International Journal of Grid Distribution Computing
Vol.7, No.4 (2014), pp. 113-122
http://dx.doi.org/10.14257/ijgdc.2014.7.4.11

FTCOT: A Fault Tolerant Transaction Commit Protocol with Timeout Constraint for Mobile Environment

Amit Malik[1] and Muzammil Hasan[2]

[1] *M.Tech Student, Department of Computer Science & Engineering, Madan Mohan Malviya University of Technology ,Gorakhpur(U.P.),India*
[2] *Assistant Professor, Department of Computer Science & Engineering, Madan Mohan Malviya University of Technology, Gorakhpur(U.P.),India*
Email:amitrajeev@gmail.com[1],muzammil.mmmec@gmail.com[2]

Abstract

Now day's transaction failure is a common issue in mobile environment. There are various cause of transaction failure in mobile environment, some of them most common are failure of coordinator (normally which is base station in case of mobile environment), disconnection of mobile device, low battery, handoff. This paper proposes the new transaction commit protocol, Fault Tolerant Transaction Commit Protocol with Timeout Constraint (FTCOT).It support coordinator failure, handoff (in case of mobility) which reduce transaction failure. This paper extends the feature of TCOT.

Keywords: *Mobile Transaction, Handoff, Commit Set, Token*

1. Introduction

The transaction is set of read & write operation perform on database which transfer database from consistent state to another. A mobile transaction involve minimum of one mobile host, which initiate transaction, the transaction may not be completely executed on mobile host, so that transaction is fragmented and distributed among database server for execution, final result come back to same mobile unit. This create distributed mode of execution. When a transaction take place atomicity must be maintain, but maintaining atomicity in mobile environment more challenging task then fixed environment.

The challenges include network failure e.g. network partitioning and node disconnection, coordinator failure each of which involve risk of infinite blocking and can lead to higher number of aborts. With this mode of execution it is difficult to ensure ACID properties of transaction. Thus new model are being created for deal this environment and handle coordinator failure. To overcome from these issue various protocol like ICP, UCM, M2PC, TCOT have been designed for mobile environment. Integrated commit protocol for Mobile Network (ICP) is a protocol which handles coordinator failure or network partitioning but the main problem with this protocol in case of coordinator failure it transfers more number of messages as compare to other protocol. To be more reliable and efficient, we have designed a transaction commit protocol for mobile environment which handle coordinator failure in more convenient way as compare to other protocol. In this protocol for handle coordinator failure we use MSC (Part of fixed network). We use MSC for storing token (contain commit set, execution detail of transaction) which is created by coordinator at the time when transaction is initiated.

ISSN: 2005-4262 IJGDC
Copyright ⓒ 2014 SERSC

2. Proposed Model

Fault Tolerant Transaction Commit Protocol with Timeout Constraints is a one phase transaction commit protocol, supports coordinator failure and mobility (handoff). It eliminates the voting phase of 2PC and an extension of TCOT. Fig 1 shows the architecture used with FTCOT consisting of participant (Mobile Host(MH), Fixed Host(FH)),Base Station (BS), Mobile Switching Centre(MSC) ,Coordinator(CO) where the CO is located on the base station, Fault Tolerant System (FTS) Where FTS is located on the MSC in fixed environment, participant database are also situated in fixed environment. A transaction is initiated from MH, which fragment the transaction and calculate its timeout(Execution Timeout & Shipping Timeout) for its part of transaction, then transfer remaining part of transaction to coordinator. Then coordinator executes the transaction and sends decision to MH.

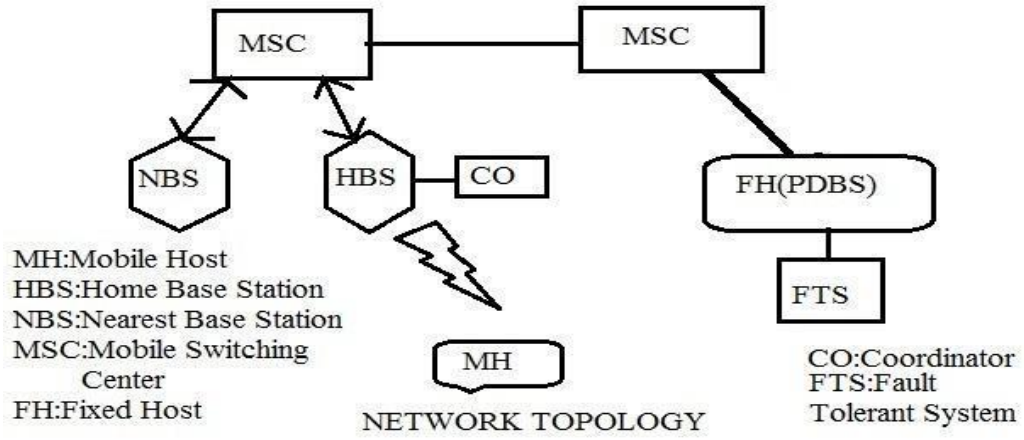

Figure 1. Typical Architecture for FTCOT

In this protocol we use two types of timeout: Execution Timeout and Update Shipping Timeout.

Execution Timeout (Et) defines an upper bound timeout value within which a node of a commit set completes the execution (not commit) of its ei. The value of Et may be node specific. It may depend on the size of ei and the characteristics of the processing unit. We identify H-MU's timeout by Et (MU) and DBS timeout by Et(DBS). The relationship between these two timeouts is Et(MU) = Et(DBS) +Δ or -Δ. The Δ accounts for the characteristics such as poor resources, disconnected state, availability of wireless channel, etc., compared to DBS. Furthermore, the value of a timeout for an ei depends on its MU, thus, Et (MUi) may not be equal to Et (MUj), (i not equal to j). It is possible that a MU may take less time than its Et to execute its ei. We also do not rule out the possibility that in some cases Et(DBS) may be larger than Et (H-MU). Et typically should be just long enough to allow a fragment to successfully finish its entire execution in a normal environment (*i.e.*, no failure of any kind, no message delay, *etc*.)

 Copyright ⓒ 2014 SERSC

Shipping timeout (St) defines the upper bound of the data shipping time from H-MU to DBS. Thus, at the end of Et the CO expects the updates to be shipped to the DBS and logged there within St . We compute St as Time to compose updates (Ut) + Time for the updates to reach CO (Sh).

The FTCOT initiate and fragment transaction (T_i) at MH, the first fragment ei0 is being executed at MH and the remaining fragments of Ti i.e. Ti -ei0 are sent to the Coordinator with execution timeout(E_t) & shipping timeout (S_t),FTS id to coordinator, then coordinator distribute transaction fragment to PDBS(PDBS) and create commit set(which contain all participants details). After receiving their fragment participant calculate their execution timeout and send it to the coordinator after receiving all execution timeout coordinator create a token which contain commit set & execution timeout to commit their part of fragment & also shipping timeout. Now this token is sent to FTS which is used at the time of coordinator failure. At each time when any PDBS or MH extends it timeout then they send their extended timeout to coordinator the coordinator updates the token & then said updated token to fts. After when PDBS or MH complete the processing of their part of fragments they send their decision to coordinator and then after receiving decision from all member of commit set coordinator finally execute the transaction.

2.1. Algorithm For H-MU:-

1. Initiate T_i
2 .Set H-BS as CO, and store its ID
3. Set H-MSC as fault tolerant system
4. Extract its part of e_i from T_i
5. Compute E_t & S_t for its e_i
6. Send E_t , S_t, T_i-e_i to CO
7. Start processing of its e_i
8. If extension of E_t required
9. extend E_t and S_t and send to CO
10 else
11. process e_i
12. If H-MU want to Commit
13. then write update shipped to log
14. update cached copy
15. Send updates to CO and start wait
16. If CO fails
17. then connect to new BS
18. sent FTS id to new BS
19. exchange(H-MU,new BS ID)
20. if abort message is received or E_t, S_t expires
21. then write abort to log
22. start abort procedure
23. else
24 .commit

2.2. Algorithm for CO:-

1. After receiving split T-e(MU), Create commit set
2. Send assignment to participant DBS

Copyright ⓒ 2014 SERSC

3. Receive E_t from all participant DBS
4. Create token write it to log and send to FTS
5. Wait for extension of E_t and S_t from participant
6. if any E_t or S_t is received
7. then update token
8. send extend e_t or S_t to FTS
9. If min E_t or S_t expired or abort message is received
10. then write abort to log
11. send global abort message to all participant
12. else
13. write commit to log
14. commit

2.3 Algorithm for Participant DBS:

1. Receive assignment from CO
2. Write assignment to log
3. Compute E_t & send E_t to CO
4. Start processing of its e_i
5. If need of extension of E_t
6. then extend E_t
7. send it to CO
8. If it receive global abort message
9. then write abort to log
10. start abort procedure
11. If processing of its transaction complete
12.then send decision to CO
13. If decision is commit
14.then write commit to log
15. send commit to CO
16.if decision is abort
17.then write abort to log
18.send abort to CO

2.4. Algorithm For FTS(H-MSC):-

1. Receive token from CO
2. wait
3. If it receive any E_t or S_t from CO
4. then update tokent
5. if new BS ID and H-MU ID received then
6. sent token to New BS

 Copyright ⓒ 2014 SERSC

2.5. Sequence Diagram: FTCOT

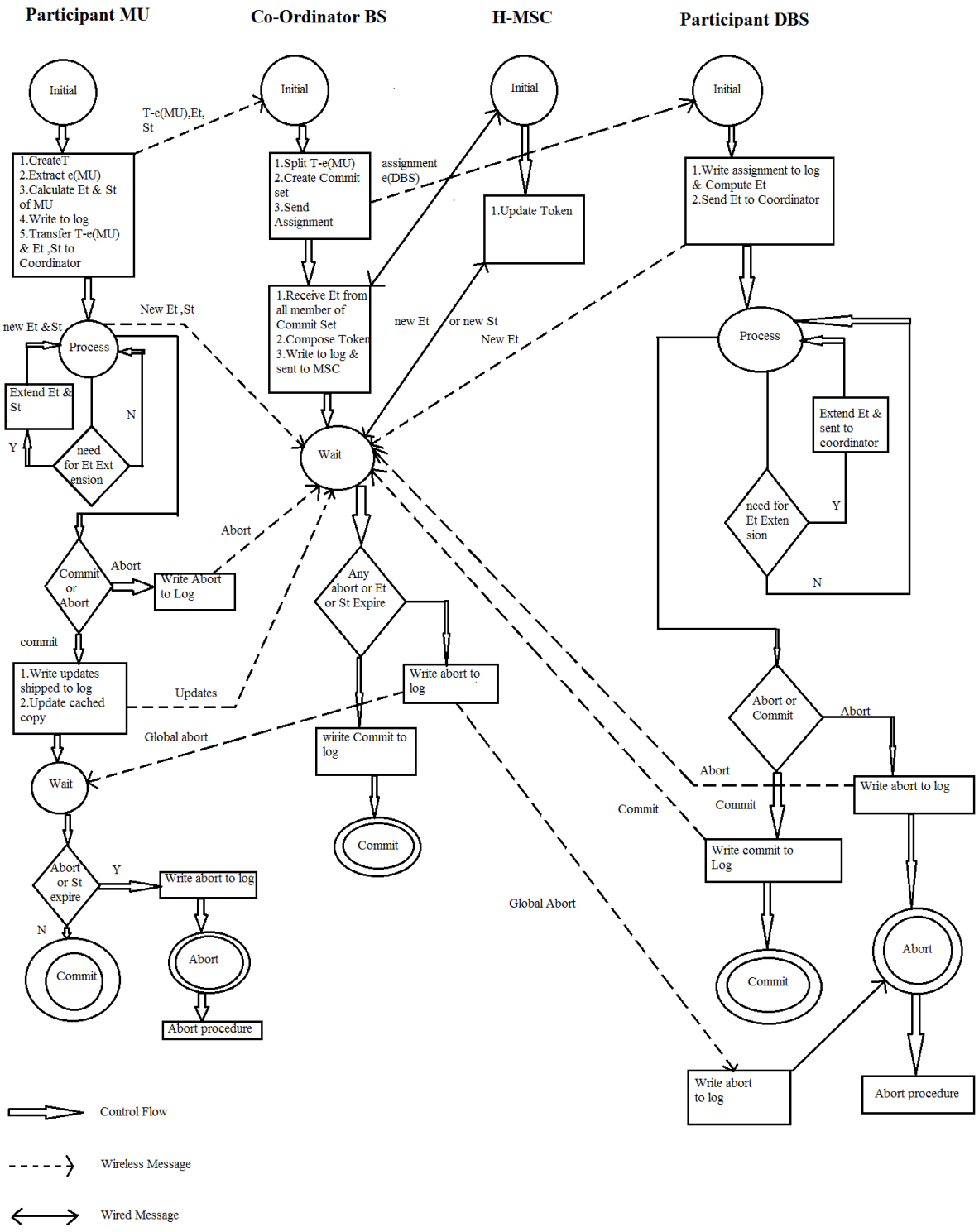

Figure 2. Sequence Diagram

2.6. Rule For increase E_t (Execution Timeout) & S_t (Shipping Timeout):-

1. If H-MU want to increase E_t then it will increase E_t by only initial value of E_t only for 2 times and also increase S_t by initial value of E_t.
2. If participant DBS want to increase its E_t then it can only extend it by initial value of its E_t only for 2 times.

2.7. A **Fault Tolerant:**

In case of failure of coordinator (H-BS) MH will connect to its nearest BS, which now act as a coordinator then new coordinator will communicate to MSC of previous BS and collect the token which was stored at MSC by failed coordinator ,then it send token to MSC by updating it . The process of storing token at MSC introduced ($1+N_{ext}$) wired message in case of not any failure of coordinator.

If coordinator fails then 1 wireless message and 2 wired message are exchanged and total $4+N_{ext}$ message are exchanged.

So it does not make any impact on commit time of our protocol because wired message does not make any impact on the performance of commit time of our protocol.

2.7. **B. Handoff:**

In case of handoff MH is got registered into nearest Base Station now which act as coordinator now in this case new base station exchange token with previous base station and send updated token to coordinator.

2.8 Analysis of Message Transfer:

As can be seen above by the sequence diagram number of wireless message transfer in an execution of transaction are when CO does not fail and handoff not occur

$2+N_{ext}$ (Where N_{ext} is number of extension) (1)

Or if we talk about the wired message introduce due to use of FTS for making system fault tolerant

$1+N_{ext}$ (If coordinator does not fail) (2)

If Coordinator fails then number of wireless message and wired message transfer are

$3+N_{ext}$ (wireless message) (3)

$3+N_{ext}$ (wired message) (4)

In case of handoff then number of total message transfer in the successful execution of a transaction:

$3+N_{ext}$ (wireless message) (5)

$2+N_{ext}$ (wired message) (6)

In case of mobile environment wireless message have impact on the performance of system while wired message does not affect the performance of system so here we concern only wireless message which are same as number of wireless message transfer in case of TCOT.

3. Simulation

The performance of Fault Tolerant Transaction Commit Protocol with Timeout Constraint detailed using simulation mode. The simulator is implemented in .NET platform using c# language. It consist of Mobile Host, Which has Transaction Generator which generate transaction randomly, Transaction Fragment function which fragment transaction randomly.

 Copyright ⓒ 2014 SERSC

It consist a PDBS which executes its parts of transaction fragment and a coordinator which manage execution of transaction.

Transaction are generated by Transaction generator function at MH, at each simulation run a parameter is used to define number of transaction to generate for simulation run. After that transaction is sent to transaction fragment function where transaction is fragmented. After that a fragment is assigned to MH, on the basis of average time to perform read & write operation on MH, It calculate its execution timeout & shipping timeout is calculated. After that remaining fragment transaction with MH timeout send to coordinator. Where coordinator assign fragment to PDBS, then PDBS calculate its timeout on the basis of average time to perform read & write operation on the PDBS & send it to coordinator. Then coordinator create a token which contain commit set & execution timeout to execute each fragment, shipping timeout of MH After creating it send it to FTS. After that a function named parallel processing is called which further handle the execution of transaction.

A summary of the simulation parameter used in simulation model is presented by Table1.

Table 1. Simulation Parameters

01	Mobile host	MH
01	Participant database	PDBS
01	Fault Tolerant System stored at MSC	FTS
40 ms	Average time to perform read operation at MH	T1
60 ms	Average time to perform write operation at MH	T2
30 ms	Average time to perform read operation at FH	T3
50 ms	Average time to perform write operation at FH	T4
50 ms	Average time to send updates to CO	T5
2	Max number of read operation transaction	Max1
12	Max number of write operation in transaction	Max2
1	Minimum number of read operation in transaction	Min1
6	Minimum number of write operation in transaction	Min2
02	Number of transaction fragment	Fragmnet
.010	Disconnection probability of MH	FailureProbMH
.005	Failure Probability of coordinator	FailureProbCO
.005	Failure probability of PDBS	FailureProbPDBS
.02	Probability of increase E_t by MH	Prob1
.02	Probability of increase E_t by PDBS	Prob2
0	Transaction generation time	TrnGn

4. Result &Discussion:

4.1 Number of Message abort due to Coordinator failure:

When we perform simulation on the basis of above parameter, the simulation result shows that number of transaction abort in case of TCOT is more than FTCOT. Figure 3 shows that number of abort in case of TCOT more than FTCOT.

Copyright ⓒ 2014 SERSC

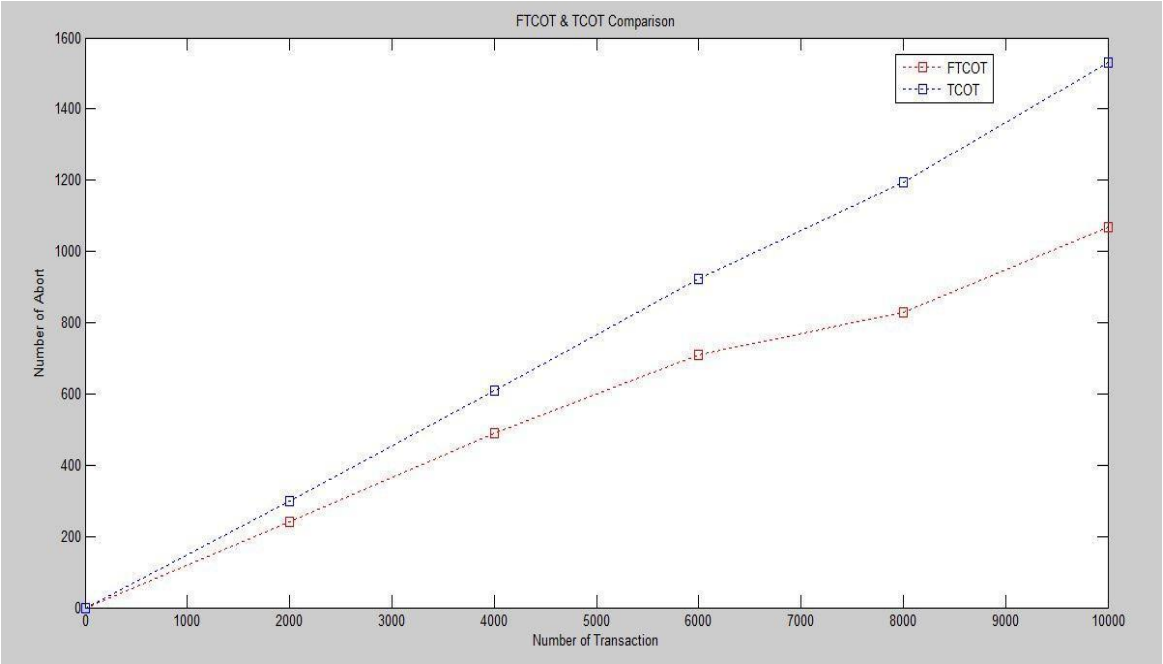

Figure 3. FTCOT & TCOT Comparison

B. Number of Wireless Message Transfer:

Here we compare different protocol on the basis of number of wireless message transfer. We calculate the number of wireless message transfer for the M2PC,UCM & FTCOT by assuming following parameter: Number of nodes taken place in execution is 2. In case of FTCOT probability of Extension of timeout is 0.2 and probability of coordinator failure is 0.1. It is clear from figure 4 number of wireless message transfer in case of FTCOT is less then M2PC & UCM.

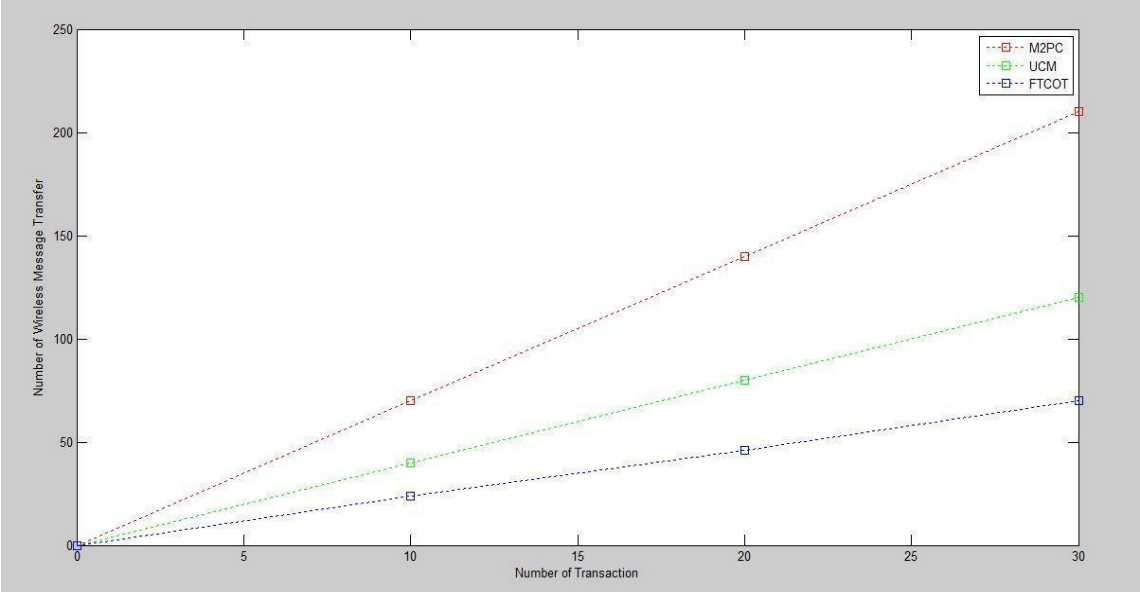

Figure 4. Number of Wireless Message Transfer

 Copyright ⓒ 2014 SERSC

Table 2. Number of Wireless Message Transfer

Number of Wireless Message Transfer/ Number of Transaction	M2PC	UCM	FTCOT
10	70	40	24
20	140	80	46
30	210	120	76

5. Conclusion

This paper presents an atomic commit protocol which is designed for mobile networking usage, because the traditional 2-Phase-Commit protocol is not suitable in mobile networks due to the blocking behavior in case of node or link failures .Our protocol is aimed at handling new challenges including Coordinator failure and Handoff. The FTCOT protocol avoids blocking situations; however it handles disconnection and mobility. Compared to the TCOT protocol, our protocol handle coordinator failure. The FTCOT protocol proposes an extension to the TCOT, where it is able to handle Coordinator failure and mobility. Therefore, we consider our approach to be a useful contribution for performing transactions in mobile environments with one phase, reduced message complexity and supports handoffs and disconnectionsin case of wireless link failures**.**

References

[1] Jim Gray, the Transaction Concepts, "Virtue and Limitation (June 1981), Seventh International Conference on Very Large Database," Tandem Computers Incorporated, Sept 1981, TR 81.3.
[2] Christophe Bobineau, Philippe Pucheral, Maha Abdallah, "A Unilateral Commit Protocol for Mobile and Disconnected Computing," In Proceedings of ther 12th International Conferenceon Parrallel and Distributed Computing System (PDCS), Las Vegas, USA, 2000.
[3] Joos-Hendrik Bose, Stefan Bottcher, Le Grunewald, "An Integrated Commit Protocol for Mobile Network Databases," 9th international database engineering & application symposium (IDEAS 05) in IEEE 2005.
[4] Vijay Kumar, Nitin Prabhu, Maggie Dunhamy, Ayse Yasemin Seydim, "TCOT: A Timout Based Mobile Transaction Commitment Protocol," IEEE Transactions on Computers, 51(10), 2002**.**
[5] Jim Gray and Leslie Lamport, "Consensus on Transaction Commit," CoRR, cs.DC/0408036, 2004.
[6] N.Nouali, A. Doucet, and H. Drias, "A two-phase commit protocol for mobile wireless environment," In H. E.Williams and G. Dobbie, editors,Sixteenth Australasian Database Conference (ADC2005), volume 39 of CRPIT, pages 135–144, Newcastle, Australia, 2005. ACS.
[7] B. Harsoor, S.Ramachandram, "Reliable Timeout Based Commit Protocol," Proceedings of 2nd International Workshop on Trust Management in P2P Systems (IWTMP2PS-2010) CNSA-2010, Springer Verlag 2010, pp. 417-423.
[8] Bharati Harsoor, Dr. S. Ramachandram, "Modified Reliable Timeout base Commit Protocol," 978-1-457 7-0261-7/11, IEEE 2011.

Authors

Amit Malik, obtained his bachelor degree in Information Technology from Anand Engineering College, Agra in 2010. Currently pursuing MTECH in Computer Science & Engineering from Madan Mohan Malviya Engineering College Gorakhpur. His area of interest is Bio informatics, Database Transaction Model, Natural Language processing.

Copyright ⓒ 2014 SERSC

Mr Muzammil hasan, obtained his bachelor degree in Computer Science & Engineering from Madan Mohan Malviya Engineering College in 2002, He also obtained his master degree in Computer Science & Engineering From Madan Mohan Malviya Engineering College. He has 12 year teaching experience at UG & PG level, Currently he is working as Assistant professor in MMMUT. His area of interest is real time database.

Copyright ⓒ 2014 SERSC