A survey of Commit Protocols in Distributed Real Time database systems

Fadia A. Elbagir[#1], Ahmed Khalid[*2], Khalid Khanfar[#3]

[#1] *PhD. Program in Computer Science, Sudan University of Science and Technology, Sudan*

[#2] *Department of Computer Science, Community College, Najran University, Najran, KSA*

[#3] *Full professor Head of Information Security Department at Naif Arab University for Security, Saudi Arabia*

Abstract— *The commit processing in a Distributed Real Time Database (DRTDBS) can significantly increase execution time of a transaction. Therefore, designing a good commit protocol is important for the DRTDBS; the main challenge is the adaptation of standard commit protocol into the real time database system and so, decreasing the number of missed transaction in the systems. In these papers we review the basic commit protocols and the other protocols depend on it, for enhancing the transaction performance in DRTDBS. We propose a new commit protocol for reducing the number of transaction that missing their deadline.*

Keywords— *DRTDBS, Commit protocols, Commit processing, 2PC protocol, 3PC protocol, Missed Transaction, Abort Transaction***.**

I. Introduction

In Distributed Real-Time Database System (DRTDBS) it is very important to design an efficient commit protocols to grantee transaction atomicity. The commit processing in a DRTDBS can significantly growing the execution time of a transaction [**37**, **32**, **41**]. 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 aborted, in the other side the successful transaction is committed [**45, 3**].

For Reducing unavailability of the data , most of the existing commit protocols allowing a committing cohort to transfer its data to an executing cohort therefore, the system performance will be improved [38, 36].

In Distributed Real time systems, a transaction may decide to commit at some sites while at some other sites it could decide to abort, these resulting in infraction of transaction atomicity, to avoid these problems the commit protocol are used [**45, 38, 39**]. To take control of this problem, distributed database systems use a distributed commit protocol to ensure that all the participating sites accept on the final outcome (commit/abort) of the transaction [**32**, **19**].

A distributed real-time transaction commit is confirming to meet the requirements of both the atomicity and the time constraints. And need commit processing so that transactions executing on them still preserve the Atomicity, Consistency, Isolation and Durability (ACID) property [9].

The rest of this paper is organized as follows: Section II introduces Distributed Commit protocols. Section III Describe differences between 2PC and 3PC protocols. In section IV the Implementation of Commit Protocols in Distributed real time Environment is presented. Section V Describes the proposed commit protocol and section VI concludes the paper.

II-Distributed Commit protocols

A real time distributed computing system has heterogeneously connected computers to resolve a single problem. If the transactions run across different sites, it may commit at one site and may drop at another site, leading to an inconsistent transaction. The transaction in a real time database system has deadlines to process the workloads and it need to process transactions before these deadlines expired [3]. Distributed database systems implement a transaction commit protocol to ensure transaction atomicity. A commit protocol guarantees the uniform of commitment of distributed transaction execution [**24**]. There are two types of commit protocols these are the Two-Phase Commit protocol a blocking protocol and the Three-Phase Commit protocol a non-blocking protocol [**22**, **25**, **36**].

a- Two-Phase Commit protocol

Two Phase Commit (2PC) is the common used protocol in DRTDBMS and most of the exciting protocol based on it [**11**, **15**, 1, **16**, **22**, **17**].

2PC protocol has two phases: In the first phase coordinator add the record ‘begin commit’ in the log and send the messages of ‘Prepare’ to all participants , the Timer start to step into the waiting stage; participant receive the ‘Prepare’ news, if it is ready to commit its own part, it can send the message of ‘Ready’ to coordinator; if it is not ready to commit it due to some reasons, it can send the message of ‘Abort’ to coordinator, and add the message to the log. In the second phase, If all participants answer ‘Ready’ , coordinator send ‘Global Commit’ to all of them, otherwise, send the command of ‘Global Abort’; if time is out, it also send the command of ‘Global Abort’ to participants, add the command to the log.

Participants commit or undo the transactions depend on the command of coordinator, and send the message of ‘Acknowledge’ to coordinator to take in the message to the log. Coordinator gathers the message of ‘Acknowledge’ from all participants, add the message to the log and terminate the transaction**.** Fig.1 show the Two Phase Commit [**42, 26, 31, 28, 7, 2, 5**].

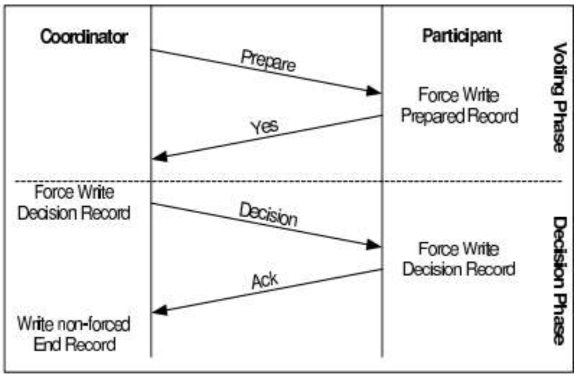

Fig.1 Two Phase Commit Protocol (2PC) Process

There are variant types of 2PC the following are some of these protocols [**32, 24, 29, 10**].

(1) Presumed Abort/Presumed Commit Protocols
(2) One Phase Commit Protocols
(3) Group Commits Protocols
(4) Pre Commit/Optimistic Commit Protocols.

(1) **Presumed Abort/Presumed Commit Protocols**: The Presumed Abort (PA) protocols tries to minimize the message and logging Overheads by requiring all the participants to follow in case of doubt abort rule, that is, if after coming up from a failure a site queries the master about the final outcome of a transaction and finds no information available with the master, the transaction is assumed to have been aborted. suppose that the transactions abort if they are not explicitly committed to reduce the messages such as acknowledgement message from the cohorts to the coordinator and the disk write for the abort log record, while the Presumed Commit (PA**)** protocol is based on notice that, the number of committed transactions is much more than the number of aborted transactions, assume the transactions commit if they are not explicitly aborted. Since transactions usually commit under the normal conditions, it has the advantage if we can skip the messages related to the commit processing. However, it still has the overhead that the coordinator must force-write a “collecting” log record before initiating the commit processing [10, 7, 28, 44, 3].

(2) **One-Phase commits protocol:** Excluded the voting phase of the 2PC, by compel some properties on the cohort’s behavior during the transaction execution. This protocol interfere the voting phase with the execution of transaction and it just has a decision phase. There are two stages the Implicit Yes Voting and the Coordinate Log. This protocol contain fewer overhead therefore it is a simple protocol, It has low latency as it holds less disk spaces, and it is free from bandwidth speed as fewer messages have to be exchanged in it [**36**,14,**8**] .The greatest disadvantage of 1PC it can only handle immediate consistency operation because it lack the voting phase. It does not work on deferred consistency operation [**9**, **16**, **19**].

(3) **Group Commits Protocols:** Many database systems perform an optimized form of commit processing where commit information for a group of transactions is written to disk in one I/O operation, that consumption the cost of the I/O across multiple transactions. So, instead of each transaction write its own commit list to disk, in the group commit one transaction writes to disk a commit list include the commit information for a number of other transactions [**20**].

(4) **Pre-commit/Optimistic commit**: the protocol allows transactions to access uncommitted data carried by prepared transactions in the ‘optimistic’ belief that this data will finally be committed. It reduce the lock difference by releasing the locks earlier, focus on reducing the lock waiting time [**11, 26**].

b- The Three-Phase Commit protocol (3PC):

The three phase commit (3PC) protocol was proposed to address the blocking problem in 2PC. This protocol achieves a non-blocking capability by inserting an extra phase, called the pre-commit phase, between the two phases of the 2PC protocol. In the pre-commit phase, a preliminary decision is reached regarding to the destiny of the transaction. The Three Phase Commit protocol (3PC) performs the operations Prepare phase, Pre-commit phase, Commit/Abort phase [22, 35, 40].

a) Prepare phase

Initially the coordinator will broadcast the Begin-commit request message to all participants and enter into wait state. When, the participant receive the request message, If the participant want to commit the transaction means it respond with the ‘Vote-commit’ message(Yes) to the coordinator and enters into ready state. Otherwise, the participant responds with the Vote-abort message (No) to the coordinator. When the coordinator receives the reply from participant it starts second phase

a) Pre-Commit or Buffering

When the coordinator receives Vote commit message within the time from the participant, the coordinator broadcast the Pre-Commit message to all participants. At this phase introductory decision can be made and it moves to prepared state. When the participant accepts the Pre_commit message acknowledge message will be sent to coordinator. When the Coordinator received ACK message from participant it starts the third phase.

b) Commit/Abort phase

The coordinator decided to commit or abort the transaction and it will inform the participant about the outcome of the transaction. Three-Phase Commit Protocol is problematic only when there are multiple site failures, although it remove the blocking problem, it include an extra overhead of one more cycle and in turn increases time taken for the transaction to complete, However because of high communication overhead 3PC has not been implemented so far [**35**].

III-Difference between 2PC and 3PC Protocols

In the 2PC, the coordinator may abort the transaction globally or resend the global decision; the participant can leave the process blocked until communication with the coordinator is re-established such as sending abort message to the coordinator or invoke the cooperative termination protocol. For 3PC, the coordinator can abort the transaction globally, send global-commit message to the participants or simply send the global decision to all sites that have not acknowledged. The participant can abort a transaction from one side, follow an election protocol, or elect a new coordinator.

IV-Implementation of Commit Protocols in Distributed real time Environment

The design of an efficient commit protocol is very important for distributed real time database systems (DRTDBS), the atomicity property of distributed transactions can only be ensured with the use of an atomic commit protocol, therefore it is very important to choice a better commit protocol for distributed real-time database system (DRTDBS), atomic commit protocols received comprehensive work in the late 1970s till now [**38, 30**].this section introduce the researchers effort for implementation of the Commit Protocols in DRTDBS.

R.Gupta et al (1996) proposed Optimistic Commit Protocol (**OPT**), for designing high performance real-time commit protocols that do not require transaction atomicity requirements, OPT, was designed specifically for the real-time environment and included features such as controlled optimistic access to uncommitted data, active abort and silent kill [**14**]. In 1997 R. Gupta improved OPT and proposed Shadow-Opt and Healthy-OPT protocols, they note that Healthy-OPT provides this high level of performance without incurring the potentially significant overheads associated with implementing the Shadow mechanism in a real system, However, it does not consider the type of dependencies between two transactions [**25**].

Yongik Yoon et al. (1996), proposed a new "protocol Real-time Commit Protocol" (**RCP**). The proved that the RCP satisfies both the correct and the timely completion and produces several desirable effects for fast computing like the elimination of voting phase and the reduction of the number of messages in two phase commit protocol [**42**].

Lam et al. (1997) proposed deadline-driven conflict resolution (**DDCR**) protocol which integrates concurrency control and transaction commitment protocol for firm real time transactions .DDCR resolves different transaction conflicts by maintaining three copies of each modified data item (before, after and further) according to the dependency relationship between the lock-requester and the lock holder. The protocol aims to reduce the impact of a committing transaction on the executing transaction which depends on it. The conflict resolution in DDCR is divided into two parts (a) resolving conflicts at the conflict time; and (b) reversing the commit dependency when a transaction, which depends on a committing transaction, wants to enter the decision phase and its deadline is approaching [**13**].

C Pang, K Lam (1998) proposed an enhancement based on the deadline driven conflict resolution (DDCR) called the Deadline Driven Conflict Resolution with Similarity with similarity (**DDCR-S**) to resolve the executing- committing conflicts in DRTDBS with mixed requirements of criticality and consistency in transactions. In DDCR-S, conflicts involving transactions with looser consistency requirement and the notion of similarity are adopted so that a higher degree of concurrency can be achieved and at the same time the consistency requirements of the transactions can still be met. The simulation results show that the use of DDCR-S can significantly improve the overall system performance as compared with the original DDCR approach [5].

R. Haritsa et al. (1999, 2000) defined the process of transaction commitment and the conditions under which a transaction is said to miss its deadline in a distributed firm real time setting, they proposed and evaluate a new commit protocol **PROMPT** (Permits Reading of Modified Prepared data for Timeliness) for the real time domain to allows transactions to optimistically borrow in a controlled manner, the updated data of transactions currently in their commit phase. The new PROMPT protocol as they explain provided significantly improved performance over the classical commit protocols, however, it does not consider the type of dependencies between two transactions. [**12**, **28**]

R. Haritsa et al. (2000) presented a new one-phase real-time commit protocol, called **PEP**, to address the problem of One-phase commit protocols, which significantly increase the occurrence of priority inversions.

The result of PEP evaluation for real-time applications with firm deadlines demonstrates that, for a variety of environments, it substantially reduces the number of killed transactions as compared to its multi-phase counterparts. They improve that PEP often provides better performance than even an equivalent centralized system. [**27**]

B. Qin and Y. Liu, (2003), proposed an optimistic real-time commit protocol based on PROMPT and DDCR protocols, called double space commit (2SC), which is specifically designed for the high-performance distributed real-time transaction. 2SC allows a non-healthy transaction to lend its held data to the transactions in its commit dependency set. When the prepared transaction aborted, only the transactions in its abort dependency set are aborted while the transactions in its commit dependency set will execute as normal. The two properties of 2SC can reduce the data inaccessibility and the priority inversion that is inherent in distributed real-time commit processing. Extensive simulation experiments have been performed to compare the performance of 2SC with that of other protocols such as PROMPT and DDCR. The simulation results show that 2SC has the best performance. Furthermore, it is easy to incorporate it in any commit protocol [23].

Q.Biao et al. (2003) proposed Optimistic Commit Protocol **2LC**(two-Level Commit),which specially designed for distributed real time domain, it allows transaction to optimistically access the locked data in a controlled manner, which reduces the data an accessibility and priority inversion inherent and undesirable in distributed real time database systems. They used distributed firm – deadline database system model , compared the real time performance of the proposed protocol with others protocols and the simulation results shows that 2LC is effective in reducing the number of missed transaction deadline [**23**].

Inseon Lee et al (2004) evaluated the various distributed commit protocols and proposed a causal commit protocol which suitable for distributed main memory database systems. They performed simulation study to evaluate the performance of proposed protocol and in the result of this simulation they reached that the new protocol greatly reduces the time to commit the distributed transactions without any consistency problem [**10**].

U. Shanker et al. (2006) analyzed all kind of dependencies that may arise due to data access conflicts among executing-committing transactions when a committing cohort is allowed to lend its data to an executing cohort. It then proposes a static two-phase locking and high priority based, write-update type, ideal for fast and timeliness commit protocol "**SWIFT".** They analyzed the performance of SWIFT for partial read-only optimization, which minimizes interstice message traffic, execute-commit conflicts and log writes consequently resulting in a better response time. As they appear these approach reduces the time needed for commit processing and is free from cascaded aborts and Simulation results show that SWIFT improves the system performance in comparison to earlier protocol, However SWIFT is beneficial only if the database is main memory resident and his work is still needed to explore the impact of communication among the cohort and its siblings on overall system performance [**38**].

N. Noual &HDris (2006) analyzed the main features of **2PC** protocol and identified the problems they raise in mobile context. Many papers and there proposed protocols are discussed , provided differences between a traditional distributed system and mobile system and proposed protocols as alternative to 2PC to allow a participant to unilaterally commit a transaction and release resource is hold. The solution proposed for mobile transaction commitment [**18**]

Shishir Kumar & Sonali Barvey(2009) analyzed two phase commit protocols and its variants both on the basis of time and cost. They presented a new commit protocol which is non-blocking (**NBCP**) which survives the coordinator and participant failure and not even increases the cost of execution and time with the help of low cost main memory and can give even better performance in reliable systems where failure rate is not very high [**33**].

S. Agrawal & Udai Shanker (2010) described many protocols for distributed real time database systems (Shadow, Piggy bag, Elemental External Dependency Inversion and in Time Yielding (**SPEEDITY**) protocols. compared performance of proposed commit protocol "SPEEDITY" with shadow PROMPT, SWIFT and DSS-SWIFT commit protocols, Simulation results show that the proposed protocol improves the system performance up to 5% as transaction miss percentage [**30**].

Udai Shanker & Nikhil Agarwal(2010) proposed a modified real time commit protocol for distributed real time database systems (DRTDBS), Allow Commit Dependent and in Time borrowers for Incredible Value added data lending without Extended abort chain (**ACTIVE**), where borrower cohorts are categorized as commit and abort dependent. Further, the commit dependent borrowers can lend data to executing cohorts with still limiting the transaction abort chain to one only and reducing the data inaccessibility the performance of ACTIVE is compared with PROMPT, 2SC and SWIFT protocols for both main memory resident and disk resident databases with and without communication delay. Simulation results show that the proposed protocol improves the system performance up to 4 % as transaction miss percentage [**37**].

Xiai YAN et.al(2012) proposed a protocol adapted to the distributed real-time transaction commit, which can avoid the blocking problem when dealing with transactions by coordinator redundancy. They analyzed 2PC protocol. They proposed modified protocol **RL2PC** adapted to the distributed real-time transaction commit. The result of exponent shows that when the average arrival interval time of transaction is small, the success rate of the improved commit protocol is significantly higher than that of 2PC [**41**].

V-proposed Commit protocol

In our proposed model we will use commit percentage which indicates the percentage of input transaction completed before deadline. And according to the time factor we will tend to consider it as a most important form of the deadline to avoid the unpredictability in the commitment process. Several workload parameters such as number of sites, size of database (i.e. pages in DB), transaction arrival rate/site, CPU page processing time, disk access time are used for the simulation , It is anticipated that the commit and abort percentage of cohorts may lead for designing a new commit protocol based on 2PC protocol.

VI-Conclusion

Designing a good commit protocol is important for the DRTDBS. In this paper, we have reviewed the basic concepts of commit protocol and committing process. We discuss the basic concept of Two Phase Commit (2PC) which is the most of the exciting protocol based on it, and 3PC non-blocking protocol, Also, we have discussed the different implementation of the commit protocols. Finally a commit protocol depends on the commit percentage is proposed.

References

[1] Ahmad Waqas et al., "Transaction Management Techniques And Practices In Current Cloud Computing Environments : A Survey", International Journal of Database Management Systems (IJDMS) Vol.7, No.1, February 2015

[2] Anup A. Dange, Prof. Neha Khatri-Valmik ," Analysis of Scheduling Nested Transactions in Distributed Real-Time Environment", International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014 ISSN 2091-2730.

[3] Bandaru Vishnu Roopini ,"Transaction Management Policy in Distributed Real Time System", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-2, May 2013

[4] Butler Lampson and David Lomet ," A New Presumed Commit Optimization for Two Phase Commit", Proceedings of the 10th VLDB Conference, Dublin, Ireland, 1998

[5] C Pang, K Lam ," On Using Similarity for Resolving Conflicts at Commit in Mixed Distributed Real-time Databases", Proceedings of the 5th International Conference on Real-Time Computing Systems and Applications, 1998.

[6] C. MOHAN et al, "Transaction Management in the R* Distributed Database Management System", ACM Transactions on Database Systems, Vol. 11, No. 4, December 1986, Pages 373-396.

[7] Giuseppe Congiu et al., "One Phase Commit: A Low Overhead Atomic Commitment Protocol for Scalable Metadata Services", 2012 IEEE International Conference on Cluster Computing Workshops, 978-0-7695-4844-9/12 $26.00 © 2012 IEEE DOI 10.1109/ClusterW.2012.16- 9

[8] Gunjan Verma et al ,"Transaction Processing and Management in Distributed Database Systems", IJCST Vol. 2, Issue 3, September 2011 ISSN : 2229-4333(Print) | ISSN : 0976-8491(Online)

[9] Himanshu Dubey et al, "Enhancer- A Time Commit Protocol", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 1, Issue 10, December 2012

[10] Inseon Lee & Heon Y. Yeom,"A Single Phase Distributed Commit Protocol for Main Memory Database Systems", International, IPDPS 2002, - ieeexplore.ieee.org –

[11] Inseon Lee et al ," A New Approach for Distributed Main Memory Database Systems: Causal Commit Protocol", LEE Inseon, P Taesoon - IEICE Transactions on Information, 2004 - search.ieice.org.-

[12] J.R. Haritsa et al. "The PROMPT Real Time Commit Protocol", IEEE Transactions On Parallel And Distributed Systems, Vol. Xx, No. Y, Month 1999

[13] Lam et al, "Resolving executing-committing conflicts in distributed real-time database systems". J. Comput. 42(8), 674–692 (1999), In: Proceedings of the Third IEEE International Conference on Engineering of Complex Computer Systems, Como, Italy, 8–12 September 1997, pp. 49–58 (1997)

[14] LI Taoshen, SONG Qingzhen, "On the Open One-Phase Atomic Commit Protocol", computer Science Applications and Education Vol.3 No.2 November 2013, 2159-8223 /© 2013 ISAEP.

[15] M.S.Khatib & Dr. Mohammad Atique , "An Analysis of Transaction Management in Distributed Real Time Databases: An Overview", (IJITR) International Journal Of Innovative Technology And Research ,Volume No.2, Issue No. 3, April – May 2014, 985 – 990

[16] Maha Abdallah et al, "One Phase Commit Does it makes sense? " ,This work has been partially funded by the CEC under the OpenDREAMS Esprit project n°20843, All Rights Reserved © 2012 IJARCET

[17] Mandeep Kaur & Harpreet Kaur, "Concurrency Control in Distributed Database System", International Journal of Advanced Research in Computer Science and Software Engineering ISSN: 2277 128X, Volume 3, Issue 7, July 2013

[18] Nadia Noual et. al ., "Protocols for committing Mobile Transactions", The International Arab Journal of Information Technology, vol.3 ,No 2, April 2006

[19] Nitesh Kumar et al., "Enhanced c One Phase Commit Protocol in Transaction Management", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-4, September 2013

[20] Peter M. Spiro et al., "Designing an Optimized Transaction Commit Protocol", Digital Technical Journal Vol. 3 No. 1 Winter 1991

[21] Poonam Singh et al, "An Extended Three Phase Commit Protocol for Concurrency Control in Distributed Systems", International Journal of Computer Applications (0975 – 8887) Volume 21– No.10, May 2011

[22] Q. Biao et al., "A commit Strategy for Distributed Real Time transaction", J. computer. Sci. & Technol., Vol 18, No 5, pp.626-631, Sept.2003

[23] QIN Biao, LIU Yun-sheng, "Distributed Real-Time Transaction Commit Processing", 1000-9825/2002/13(08)1395-07 ©2002 Journal of Software, Vol.13, No.8

[24] R .Gupta et al., "Commit processing in distributed real time database systems". In Proc. the 17th IEEE Real- Time Systems Syrup., Oct. 1996, pp.220-22929.

[25] R .Gupta et al.(1997) ,More optimistic about real-time distributed commit processing. In Proc. the 18th IEEE Real-Time Systems Symp., Oct. 1997, pp.123-133.

[26] R .Gupta et al. , "Revisiting Commit processing in distributed database systems", ACM SIGMOD Record, 1997 - dl.acm.org

[27] R. Haritsa & k. Ramamrithamt, "Adding PEP to Real-Time Distributed Commit Processing", 0-7695-0900-2/00 $10.00 *0* 2000 IEEE

[28] R. Haritsa et al., "The PROMPT Real-Time Commit Protocol", IEEE Transactions On Parallel And Distributed Systems, VOL. 11, NO. 2, FEBRUARY 2000

[29] Rabin Kumar Singh et al., "FIVE: A Real-Time Commit Protocol", International Journal of Computer Applications (0975 – 8887) Volume 13– No.5, January 2011

[30] S. Agrawal et al., "SPEEDITY-A Real Time Commit Protocol", ©2010 International Journal of Computer Applications (0975 – 8887) Volume 1 – No. 3

[31] Saud A. Aldarmi, "Real-Time Database Systems: Concepts and Design "Department of Computer Science the University of York. (1998)

[32] Shetan Ram Choudhary et al, "Performance Evaluation of Real Time Database Systems in Distributed Environment", Int.J. Computer Technology & Applications, Vol4 (5), 785-792. ISSN:2229-6093- (Sept-Oct 2013),

[33] Shetan Ram Choudhary&, Dr. C.K. Jha, "Performance Transaction's Assessment Of Real Time Database System In Distributed Environment", International Journal of Engineering Trends and Technology (IJETT) – Volume 4 Issue 9- Sep 2013 - 33

[34] Shishir Kumar&Sonali Barvey, "Non-Blocking Commit Protocol", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.8, August 2009. – 33

[35] Tanuja Shukla & Radha Krishna Rambola, "Perfect Commit Protocol for Distributed Database System: Analysis Review", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 11, November 2015, ISSN: 2277 128X.- 34

[36] Teresa K. Abuya et al., "A Clustering Algorithm in Two-Phase Commit Protocol for Optimizing Distributed Transaction Failure" , International Journal of Computer Science and Mobile Computing IJCSMC, Vol.4 Issue.3, March- 2015, pg. 97-106, ISSN 2320–088X. –

[37] Teresa K. Abuya et al, "An Improved Failure Recovery Algorithm In Two-Phase Commit Protocol For Transaction Atomicity " , Journal of Global Research in Computer Science Journal of Global Research in Computer Science Research Paper, volume 5, No. 12, December 2014

[38] Udai Shanker et al, "ACTIVE-A Real Time Commit Protocol", Wireless Sensor Network, 2, 254-263 doi:10.4236/wsn.2010.23035 Published Online March 2010 (http://www.scirp.org/journal/wsn(2010)

[39] Udai Shanker et al. "SWIFT—A new real time commit protocol" , Distrib Parallel Databases (2006) 20:29–56 DOI 10.1007/s10619-006-8594-8

[40] Udai Shanker et al., "Distributed real time database systems: background and literature review", Distrib Parallel Databases (2008) 23: 127–149 , DOI 10.1007/s10619-008-7024-5

[41] V. Manikandan et al., "An Efficient Non-Blocking Two Phase Commit Protocol for Distributed Transactions", International Journal of Modern Engineering Research (IJMER) www.ijmer.com Vol.2, Issue.3, May-June 2012 pp-788-791 ISSN: 2249-6645

[42] Xiai YAN1,et al , "An Improved Two-phase Commit Protocol Adapted to the Distributed Real-time Transactions", Hunan Police Academy, China(1), Hunan University, China(2), (Electrical Review), ISSN 0033-2097, R. 88 NR 5b/2012

[43] Yongik Yoon et al., "Real- time Commit protocol For Distributed Real-Time Database Systems", 0-8186-7614-0/9$65 .00 0 1996 IEEE

[44] Yousef J. AlHoumaily & Panos K. Chrysanthis, 12PC:" The One/Two Phase Atomic Commit Protocol", SAC'04, March 1417, 2004, Nicosia, Cyprus. Copyright 2004 ACM 1581138121/ 03/04 ...¥ 5.00

[45] Yousef J. Al-Houmaily et al., "Enhancing the performance of presumed commit protocol". In: Proceedings of the ACM Symposium on Applied Computing, San Jose, CA, USA, 28 February–1 March 1997

[46] Yumnam Somananda et al., "Management of missed transactions in a distributed system through Simulation", 978-1-4244-5540-9/10/$26.00 ©2010 IEEE