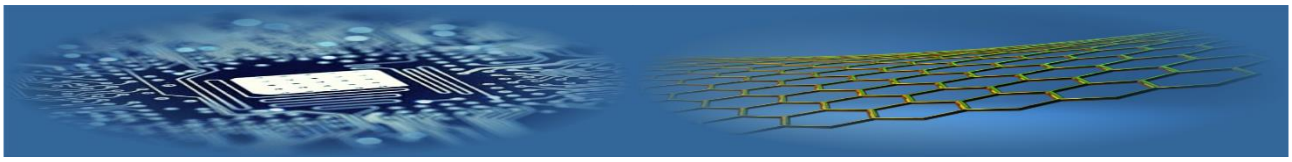

CCRTRD-A concurrency control technique in real time replicated databases

Anil Kumar Gupta*[1], Vishnu Swaroop[2]

[1]*Computer Sc. and Eng. Department, Bhagwant University, Ajmer, India*
[2]*Computer Sc. & Eng. Department, M. M. M. University of Technology, Gorakhpur, India*

**Corresponding author e-mail: anilgupta129@gmail.com*

Received 25 November 2017, www.cmnt.lv

Abstract

Data Replication is the process of using multiple copies of data located at multiple servers to help database systems to meet the stringent temporal constraints of time-critical applications, especially Internet-based services to resource for superior availability Each copy of the data is called a replica. These aspects encourage the researchers to study the requirement for realizing the benefits of replication. Current applications such as Web-based services, electronic commerce, mobile telecommunication system use the concept of replication for their functioning. The main goal of replication [5] is to improve availability and consistency. This helps mission critical services, such as many financial systems or reservation systems, where even a short outage can be very disruptive and expensive. Therefore, the major issue is to develop efficient replica concurrency control protocols that can tolerate the overload of the distributed system **.** Here, a new Concurrency Control protocol (CCRTRD) has been proposed for High Priority point and firm real-time database system using Static Two-Phase Locking (S2PL) for being deadlock free. It also includes High Priority given to a cohort after receiving PREPARE message from its coordinator. Proposed protocol shows significant performance improvement over O2PL and MIRROR in decreasing execution time of the current transaction and waiting time of transactions in queue.

Keywords:

Replicated database,
Distributed real time databases systems,
two-phase commit,
Concurrency control

1 Introduction

The Existing applications such as web-based services, electronic commerce, mobile telecommunication system, etc. are distributed in nature and manipulate time-critical databases. To enhance the performance and availability of such applications, one of the main techniques used is to replicate data on multiple sites of the network. The major issue is to develop efficient concurrency control protocols for replica that can tolerate the overload of the distributed system. If the system is not designed to handle overloads then it may lead to catastrophic results and some primordial transactions of the application can miss their deadlines. Although, many efforts have been made in the management of transactions for replicated databases in the real-time context [5, 6, 7]. No work deals with protocols that manage distributed real-time databases and simultaneously control the overload of the system. Some researchers have dealt with the scheduling of tasks under overload conditions when the real-time system is centralized such as in [5, 37] distributed as in [2, 39].

A distributed database is a single logical database that is spread physically across computers in multiple locations that are connected by a data communications network. The network must allow the users to share the data i.e. user at location A must be able to access the data at location B. Distributed real time database systems)DRTDBSs(can be defined as database systems that support real time transactions. They are used for a wide spectrum of applications such as air traffic control, stock market trading, banking, telemedicine etc. In DRTDBS, prior to, they have resulted in schemes wherein either the standard notions of database correctness are not fully supported or the maintenance of multiple historical versions of the data is required, or the real-time transaction semantics and performance metrics pose practical problems. Further, none of these studies have considered the optimistic two-phase locking)O2PL(protocol]4, 40[. Although, it is the best-performing algorithm in conventional)non-real time(replicated database systems]4, 38[. Transactions in a real-time database are classified into three types, viz. hard, soft and firm. The classification is based on how the application is affected by the violation of transaction time constraints. This paper reports efficient solutions for some of the issues important to the performance of replicated firm deadline based DRTDBS]9, 10, 41[. The performance of DRTDBS depends on several factors such as specification of transaction's deadline, priority assignment policy, scheduling transactions with deadlines, time conscious buffer and locks management, commit procedure etc. One of the primary performance determinants is the policy used to schedule transactions for the system resources. The resources that are typically scheduled are processors, main memory, disks and the data items stored in database. One possible goal of replication is to have replicas, which behaves functionally like no replicated servers. This goal can be stated precisely by the concept of one-copy serializability, which extends the concept of serializability to a system where multiple replica are present. An execution is one-copy serializable if it has the same effect as a serial execution on a one-copy database. We would like a system to ensure that its executions are one-copy serializable. In such a system, the user is unaware that data is replicated. There are two approaches to sending a transaction's updates to replicas: synchronous and asynchronous. In the synchronous approach, when a transaction updates a data item, say x, the update is sent to all replicas of x. These updates of the replicas execute within the context of the transaction [16, 17, 18, 19]. This is called synchronous because all replicas are, in effect, updated at the same time. Although sometimes this is feasible, often it is not, because it produces a heavy distributed transaction load. It implies that all transactions that update replicated data must use two-phase commit, which entails significant communications cost. Fortunately, looser synchronization can be used which allows replicas to be updated independently. This is called asynchronous replication, where a transaction directly updates one replica and the update is propagated to other replicas later. In a replicated environment means when there is more than one copy of a data item distributed on different sites then there is one major issue how to update replicas of data item on different site with efficient concurrency control mechanism. While few works have been done in area of replica concurrency control like 2PL, O2PL and OCC but they are not up to the standard of present requirement.

In this paper, we focus on development of concurrency control protocol and one copy serializability in replicated firm real-time database system. In firm real-time system, the major issue is deadline of completion means if the transaction misses its deadline then the transaction is of no usage, hence it is killed if the transaction misses its deadline. Our work retains the standard one-copy-serializability as the correctness criterion and focuses on removal of the disadvantages of locking and OCC based concurrency control protocols. Finally, we also include an investigation of the O2PL algorithm, which has not been studied before in the real-time context.

The rest of this paper is organised as follow. Section II enlists the related work. Section III discusses about the CCRTRD protocols. Section IV provides performance evaluation of the proposed protocol. Section V concludes the paper.

2 Related work

Concurrency control algorithms and real-time conflict resolution mechanisms for RTDBS have been studied extensively (e.g. [9, 10, 11, 25]). However, concurrency control for replicated DRTDBS has only been studied in [21, 22, 23, 25]. An algorithm for maintaining consistency and improving the performance of replicated DRTDBS is proposed in [21]. In this algorithm, a multi version technique is used to increase the degree of concurrency. Replication control algorithms, that integrate real time scheduling and replication control, are proposed in [22, 23]. These algorithms employ Epsilon-serializability (ESR) [26] which is less stringent than conventional one copy-serializability.

The performance of the classical distributed 2PL locking protocol (augmented with the priority abort (PA) and priority inheritance (PI) conflict resolution mechanisms) and of OCC algorithms in replicated DRTDBS was studied in [25, 27] for real-time applications with "soft" deadlines. The results indicate that 2PL-PA outperforms 2PLPI only when the update transaction ratio and the level of data replication are both low. Similarly, the performance of OCC is good only under light transaction loads. Making clear-cut recommendations on the performance of protocols in the soft deadline environment is rendered difficult. However, by the following points it is clear.

1. There are two metrics – Missed Deadlines and Mean Tardiness, and protocols which improve one metric usually degrade the other.
2. The choice of the post-deadline value function has considerable impact on relative protocol performance.

Concurrency control algorithms and real-time conflict resolution mechanisms for RTDBS have been studied extensively)e.g.]10, 11[. An algorithm for maintaining consistency and improving the performance of replicated DRTDBS is proposed in]20[. In this algorithm, a multi version technique is used to increase the degree of concurrency. Replication control algorithms that integrate real-time scheduling and replication control are proposed in]6, 7, 30, 31[. These algorithms employ Epsilon-serializability)ESR(which is less stringent than conventional one-copy-serializability. The performance of the classical distributed 2PL locking protocol)augmented with the priority abort)PA(and priority inheritance)PI(conflict resolution mechanisms(and validation-based algorithms was studied in for real time applications with "soft" deadlines operating on replicated DRTDBS]6,32[. The performance of OCC is good only under light transaction loads. In]4, 33[, a conditional priority inheritance mechanism is proposed to handle priority inversion. This mechanism capitalizes on the advantages of both priority abort and priority inheritance in real-time data conflict resolution. It outperforms both priority abort and priority inheritance when integrated with two phase locking in centralized real-time databases. However, the protocol assumes that the length)in terms of the number of data accesses(of transactions is known in advance, which may not be practical in general, especially for distributed applications. In contrast, our state-conscious priority blocking and state-conscious priority inheritance protocols resolve real-time data conflicts based on the states of transactions rather than their lengths. ROWA)"read one copy, write all copies"(category with respect to their treatment of replicated data is another approach in the following description; we assume that the reader is familiar with the standard concepts of distributed transaction execution]4, 8,[.

A Real-Time Database Systems (RTDBS) processes transactions with timing constraints (deadlines) [12, 28, 29]. Its primary performance criterion is timeliness, not average response time or throughput. The scheduling of transactions is driven by priority order. Given these challenges, considerable research has recently been devoted to designing concurrency control methods for RTDBSs and to evaluating their performance. Most of these methods are based on one of the two basic concurrency control mechanisms: locking or optimistic concurrency control (OCC). In real-time systems transactions are scheduled according to their priorities. Therefore, high priority transactions are executed before lower priority transactions. This is true only if a high priority transaction has some database operation ready for execution. If no operation from a higher priority transaction is ready for execution, then an operation from a lower priority transaction can execute its database operation. Therefore, the operation of the higher priority transaction may conflict with the already executed operation of the lower priority transaction. In traditional methods, a higher priority transaction must wait for the release of the resource. This is the priority inversion problem presented earlier. Therefore, data conflicts in concurrency control should also be based on transaction priorities or criticalness or both. Hence, numerous traditional concurrency control methods have been extended to the real-time database systems.

In classical two-phase locking protocol, transactions set read locks on objects that they read, and these locks are later upgraded to write locks for the data objects that are updated. If a lock requested is denied, the requesting transaction is blocked until the lock is released. Read locks call is shared. While write locks are exclusive. For real-time database Systems, two phase locking needs to be augmented with a priority based conflict resolution scheme to ensure that higher priority transactions are not delayed by lower priority transactions. In High Priority scheme, all data conflicts are resolved in favour of the transaction with the higher priority. In distributed Two Phase Locking as described in [1] when a transaction arrives at a site in distributed system, it is divided in sub transactions known as cohorts and processes, which update replicas of data item, are known as replica updaters. If a request by a cohort is a read lock on a data item, then any lock is not required on the replicas at remote sites but if the request is a write lock then write locks are required on all the replicas of the data item. In this protocol write locks are obtained as the transaction executes with the transaction blocking on a write request until all the copies of the data items to be updated have been successfully locked by a local cohort and its remote updaters on replicas. Only the data locked by a local cohort is updated in the data processing phase of transaction. Remote copies locked by updaters are updated after those updaters have received list of items to be updated with the PREPARE message during the first phase of commit protocol. Write locks are only released after they are committed or aborted.

In distributed optimistic two-phase locking as described in distributed environment for replica concurrency control is termed as O2PL. In this algorithm when a cohort requests for a write lock, it is immediately given to it if lock is available. However, it defers requesting write locks on replicas at remote site in the second phase of commit protocol. In this protocol, when a cohort updates a data item it requests for write locks on replicas after it has received PREPARE message from its master site. What happens that after getting PREPARE message from its coordinator the cohort sends a PREPARE message to all of its remote updaters of the corresponding data item. With the PREPARE message, it also sends list of the data items to be updated and the processes used in updating the data items. After that, remote updaters obtain the locks on data item to be updated and sends COMMIT message to the cohort after completing the updating. Now after getting COMMIT message from replica updaters, the cohort sends PREPARED message to its coordinator. Since the locks are deferred to the second phase of commit protocol, there is a chance of both block and abort also due to arriving of higher priority transaction than the executing one.

3 The CCRTRD protocol

We now present our new replica concurrency control protocol called CCRTRD augments. The O2PL protocol with a novel, simple to implement, state based conflict resolution mechanism called state-conscious priority blocking. In this scheme, the choice of conflict resolution method is a dynamic function of the states of the distributed transactions involved in the conflict. A feature of the design is that acquiring the state knowledge does not require inter-site communication or synchronization, nor does it require modifications of the two-phase commit protocol. Two real-time conflict resolution mechanisms are used in CCRTRD. Even though S2PL [4, 35, 36] is a deadlock free mechanism but it slows down the concurrent processing of multi transactions. This is due to locking of all the data until the end of the commit phase. Also, if a higher transaction arrives at a site than executing one then current transaction is aborted and lock is made available to higher priority one. This makes the wastage of consumed resources. Hence, we propose here a new mechanism with augmentation of S2PL. In classical two-phase locking protocol [14,15], transactions set read locks on objects that they read, and these locks are later upgraded to write locks for the data objects that are updated. If a lock requested is denied, the requesting transaction is blocked until the lock is released. Read locks can be shared, while write locks are exclusive. For real-time database systems, two-phase locking needs to be augmented with a priority-based conflict resolution scheme to ensure that higher priority transactions are not delayed by lower priority transactions. In High Priority scheme [13], all data conflicts are resolved in favor of the transaction with the higher priority. When a transaction requests a lock on an object held by other transactions in a conflicting lock mode, if the requester’s priority is higher than that of all the lock holders, the holders are restarted, and the requester is granted the lock; if the requester’s priority is lower, it waits for the lock holders to release the lock. In addition, a new read lock requester can join a group of read lock holders only if its priority is higher than that of all waiting write lock operations. This protocol is referred to as 2PL-HP [13]. It is important to note that 2PL-HP loses some of the basic 2PL algorithm’s blocking factor due to the partially restart-based nature of the High Priority scheme. We will use here a term High Priority point(HPP) with Block(BK)/ Do not Abort(DAB) which means if a cohort reaches its High Priority point (HPP) than it will not be aborted against a higher priority transaction at that site. It means DAB is used here. And if a lower priority transaction demands a lock then it will be blocked against a higher priority executing one. It means BK is used here. The proposed mechanism is:

HPP of a cohort: A cohort reaches its HPP after sending a PREPARE message to its replica updaters in its execution phase i.e. in first phase of 2PC.

HPP of a replica updater: A replica updater reaches HPP after gaining locks on needed data items. By this mechanism some significant improvements can be noted in S2PL. Since after HPP a cohort has a less probability of abortion hence a blocked transaction can borrow data from executing one. It means waiting and executing time of blocked transaction will get reduced and less probability to access which is very needed in a firm RTDBMS. Also by sending PREPARE message to its replica updaters as shown in the waiting time of a cohort between sending PREPARE message to its updaters and receiving COMMIT message from them will get reduced. Hence, over all time of execution of transaction will get reduced.

Algorithm

```
For a cohort: HPP=F;
EXTfinish (execution finished) =f;
if (message=INITIATE COHORT)
        {
        Start execution of cohort;
        EXTfinish=T;
        }
else if (EXTfinish=T)
        {
        Send PREPARE message to its replica updaters;
        HPP=T;
        HPP Send WORK DONE message to its coordinator;
        }
For a replica updater:
if (message=PREPARE && lock obtained=T)
        {
        HPP=T;
        After execution send COMMIT message to its cohort;
        }
```

4 Performance evaluation of ccrtrd

We have addressed the problem of managing firm-real time database system that access replicated data in a distributed system that may be replicated. The main ideas of the protocol are to associate an importance value to each submitted transaction. The contributions are given below.

1. Proposed mechanism provides a deadlock free environment.
2. The waiting time of blocked transaction is reduced.
3. The execution time of current transaction is reduced.
4. Wasting of resources is minimized.
5. Easy implementation and integration with S2PL and 2PC.
6. Decreases the waiting time of transactions in wait queue.

To evaluate the performance of the CCRTRD protocol, we have developed a detailed simulation model of a distributed real-time database system)DRTDBS(. Our model is based on the distributed database model presented in]3[, which has also been used in several other studies of distributed database system behaviour.

The database is modelled as a collection of DBSize pages that are distributed over NumSites sites. The number of replicas of each page, that is, the “replication degree”, is decided by the ReplDegree parameter. The physical resources at each site consist of NumCPUs CPUs, NumDataDisks data disks and NumLogDisks log disks. At each site, there is a single common queue for the CPUs and the scheduling policy is pre-emptive Highest- Priority-First. Each of the disks has its own queue and is scheduled according to a Head-on-Line policy, with the request queue being ordered by transaction priority. The PageCPU and PageDisk parameters capture the CPU and disk processing times per data page, respectively. The parameter InitWriteCPU models the CPU overhead associated with initiating a disk write for an updated page. When a transaction makes a request for accessing a data page, the data page may be found in the buffer pool, or it may have to be accessed from the disk. The BufHitRatio parameter gives the probability of finding a requested page already resident in the buffer pool. The communication network is simply modelled as a switch that routes messages and the CPU overhead of message transfer is taken into account at both the sending and receiving sites and its value is determined by the MsgCPU parameter – the network delays are subsumed in this parameter. This means that there are two classes of CPU requests local data processing requests and message processing requests. Transactions arrive in a Poisson stream with rate ArrivalRate, and each transaction has an associated firm deadline, assigned as described below. Each transaction randomly chooses a site in the system to be the site where the transaction originates and then forks off cohorts at all the sites where it must access data. Transactions in a distributed system can execute in either sequential or parallel fashion. The distinction is that cohorts in a sequential transaction execute one after another, whereas cohorts in a parallel transaction are started together and execute independently until commit processing is initiated. We consider only sequential transactions in this study. Note, however, that the execution of replica updaters belonging to the same cohort is always in parallel. A summary of the parameters used in the simulation model are presented in Table 1 given below.

Parameter	Meaning	Setting
NumSites	Number of sites	4
DbSize	Number of Pages in the databases	1000 pages
ReplDegree	Degree of Replication	4
NumCpus	Number of CPUs per site	2
NumDataDisks	Number of data disks per site	4
NumLogDisks	Number of log disks per site	1
BufHitRatio	Buffer hit ratio on a site	0.1
ArrivalRate	Transaction arrival rate (Trans./Second)	Varied
SlackFactor	Slack factor in deadline assignment	6.0
TranSize	No. of pages accessed per trans.	16 pages
UpdateFreq	Update frequency	0.25
PageCpu	CPU page processing time	10 ms
InitWriteCpu	Time to initiate a disk write	2 ms
PageDisk	Disk page access time	20 ms
LogDisk	Log force time	5 ms
Msgcpu	CPU message send/receive time	1 ms

The total number of pages accessed by a transaction, ignoring replicas, varies uniformly between 0.5 and 1.5 times *TransSize*. These pages are chosen uniformly (without replacement) from the entire database. The proportion of accessed pages that are also updated is determined by *UpdateFreq* Upon arrival, each transaction *T*is assigned a firm completion deadline using the formula.

Deadline (T) =Arrival Time (t)+ Slack Factor*R_T,

where *Deadline(T)*, *ArrivalTime(T)*, and R_T are the deadline, arrival time, and resource time, respectively, of transaction *T*, while *SlackFactor* is a slack factor that provides control of the tightness/slackness of transaction deadlines. The resource time is the total service time at the resources at all sites that the transaction requires for its execution *in the absence of data replication*.

Figures 1 and 2 present the missed deadline percentages of transactions for the O2PL, MIRROR and CCRTRD protocols under normal loads and heavy loads, respectively. To help isolate the performance degradation arising out of concurrency control, we also show the performance of CCRTRD that is, a protocol that processes read and write requests like O2PL, but ignores any data conflicts that arise in this process and instead grants all data requests immediately. It is important to note that CCRTRD is only used as an artificial baseline in our experiments.

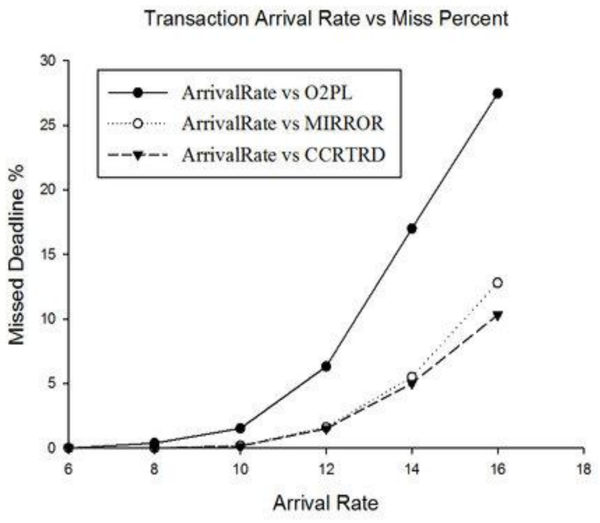

FIGURE 1 Miss rate under normal load

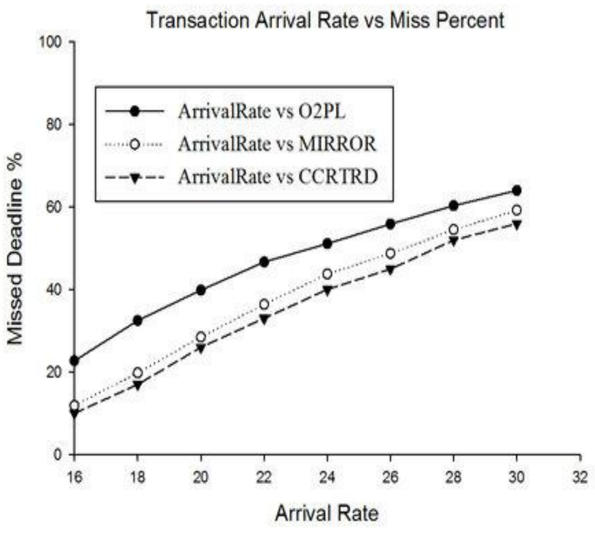

FIGURE 2 Miss rate under heavy load

We observe that CCRTRD protocol has the best performance among all the protocols.

5 Conclusion

At this point, we have exploited the difficulty of accessing the replicated data in firm RTDBS mainly concurrent execution of transaction in real time replicated database problem in conflict mode. This proposed mechanism can be easily integrated and implemented in current systems. This proposed CCRTRD protocol outperform other protocols like O2PL, OCC and MIRROR. Also in this paper, it is ssaid that a blocked transaction can borrow data item after reaching the High Priority point by the executing transaction, in this way the waiting time of transaction in queue will be decreased.

As a part of future work, an exhaustive real-life implementation work is required to establish this approach as a value-based commercial product.

References

[1] Gray J 1979 *Notes On Database Operating Systems,"in Operating Systems*: *An Advanced Course* R. Bayer, R. Graham, G. Seegmuller, eds., Springer-Verlag

[2] M Valduriez P 1991 *Principles of Distributed Database Systems* Prentice-Hall. P. Bernstein, V. Hadzilacos and N. Goodman

[3] Xiong M et al. 1999 MIRROR: A State-Conscious Concurrency Control Protocol for Replicated Real-Time Databases *Computer Science Department Faculty Publication Series*

[4] Shanker U et al 2006 The SWIFT-Real Time Commit Protocol *International Journal of Distributed and Parallel Databases, Springer Verlag* **20**(1) 29-56

[5] Abbadi A, Toueg S 1989 Maintaining availability I partitioned replicated databases *ACM Trans. Database Syst* **14**)2(264–90

[6] Ramamritham Son S H, Di Pippo L 2004 Real-Time Databases and Data Services *Real-Time Systems J* **28** 179-216

[7] Robert A, Garcia-Molina H 1992 Scheduling Real-Time Transactions *ACM Trans. on Database Systems* **17**)3(

[8] Levy E, Korth H, Silberschatz 1991 An optimistic commit protocol for distributed transaction management *Proceedings of ACM SIGMOD Conference*

[9] Jayant H, Carey M, Livney 1992 Data Access Scheduling in Firm Real Time Database Systems *Real Time Systems Journal* **4**)3(

[10] Singh Jayanta, Mehrotra S C 2006 Performance analysis of a Real Time Distributed Database System through simulation *15th IASTED International Conf. on APPLIED SIMULATION & MODELLING, Greece*

[11] Singh Jayanta, Mehrotra S C 2009 A study on transaction scheduling in a real-time distributed system *EUROSIS's Annual Industrial Simulation Conference, UK*

[12] Ramamritham K, Son S H, DiPippo L 2004 Real-Time Databases and Data Services *Real-Time Systems J* **28** 179-216

[13] Abbott R, Garcia-Molina H 1988 Scheduling Real-Time Transactions: A Performance Evaluation *Proceedings of the 14th VLDB Conference, August*

[14] Abbott R, Garcia-Molina H 1989 Scheduling Real-Time Transactions with Disk Resident Data *Proceedings of the 15th VLDB Conference, August 1989. IRACST – International Journal of Computer Networks and Wireless Communications)IJCNWC(, ISSN: 2250-3501* **2***(3), June 2012 401*

[15] Eswaran K P, Gray J N, Lorie R A, Traiger I L 1976 The Notionsof Consistency and Predicate Locks in a Database System *Communications of the ACM* **19**)11(

[16] Son S 1987 Using Replication for High Performance Database Support in Distributed Real-Time Systems *Proceedings of the 8th IEEE Real-Time Systems Symposium* 79-86

[17] Ahamad M, Ammar M H 1989 Performance Characterization of Quorum- Consensus Algorithms for Replicated Data *IEEE TSE* **15**)4(492–6

[18] Peleg D, Wool A 1995 The Availability of Quorum Systems *Information and Computation* **123**)2(210–23

[19] Thomas R H 1979 A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases *ACM Transactions on Database Systems* **4**)9(180–209

[20] Nicola M, Jarke M 2000 Performance Modeling of Distributed and Replicated Databases *IEEE Trans. on Knowledge and Data*

Engineering **12**)4(645–72

[21] Guerraoui R, Felber P, Garbinato B, Mazouni K R 1998 System support for object groups *ACM OOPSLA'98*

[22] Kemme B, Alonso G 2000 Don't be lazy, be consistent: Postgres-R, A new way to implement Database Replication *In Proc. of VLDB'01*

[23] Naor M, Wool A 1998 The Load, Capacity, and Availability of Quorum Systems *SIAM Journal of Computing* **27**)2(423–47

[24] Nicola M, Jarke M 2000 Performance Modeling of Distributed and Replicated Databases *IEEE Trans. on Knowledge and Data Engineering* **12**)4(645–72

[25] Patino Martinez M, Jimenez Peris R, Kemme B, Alonso G 2000 Scalable Replication in Database Clusters *In Proc. of Int. Conf. on Distributed Computing, DISC'00, LNCS-1914. Toledo, Spain* 315–29

[26] Peleg D, Wool A 1995 The Availability of Quorum Systems *Information and Computation* **123**)2(210–23

[27] Saha D, Rangarajan S, Tripathi S K 1996 An analysis of the average message overhead in replica control protocols *IEEE Trans. on Paral. and Dist. Syst.* **7**)10(

[28] Theel O, Pagnia H 1998 Optimal Replica Control Protocols Exhibit Symmetric Operation Availabilities *In Proc. Of Symp. on Fault-Tolerant Computing)FTCS(*

[29] Thomas R 1979 A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases *ACM Transactions on Database Systems* **4**)9(180–209

[30] Agrawal D, Abbadi A E 1990 The Tree Quorum Protocol: An Efficient Approach for Managing Replicated Data *In Proc. Of the 16th VLDB Conf., Brisbane, Australia*

[31] Ahamad M, Ammar M H 1989 Performance Characterization of Quorum-Consensus Algorithms for Replicated Data *IEEE TSE* **15**)4(492–6

[32] Alonso G, Bausch W, Pautasso C, Hallett M, Kahn A 2000 *Dependable Computing in Virtual Laboratories*

[33] Cachopo J, Rito-Silva A 2006 Versioned boxes as the basis for memory transactions *Sci. Comput. Program.* **63**)2(172–85

[34] Couceiro M, Romano P, Carvalho N, Rodrigues L 2009 D2STM: Dependable Distributed Software Transactional Memory *In Proc. International Symposium on Dependable Computing)PRDC(. IEEE Computer Society Press*

[35] Defago X, Schiper A, Urban P 2004 Total order broadcast and multicast algorithms: Taxonomy and survey *ACM Computing Surveys* **36**)4(372–421

[36] Ekwall R, Schiper A 2007 Modeling and validating the performance of atomic broadcast algorithms in high-latency networks *In Proc. Euro-Par 2007, Parallel Processing, Lecture Notes in Computer Science* 574–86 Springer

[37] Srivastava A, Shanker U, Tiwari S K 2012 A Replication Protocol for Real Time Database System *International Journal of Electronics and Computer Science Engineering (IJECSE)* **1**(3) 1602-8

[38] Srivastava A, Shanker U, Tiwari S K 2012 Transaction Processing in Replicated Data in DDBMS *International Journal of Modern Engineering Research (IJMER)* **2**(4) 2409-16

[39] Srivastava A, Shanker U, Tiwari S K 2012 Transaction Management in Homogeneous Distributed Real Time Replicated Database Systems *International Journal of Advanced Research in Computer Science and Software Engineering* **2**(6) 190-6

[40] Srivastava A, Shanker U, Tiwari S K 2012 A Protocol for Concurrency in Distributed Real Time Replicated Database System *International Journal of Computer Networks and Wireless Communication (IJCNWC)* **2**(3) 398-401

[41] Srivastava A, Shanker U, Tiwari S K 2012 Data Freshness in Distributed Real Time Replicated Database Systems *Global Journal Computing and Technology (GJCAT)* **2**(2) 1254-63

AUTHORS

	Anil Kumar Gupta, 06/11/1983, Gorakhpur, India **Current position:** Pursuing Ph.D in Computer Science from Bhagwant University, Ajmer, Rajasthan **University studies:** Master in Computer Application from Indira Gandhi National Open University, Delhi **Scientific interest:** Replicated Real Time Database **Publications:** 1 paper in National conferences
	Dr. Vishnu Swaroop, 01/06/1964, Gorakhpur, India **University studies:** Ph.D Computer Science from Uttar Pradesh Rajarshi Tandon Open University, Allahabad, 2002 **Scientific interest:** Real Time Database **Publications:** 19 paper International and 7 Paper National conference **Experience:** since 14 year as Computer professional in M.M.M University of Technology, Gorakhpur, India