayant H. (Aug 1991), N. Soparkar, E. Levy, H. Korth and A. Silberschatz (1992)]. A centralized timed Two-phase Commit protocol has been designed [S. Davidson, I. Lee & V. Wolfe (1989)] where the fate of a transaction is guaranteed to be known to all the participants of the transaction by a deadline. In case of faults, it is not possible to provide such guarantee. The works described in the papers [Y. Yoon, (May 1994), 27 N. Soparkar, E. Levy, H. Korth and A. Silberschatz (1992)] are based on a common theme of allowing individual sites to unilaterally commit. This gives more timeliness of actions. Then the compensation-based approach is considered. The standard notation of transactions atomicity is not yet consider; instead, a relaxed notion of atomicity [Y. Jayanta and S.C. Mehrotra, (2002)] is provided. Real actions such as Firing a

weapon or dispensing cash may not be compensatable at all [E. Levy, H.Korth, A. Silberschatz (May 1991)]. In a RTDBS, the performance of the commit protocol is usually measured in terms of number of Transactions that complete before their deadlines. The transaction that miss their deadlines before the completion of processing are just killed or aborted and discarded from the system without being executed to completion [Jayant. H,M.Carey,M. Livney, (1992)].

### 3. A. Simulation model

This model is based on loose combination of the distributed database model presented in [M. Carey and M. Livny, (Aug 1988)] and the real time processing model of [Jayant. H, M. Carey and M. Livney, (1992)]. The model consists of a database that is distributed in a non-replicated manner, over all the available sites (say 8 sites in this case) connected by a network. In the actual model, each of the sites has six components: (i) a source: generate transactions, (ii) a transaction manager: models the execution behavior of the transaction, (iii) a concurrency control manager: implements the concurrency control algorithm, (iv) a resource manager: models the physical resources, (v) a recovery manager: implements the details commit protocol and (vi) a sink: collects statistics on the completed transactions. A network manager models the behavior of the communications network. The study is mainly concentrated in processing timing of commit protocol with managing the arrival rate of the workloads supply to the system in heavy load conditions. In this case, the part of concurrency control is not fully implemented. The modified model is shown in Figure 1. The goal of the study is to workout to minimized numbers of the percentage of misses transactions in order to optimize the atomicity problem in a distributed database system.

**Definition of components:**

*i) The source:*

The source is the component responsible for generating the

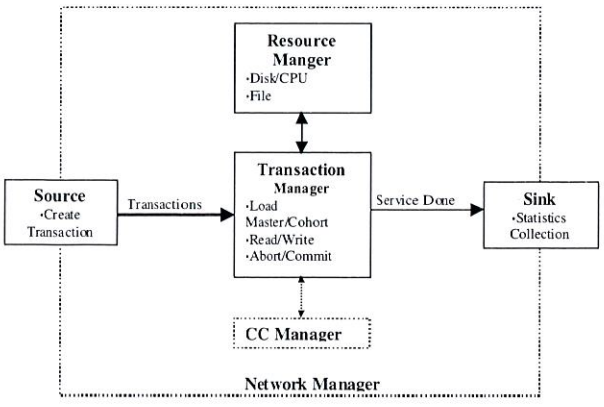


Figure 1: RTDBS Model

workloads for a site. The workloads are characterized in terms of files that they access and number of pages that they access and also update of a file.

*(ii) The transaction manager:*

The transaction manager is responsible for accepting a transaction from the source and modeling their execution. It deals with the execution behavior of the transaction. Each transaction in the workload has a general structure consisting of a master process and a number of cohorts. The master resides at the site where the transaction was submitted. Each cohort makes a sequence of read and writes requests to files that are stored at its sites. A transaction has one cohort at each site where it needs to access data.

To choose the execution sites for a transaction's cohorts, the decision rule is: if a file is present at the originating site, use the copy there; otherwise, choose uniformly from among the sites that have remote copies of the files. The transaction manager also models the details of the commit and abort protocols.

*(iii) The concurrency control manager:*

It deals with the implementation of the concurrency control algorithms. In this work, this module is not fully implemented. The effect of this is dependent on algorithm that is chosen during designing of the system.

*(iv) The resource manager:*

The resource manager models the physical resources like CPU, Disk, and files etc for writing to or accessing data or messages from them.

*(v) The sink:*

The sink deals with collection of statistics on the completed transactions.

*(vi)The Network Manager:*

The network manager encapsulates the model of the communications network. It is assumed that a local area network system, where the actual time on the wire for messages is negligible.

### 3. B. Execution model and Simulation Parameters

The execution model is discussed below. A common model of a distributed transaction is that there is one process, called Master, which is executed at the site where the transaction is submitted and a set of processes, and called Cohorts, which executes on behalf of the transaction at this various sites that are accessed by the transaction. In other words, each transaction has a master process that runs at its site of origination. The master process in turn sets up a collection of cohort's processes to perform the actual processing involved in running the transaction. When a cohort finishes executing its portion of a query, it sends an execution complete message to the master. When the master receives such a message from each cohort, it starts its execution process.

When a transaction is initiated, the set of files and data items that it will access are chosen by the source. The master is then loaded at its originating site and initiates the first phase of the protocol by sending PREPARE (to commit) messages in parallel to all the cohorts. Each cohort that is ready to commit, first force-writes a prepared log record to its local stable storage and then sends a YES vote to the master. At this stage, the cohort has entered a prepared state wherein it cannot unilaterally commit or abort the transaction but has to wait for the final decision from the master. On the other hand, each cohort that decides to abort force-writes an abort log record and sends a NO vote to the master. Since a NO vote acts like a veto, the cohort is permitted unilaterally abort the transaction without waiting for a response from the master.

After the master receives the votes from all the cohorts, it initiates the second phase of the protocol. If all the votes are YES, it moves to a committing state by force-writing a commit log record and sending COMMIT messages to all the cohorts. Each

cohort after receiving a COMMIT message moves to the committing state, force-writes a commit log record, and sends an acknowledgement (ACK) message to the master. If the master receives even one NO vote, it moves to the aborting state by force writing an abort log record and sends ABORT messages to those cohorts that are in the prepared state. These cohorts, after receiving the ABORT message, move to aborting state, force-write an abort log record and send an ACK message to the master. Finally, the master, after receiving acknowledgement from all the prepared cohorts, writes an end log record and then forgets and made the transaction free. The statistics are collected in the Sink.

The database is modeled as a collection of DBsize pages that are uniformly distributed across all the NumSites sites. At each site, transactions arrive under Poisson stream with rate Arrival Rate, and each transaction has an associated firm deadline. The deadline is assigned using the formula DT=AT+SF*RT, where DT, AT, SF and RT are the deadline, arrival rate, Slack factor and resource time respectively, of transaction T. The Resource time is the total service time at the resources that the transaction requires for its execution. The Slack factor is a constant that provides control over the tightness or slackness of the transaction deadlines.

In this model, each of the transaction in the supplied workload has the structure of the single master and multiple cohorts as described above. The number of sites at which each transaction executes is specified by the Fileselection time (DistDegree) parameter. At each of the execution sites, the number of pages accessed by the transaction's cohort varies uniformly between 0.5 and 1.5 times CohortSize. These pages are chosen randomly from among the database pages located at that site. A page that is read is updated with probability of WriteProb. Summary of the simulation parameter is given in table 1.

This simulation is mainly concentrated on processing timing of commit protocol. If the transaction's action deadline expires either before completion of its local processing, or before the master has written the global decision log record, then that transaction is killed and discarded.

## 4. Experiments and Result

The process of performance evaluation begins by first developing a base model. Further experiments were constructed around the base model experiments by varying a few parameters at a

| Parameters | Description |
|---|---|
| NumSites or Selectfile | Number of sites in the Database |
| Dbsize | Number of pages in the database. |
| ArrivalRate | Transaction arrival rate/site |
| Slackfactor | Slack factor in Deadline formula |
| FileSelectionTime | Degree of Freedom (DistDegree) |
| WriteProb | Page update probability |
| PageCPU | CPU page processing time |
| PageDisk | Disk page access time |
| TerminalThink | Time between completion of 1 transaction & submission of another |
| Numwrite | Number of Write Transactions |
| NumberReadT | Number of Read Transactions |

Table 1 Simulation Parameters

The values of the parameter set in the simulation are given in table 2. The CPU time to process a page is 10 milliseconds while disk access times are 20 milliseconds.

| Parameter Sets | Set Values | Parameters | Set Values |
|---|---|---|---|
| NumSites | 8 | FileSelection | 3 |
| Dbsize | vary (max. | Time | 10ms |
| ArrivalRate | 2400) | PageCPU | 20ms |
| Slackfactor | 6 to 8 job/sec | PageDisk | 0 to 0.5 sec |
| WriteProb | 4 | Terminal- | |
| | 0.5 | Think | |

Table 2. Values of simulation model Parameters

time. This experiment has been performed using different simulation language, such as, in paper [Jayant H., Ramesh G. Kriti.R, S. Seshadri January (1996)] using C++Sim [M.C. Little and D.L. McCue, (1994)], and in paper [Jayant. H, M. Carey and M. Livney, (1992)] using DeNet [Livny, M (1988)]. Here, for this study, the GPSS World simulator [Minutesmansoftware, (2001(4E))] is used as a simulator.

The performance metric of the experiments is MissPercent, that is the percentage of input transaction that the system is unable to complete before their deadline. The MissPercent values in range of 0% to 20% are taken to represent system performance under "Normal" loads, while range of 21% to 100% represents system performance under "heavy" loads. It is analyzing the performance of the system under different workload with varying the arrival rate of the transaction and their deadlines. It also analyzed the performance using this new concept of firm randomization technique (given below). Only the statistically significant results are discussed. Thus it also compares the performance of the system under Centralized commit and Distributed commit processing. The experimental results are discussed below.

### 4.1. Firm Randomization.

Classical distribution or randomization rules may give a repetition value in a short iteration. A work is assigned to a site number, which is generated by the randomization rule among 8 sites. For example, the first work may be assigned to site number 5 and the second work may be assigned site number 2. The third work may be assigned to site number 5 again, if the randomization rules generate 5. This value 5 should not be generated again at least before completion of assigning work to other free sites (for 8th iterations). A site or cohort need some time to complete a previous work and to process another new work. A new concept of randomization is introduced for distributing workload to sites so as to avoid assigning more worked load to a busy site. There are "n" numbers of sites in the system. It made the firm randomization in such a way that, any randomized value should not be repeated twice, at least in between "1 to n" iteration; repetition is allowed after "n+1" iteration. This at least helps to distribute a heavy work uniformly to all cohorts. The new techniques also give better statistics as compared to the normal randomization or distribution rule. The advantage of the method is shown in following discussion.

### 4.2. Normal load: Comparison of Centralized and Distributed performance

This section discusses the statistical results of the simulation. The distributed systems have higher percentage of miss Transactions than centralized system. This leads to design of a new

distributed commit-processing protocol to have a real-time committing performance. The comparison of Centralized and distributed performances is shown in figures 4.2.1

In heavy workload the number of miss transaction is higher for larger values of arrival rate. The results are shown in figure 4.2.2. and more details are given in figure 4.2.3.

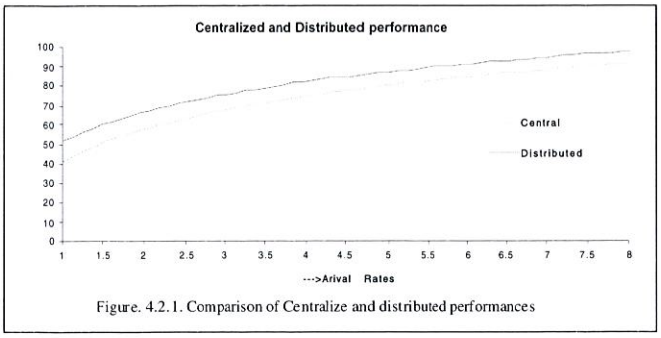


Figure. 4.2.1. Comparison of Centralize and distributed performances

### 4.3. Normal load: Analysis of Individual sites varying File selection time

This experiment is simulated by taking 8 distributed sites. Varying the file selection timing shows the performance analyses of each of the sites under a normal workload by varying the file selection timing in figure 4.3. The miss percentage of transaction is reduced in increasing the file selection timing.

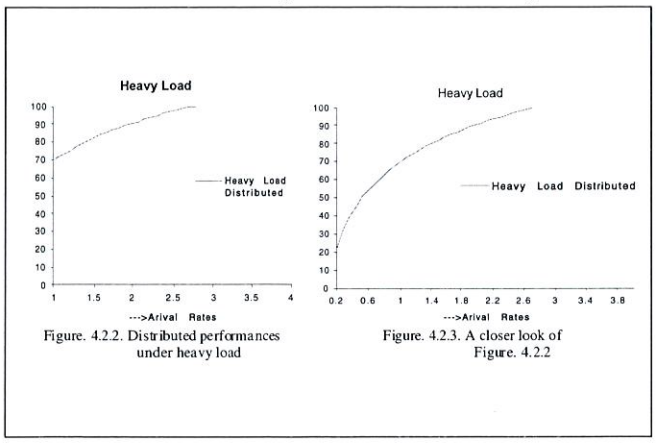


Figure. 4.2.2. Distributed performances under heavy load

Figure. 4.2.3. A closer look of Figure. 4.2.2

### 4.4. Normal Loads: Analysis of Individual sites varying arrival rate

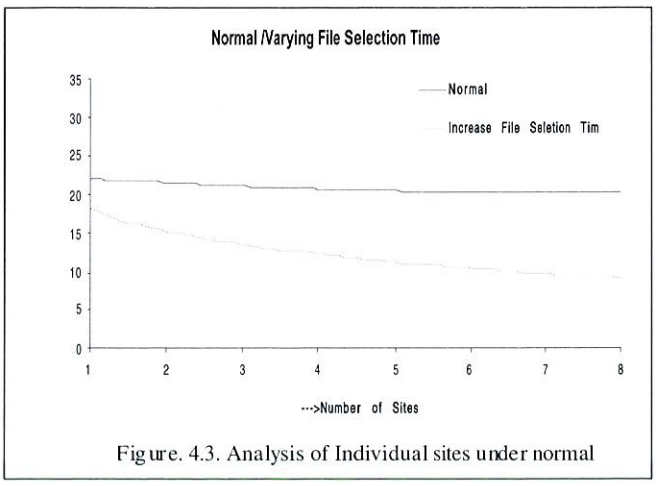


Figure. 4.3. Analysis of Individual sites under normal

The performances of the each of sites have been analyzed under normal load. The miss percentage is reduced at the lower values of arrival rate of the transactions. The success ratios of the transaction are also increased by lowering the arrival rate [Jayant H., Ramesh G. Kriti.R, S. Seshadri January (1996)]. The report is shown in figure 4.4.

### 4.5. Heavy loads: Analysis of Individual sites varying arrival rate

The performances of the each of sites have been analyzed under heavy load. The miss percentage is reduced at the lower values of arrival rate of the transactions. In both the normal and heavy

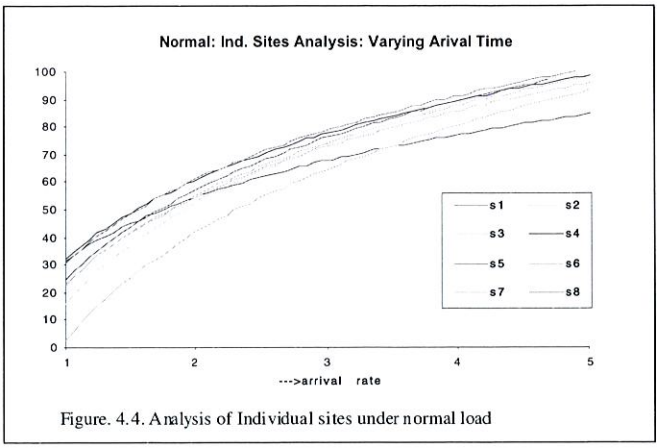


Figure. 4.4. Analysis of Individual sites under normal load

load the arrival rate play an important role to give a minimized miss percentage. The detail is shown in figure 4.5.1. More details are shown in figure 4.5.2.

4.6. Heavy loads: Comparison of Normal processing and firm Randomization

The performance under Normal and firm randomization process is given in figure 4.6. In distributing the transitions to different sites, the firm randomization process helps in avoiding continuous assignment of works to some specific busy sites. It helps to distribute the work uniformly to all sites. The figure 4.6, shows that the performance of the system gets better under firm randomization technique by about 10 to 6%. So proper choice of the distribution or randomization rules are also required to design a best-distributed commit processing system.

### 5. Conclusion

The distributed commit protocol has been simulated under different types of workload such as normal load and heavy load. The results of the simulation can be concluded in following points:

1. In both conditions the arrival rate plays a major role in reduc-

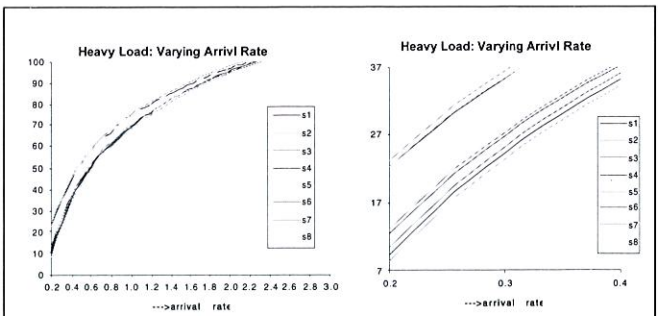


Figure. 4.5.1. Analysis of Individual sites under heavy load

Figure. 4.5.2. A closer look of Figure. 4.5.1

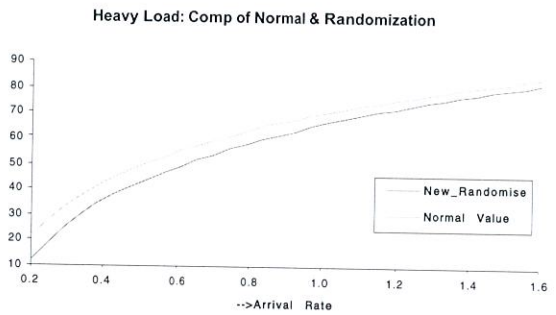


Figure. 4.6. Comparison performance under Normal and firm randomization process

ing number of miss percentage of the transactions.
2. In heavy load minimizing the value of arrival rate gives a lesser number of miss percentages.
3. It also demands the larger value of Fileselection timing to minimize the miss percentage.
4. The analysis report of the individual sites of the distributed system also shows that a lower value of the arrival rate is important to have a less miss percentage.
5. More time may be consumed, if there is no proper distribution, like the work is assigned to a busy site repetitively.
6. In such a situation the new approach of "firm randomization" method helps to distribute the works to sites uniformly and to minimize the miss percentage.

Acknowledgement
It is gratefully acknowledge Minuteman Software, P.O. Box 131, Holly Springs, North Carolina, 27540-0131, USA for providing free Study materials in their site http://www.minutemansoftware.com.

**References**

Abbott Robert and Hector Garcia- Molina (Sept.1992), "Scheduling Real-Time Transactions", ACM Trans. on Database Systems, 17(3).

B. Lampson and D. Lomet (1993), "A new Presumes Commit Optimization for Two phases Commit", Pro.of 19th VLDB Conference.

C. Mohan, B. Lindsay and R. Obermark(1986), "Transaction Management in the R* Distributed Database Management Systems, ACM TODS, 11(4)

D. Skeen (June1981), "Nonblocking Commit Protocols", Pro. ACM SIGMOD Conf.

E. Levy, H.Korth and A. Silberschatz (May 1991), "An optimistic commit protocol for distributed transaction management", Pro.of ACM SIGMOD Conf.

G.Samasras, K.Britton, A.Citon,C.Mohan (Feb.1993), "Two-phase Commit Optimizations & Tradeoffs in Commercial Environment", Pro.of IEEE Data Engineering Conf..

J. Gray (1978), "Notes on Database Operating Systems", Operating Systems: An Advanced Course, Lecture notes in Computer Science, 60.

Jayant. H, M. Carey and M. Livney (1992), "Data Access Scheduling in Firm Real time Database Systems", Real Time systems Journal, 4(3)

Jayant H., Ramesh G. Kriti.R, S. Seshadri January (1996), "Commit processing in Distributed Real-Time Database Systems", Tech. Report-TR-96-01, Pro. National Conf.on Software for Real-Time systems, Cochin, India Also as Pro. Of 17th IEEE Real-Time Systems Symposium, USA, December 1996.

Jayant H, "Performance Analysis of Real-time Database system", Tech .Report, TR 92-96, University of Maryland, USA.

Jayant H. M. Carey and M. Livney (Dec 1990), "Dynamic Real-Time Optimistic Concurrency Control", Proc. of 11th IEEE Real-Time Systems Symp.

Jayant H. (Aug 1991), "Transaction Scheduling in Firm Real-Time Database Systems", Ph.D. Thesis, Computer Science Dept. Univ. of Wisconsin, Madison.

Livny, M (1988)., DeNet User's Guide, Version 1.0, Computer Science Department, Univ. of Wisconsin, Madison

M.Oszu, P.Valduriez(1991), Principles of Distributed Database Systems, Prentice-Hall

M. Carey and M. Livny (Aug 1988), "Distributed Concurrency Control performance: A study of Algorithms, Distribution & Replication",Pro.of 14th VLDB Conf.

M.C. Little and D.L. McCue (1994), C++SIM User's Guide Public Release 1.5., Dept of Computing Science, Univ. of Newcastle upon Tyne, U.K.

Minutesmansoftware, (2001(4E)),GPSS world, North Carolina, U.S.A..[GPSS-Book]

N. Soparkar, E. Levy, H. Korth and A. Silberschatz (1992), "Adaptive Commitment for Real time Distributed Transactions", TR-92-15, CS, University of Texas (Austin)

O. Ulusoy (1.1994), "Research Issues in Real-time Database systems", Tech. Report BU- CEIS-94-32, Dept. of Com. Engg. and Inf. Science, Bilkent University, Turkey

O. Ulusoy (2.1994), "Processing Real-Time Transactions in a Replicated Database systems", Intl. Journal of Distributed and Parallel Database, 2(4)

P. Spiro, A Joshi and T. Rengarajan (1991), "Designing an Optimized Transaction Commit Protocol, Digital Technical Journal, 3(1)

Robert Abbott & Hector Garcia-Molina (1988), "Scheduling Real-Time Transaction: A performance Evaluation", Pro.of 14th VLDB Conf. Los Angeles.

Silberschatz, Korth, Sudarshan(2002), "Database system concept",4th (I.E), McGraow-Hill Pub. 698-709,903

S. Davidson, I. Lee & V. Wolfe (1989), "A protocol for Times Atomic Commitment" Proc. of 9th Intl. Conf. On Distributed Computing Systems

W. Kohler June (1981), "A survey of Techniques for Synchronization and Recovery in Decentralized Computer System", ACM Computing Surveys, 13(2)

Y. Yoon, (May 1994)"Transaction Scheduling and Commit processing for Real time Distributed Database Systems", Ph. D. Thesis, Korea Adv. Inst. of Sc. & Tech..

Y. Jayanta and S.C. Mehrotra, (2002), "Management of atomicity problem in worst possible Environment" Library Progress (International), 22(1), 25.

Dr. Y. JAYANTA SINGH
Department of Computer Studies
Skyline College, P.O.Box. 1797
Sharjah, UAE,
Phone: 00971+6+ 5439444 [Ext. 229]
Fax: 00971+6+ 5439994
Email: Email: y_jayanta@yahoo.com

Dr. VIKAS HUMBE
Research Scholar
Department of Computer Science and
I.T. Dr. Babasaheb Ambedkar
Marathwada University.
Aurangabad - 431004. (M.S). India.
Email: vikashumbe@yahoo.co.in

Dr. S. C. MEHROTRA
Department of Computer Science and I.T.
Dr. Babasaheb Ambedkar
Marathwada University.
Aurangabad - 431004. (M.S). India.
Phone: 0091+240+ 2400431-37 [Ext. 461]
Fax: 0091+240+ 2400291
Email: mehrotrasc@rediffmail.com