

## **New Concurrency Model for Mobile Database Environment**

**Ziyad Tariq Abdul-Mehdi**  
*FIST-MMU*  
*Ziyad.Tariq@mmu.edu.my*

**Ali Bin Mamat & Hamidah Ibrahim**  
*FSKTM-UPM*  
*ali@fsktm.upm.edu.my*  
*hamidah@fsktm.upm.edu.my*

**Mustafa M. Dirs**  
*FITM*  
*College University Technology Tun Hussein Onn*  
*Mustafa@kuiittho.edu.my*

### **Abstract**

Tremendous advances in wireless networks and portable computing devices have led to development of mobile computing. Traditional transaction management in multidatabase systems cannot be applied to mobile environment. In this paper, we analyze the features of mobile Disconnection modes. A new concurrency mobile transaction model on Multi-Check-out Timestamp Order Technique (MCTO) is proposed and a Serializability theory is presented for this model.

**Keywords:** Distributed Database; Replicated system; planned disconnection models; MTCO.

### **1. Introduction**

Mobile database permeates into today's computing and communication area; we envision application infrastructures that will increasingly rely on mobile technology. Current mobility applications tend to have a large central server and use mobile platforms only as caching devices. We want to elevate the role of mobile computers to

first class entities in the sense that they allow the mobile user work/update capabilities independent of a central server. In such an environment, several mobile computers may collectively form the entire distributed system of interest. These mobile computers may communicate together in an ad hoc manner by communicating through networks that are formed on demand. Such communication may occur through wired (fixed) or wireless (ad hoc) networks. At any given time, a subset of the computer collection may connect and would require reliable and dependable access to relevant data of interest. Peer-to-peer (P2P) computing, basically, is an ad hoc network and it can be built on the fixed or wireless network. With P2P, computers can communicate directly and share both data and resources. So far, many applications such as ICQ that makes users exchange the personal messages and, Napster and Freenet that make users exchange music files, have taken the advantage of P2P technology. However, data management is an outstanding issue and is directly to the problem of low data availability. Thus, data availability is the central issue in P2P data management. The most important characteristic that affects data availability in P2P environment is the nature of network. For the case of ad hoc network, hosts are connected to the network temporarily. Furthermore, hosts play also the role of router and they communicate with each other directly without any dedicated hosts. Since there are no dedicated hosts that act as a router, obviously the network connections are prone to get disconnected. Thus, it is difficult to guarantee one-copy serializability since we rely on the mobile hosts, not the fix hosts, in order to communicate with other hosts not reachable directly [1]. When hosts are disconnected more often and the applications have high transaction rates, the deadlock and reconciliation rate will experience a cubic growth [1] and, the database is in an inconsistent state and there is no obvious way to repair eventually. For the case of fixed network, the network connection is relatively stable, but the availability of sufficient computing resource depends on the strategies of replication.

Walborn *et al* [2] describe the use of mobile computers in the trucking industry. Each truck has a computer with a satellite or radio link and interacts with the corporate database. Other applications involving remote or disaster areas and military applications have mobile computers forming ad hoc networks without communications with stationary computers. M.Faiz *et al* [3] discuss the impact of wireless technologies and mobile hosts on a variety of replication strategies. Distributed replicated file systems such as Ficus *et al* [4] have extensive experience with disconnected operations. J. Holiday *et al* [5] such an environment, several mobile hosts may collectively form the entire distributed system of interest. These mobile component hosts are peers and may be replicated both for fault-tolerance, dependability, and to compensate for hosts which are currently disconnected. The problem in this model is arised when every mobile host (MH) is disconnected. So, in this case we will lose all the data updated because no fixed server will update the MH when MH reconnects again. Moreover, the system only allow one MH to have a copy of data object.

In this paper, we consider the distributed database that can make up mobile hosts and peer-to-peer concept. These hosts are peers and may be replicated both for fault-

tolerance, dependability. Thus, we have a distributed replicated database where several sites must participate in the synchronization of transaction. The capabilities of the distributed replicated database are extended to allow mobile hosts to plan disconnect with the capability of update the database on behalf of mobile host by using fixed proxy server to make these update during the mobile disconnection, once a mobile reconnect automatically synchronous and integrated into database.

By using the notion of planned disconnection MTCO, we present a framework to allow the replicated data of mobile host available to access and update in low cost.

## 2. Model

A distributed database in P2P environment consists of a set of data objects stored at different sites (fixed network) and mobile hosts (mobile network) in a computer network. A site (host) may become inaccessible due to site (host) or partitioning failure. No assumptions are made regarding the speed or reliability of the network. Users interact with the database by invoking transaction must appear atomic: a transaction either commits or aborts [6, 7].

In a replicated database, copies of a data object (item ) may be stored at several sites an hosts in the network. Multiple copies of a data object must appear as a single logical data object to the transactions. This is termed as one-copy equivalence and is enforced by the replica control technique. The correctness criteria for replicated database synchronously is serializable [7], which ensures both one-copy Serializability an one copy equivalence, The selection of a quorum is restricted by the quorum intersection property to ensure one-copy equivalence: For any two operations  $o[x]$  and  $o'[x]$  on a data object  $x$ , where at least one of them is a write, the quorum must have a non-empty intersection. The quorum for an operation is defined as a set of copies whose number is sufficient to execute that operation.

The environment has two types of networks, i.e., the fixed network and the mobile network. For the fixed network, all sites are logically organized in the form of two-dimensional grid structure. For example, if the network consists of twenty-five sites, it will logically organize in the form of 5 x 5 grid as shown in Fig. 1. Each site has a master data file. In the remainder of this paper, we assume that replica copies are data files. A site is either operational or failed and the state (operational or failed) of each site is statistically independent to the others. When a site is operational, the copy at the site is available; otherwise it is unavailable.

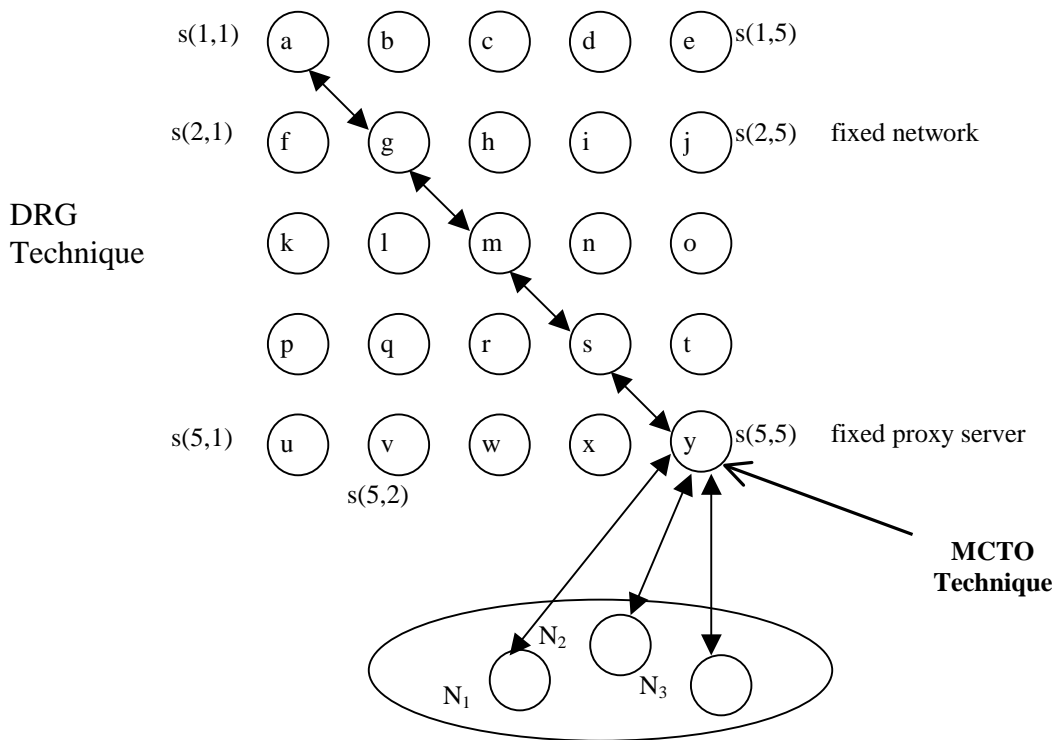
The logical structure for fixed and mobile network is as shown in Fig. 1. The circles in the grid represent the sites under the fixed network environment and  $a, b, \dots$ , and  $y$  represent the master data files located at site  $s(1,1)$ ,  $s(1,2)$ , ..., and  $s(5,5)$  respectively. The circles in the oblong-shape are sites under the mobile network.

For the fixed network, a site  $s$  initiates a DRG transaction [8] to update its data object. For all accessible data objects, a DRG transaction attempts to access a DRG

quorum. If a DRG transaction gets a DRG write quorum without non-empty intersection, it is accepted for execution and completion, otherwise it is rejected. We assume for the read quorum, if two transactions attempt to read a common data object, read operations do not change the values of the data object. Since read and write quorums must intersect and any two DRG quorums must also intersect, then all transaction executions are one-copy serializable.

For the mobile network, a mobile host (MH) will replicate the data asynchronously depending on most visit host. The ‘commonly visited host’ is defined as the most frequent MH that requests the same data at the fixed proxy server (the commonly visited MHs can be given either by a user or selected automatically from a log file/database at each center). This MH will replicate the data asynchronously, therefore it will not be considered for the read and write quorums.

*Definition 2.2.1:* Assume that the mobile environment consists of n of MH. All MHs are labeled  $N_1, N_2, \dots, N_n, 1 \leq i \leq n$ .



**Figure1:** Peer-to-peer environment

*Definition 2.2.2:* Assume the mobile disconnection consists of pre-committed transaction and request transaction for maintain data replicate.

*Definition 2.2.3:* Assume that the limitation amount of data object  $X = \Delta$ , where  $\Delta$  calculate from  $\Delta_x = f(x, N_x) = [((1/2) + (0.01*r)) x/N_x]$ .

Where  $r$  = the number of reconnection of replicated data file and also consider the time by changing the order of data amount,  $N_x$  = number of replication and  $N_x \leq x$ ,  $x$  = value amount of data object  $x$  and  $x > 0$ , We chose  $[1/2]$  to keep some  $x$  amount to request transaction and for new disconnection for MH, pre-committee = transaction process locally at mobile hosts during disconnection depended on the limitation amount of data object  $x$ , Request transaction = transaction process at fixed proxy server because the limit amount of data object  $x$  less than the request.

We are using  $r$  for considering the time by changing the order of data amount, so that, in first disconnection  $r$  will start (0), then  $(0.01 * r)$ ; After the first disconnection any disconnection the fixed proxy server will save less than the half of data object a mount depend on number of  $(r * 0.01)$ , so that the amount of data object at MH will increase depend on number of  $r$  and also the fixed proxy server can allow for multiple MH to disconnect at different time.

As an example, assume a data object,  $X$ , represents the total number of movie tickets and  $N_x$  be the number of replicas of  $X$  among mobile hosts. Initially  $x = 180$  and  $N_x = 3$ .  $X$  is replicated at  $N_1, N_2, N_3$ . The function that defined in 2.2.3 is  $\Delta_x = f(x, N_x) = \lceil ([1/2] + 0.01r)x/N_x \rceil$ , note that keep in first time half of  $X$  amount at fixed proxy server for the request transaction and we are using  $r$  to consider the new MH wish to disconnect and also from  $r$  can considering time by changing the orders amount .

Consider the scenario that  $N_1$  and  $N_2$  wish to disconnect and check-out the data object  $x$  so according to the function in 2.2.3  $\Delta_x = 45$  that mean MH ( $N_1$  and  $N_2$ ) will take privilege to maintain locally update less than or equal to the limitation  $\Delta_x$  and in this case, the MH will make pre-commit and when reconnect to fixed proxy server will serialize with other transactions (if they have) depending to timestamp ordering for each transaction and fixed proxy server will make commit.

If MH ( $N_3$ ) will disconnect with privilege update to the data object  $X$  less than or equal to the new limitation  $\Delta_x = 18$  otherwise will send request transaction.

For the multi check-out in mobile network, if MHs ( $N_1, N_2, N_3$ ) want to disconnect and be able to update the same particular data object, it declares its intention to do so before disconnection and “check-out” or “takes” some amount value.

In order to maintain Serializability in multi check-out mode, we will use timestamp ordering to serialize the mobile transaction at the fixed proxy server. The fixed proxy server will process two type of transactions, pre-committed transaction and request transaction are executed locally at mobile host and serialized on the fixed proxy in the order of their timestamp order arrival, The request transaction is a mobile transaction send the transaction(s) to fixed proxy server to update the amount of data because the amount of data at mobile host less than the request of transaction and the MH cannot update the amount of data locally. So, after the MH reconnect to fixed proxy server will send the request transaction value to fixed proxy server asking for update the data amount if the amount value at fixed proxy server greater than the request transaction in this case the fixed proxy server will update the account and committed the transaction and transfer the committed transaction to mobile host or aborted and send abort transaction to mobile host.

The pre-committed transaction is also kind of mobile transaction but this transaction can update the data amount at mobile host and we call for this case pre-committed transaction and when reconnect to fixed proxy server will transfer the pre-committed transaction to fixed proxy server in order to update it in their amount and send committed transaction to MH, But this kind of transaction is different from request transaction because it can update the data locally.

By using timestamp order, the MH that wishes to disconnect, say,  $N_1, N_2, N_3$ , are acquire a write lock on the same object at different time or same time to update while they are disconnected. The disconnect procedure is a follows:

1.  $N_1, N_2$  and  $N_3$  tell the nearest "fixed proxy server" from the fixed network to check-out the same data object at any point in time.
2. At the same time,  $N_1, N_2$  and  $N_3$  initiate a transaction to obtain locks on the object depending to the limitation amount that give equivalence to each MH participated.
3. If the transaction is successful,  $N_1, N_2$  and  $N_3$  are disconnects with update privileges on the same object. Otherwise,  $N_1, N_2$  and  $N_3$  try again.
4.  $N_1, N_2$  and  $N_3$  can make local transaction and make pre-commit during the limitation amount that given to them, if the transaction greater than limitation, in this case mobile host will send request transaction to fixed proxy server when connected.

In order to preserve correctness, it must be possible to serialize all of the transaction executed by  $N_1, N_2$  and  $N_3$  during disconnection at the point in time of reconnection. This can be done if:

1. The Data object process at fixed proxy server depending to the transaction time arrived.
2. Timestamp ordering serializes all mobile transactions when arrival to fixed proxy server.

This will guarantee Serializability because each transaction at a disconnected MH respects the timestamps ordering.

Since the data file is replicated to only the diagonal sites at fixed network, then it minimize the number of database update operations, misrouted and dropped out calls. Also, sites are autonomous for processing different query or update operation, which consequently reduces the query response time. The proving data files

### 3. Proof of Correctness and Properties

In this section, we prove the correctness of our new technique for requested transactions; multi check-out timestamps order (MCTO), that is, the MCTO technique ensures Serializability for request transaction. In the correctness proof, we prove that all committed queue  $S$  produced by requested transaction MCTO technique are serializable. In other words, the serialization graph,  $SG(S)$ , does not contain any cycle [7]. The  $SG$  is a directed graph  $G = (N, E)$  that consists of a set of hosts  $N = \{T_1, T_2, \dots, T_n\}$  and a set of directed edges  $E = \{e_1, e_2, \dots, e_n\}$ . There is one host in the graph for each transaction  $T_i$  in the queue. Each edge  $e_i$  in the graph is of the form  $(T_i \rightarrow T_j)$ ,  $1 \leq i \leq n$ ,  $1 \leq j \leq n$ , where  $T_i$  is the initial host of  $e_i$  and  $T_j$  is the end host of  $e_i$ .

Such an edge is created if one of the operations in  $T_i$  appears in the queue before some conflicting operation in  $T_j$ .

**Lemma 1:** Let  $T_i$  and  $T_j$  be two committed transactions in a queue  $S$  produced by the MCTO technique. If there is an edge  $T_i \rightarrow T_j$  in  $SG(S)$ , then  $TS(T_i) < TS(T_j)$ .

**Proof:** If there is an edge  $T_i \rightarrow T_j$  in  $SG(S)$ , there must exist one or more conflicting operations of one of the following types on data object  $x$ .

**Case 1:**  $r_i[x] \rightarrow w_j[x]$

In the MCTO technique, the serialization order among transactions may not be the same as the chronological order of transaction commitment,  $T_j$  may commit before  $T_i$  does. Therefore, two possible cases of commit order have to be considered.

**Case 2:**  $w_i[x] \rightarrow r_j[x]$

$T_i$  commits(write) before  $T_j$  reaches fixed proxy server

**Theorem 2:** If  $S$  is a committed queue produced by the MCTO technique, then  $S$  is serializable.

**Proof:** Consider there is an edge in  $SG(S)$ ,  $T_i \rightarrow T_j$ , then there must exist conflicting operations  $p_i[x]$  and  $q_j[x]$  in  $S$ , such that  $p_i[x]$  precedes  $q_j[x]$ . Hence, by Lemma 1,  $TS(T_i) < TS(T_j)$ . If a cycle  $T_1 \rightarrow T_2 \rightarrow \dots \rightarrow T_n \rightarrow T_1$  existed in  $SG(S)$ , then by induction,  $TS(T_1) < TS(T_1)$ . This is a contradiction. Therefore,  $SG(S)$  is not cyclic and thus  $S$  is serializable.

**Lemma 3:** All transactions that are committed by the DRG technique are also committed by the MCTO technique.

**Proof:** Assume a transaction  $T_i$  is committed by the DRG technique but is aborted by the MCTO technique. Let  $RC(T_i)$  be the set of recently committed transactions that are committed between the time when  $T_i$  starts its execution and the time at which it reaches the fixed proxy server phase. Since  $T_i$  is committed by the DRG technique, it must be true that  $RS(T_i) \cap WS(T_{RC}) = \emptyset$  for each transaction  $T_{RC} \equiv RC(T_i)$ . If  $T_i$  is aborted by the MCTO technique, it implies that  $TI(T_i)$  shuts out and  $T_i$  must read some data objects written by at least some transaction  $T_{RC} \equiv RC(T_i)$ . Hence,  $RS(T_i) \cap WS(T_{RC}) \neq \emptyset$ , which is in contradiction to the assumption,  $RS(T_i) \cap WS(T_{RC}) = \emptyset$ , that  $T_i$  is committed by the DRG technique. Thus, the Lemma follows.

**Lemma 4:** All transactions that are aborted by the MCTO technique are also aborted by the DRG technique.

**Proof:** Assume a transaction,  $T_i$  is aborted by the MCTO technique but is committed by the DRG technique. Since  $T_i$  is aborted by the MCTO technique, it must be true that  $TI(T_i)$  shuts out and  $T_i$  read some data objects written by at least some transactions  $T_{RC} \equiv RC(T_i)$ . That is,  $RS(T_i) \cap WS(T_{RC}) \neq \emptyset$  where  $T_{RC} \equiv RC(T_i)$ . If  $T_i$  is committed by the DRG technique, it must be true that  $RS(T_i) \cap WS(T_{RC}) = \emptyset$  for each transaction  $T_{RC} \equiv RC(T_i)$ , which is in contradiction to the assumption that  $T_i$  is aborted by the MCTO technique. Thus, the Lemma follows.

**Theorem 5:** The set of transactions committed by the DRG technique is the subset of transactions committed by the MCTO technique.

**Proof:** The theorem follows directly from the Lemma 3, 4 and Example 1 in which  $T_2$  is aborted by the DRG technique, but is committed by the MCTO technique.

## 6. Conclusion

Transaction management in Mobile Databases has the same behaviors with those in multidatabase systems in many aspects. Many approaches in multidatabase systems can be extended to mobile multidatabase environment. The differences in Mobile Databases are that transactions in Mobile Databases have mobility and long-lived nature. In this paper we have developed a mobile transaction model, which captures data and movement nature of mobile transactions. This model is based on multi check-out. The model describes a mobile transaction Management by Timestamp Order. Based on this mobile transaction model, we have presented a Serializability theory of mobile transactions in Mobile Databases. In addition, the correctness of the technique has discussed.

## References

- [1] Budiarto, S. Noshio, M. Tsukamoto, "Data Management Issues in Mobile and Peer-toPeer Environment", *Data and Knowledge Engineering*, Elsevier, 41 (2002), pp. 183-204.
- [2] D. Walborn and P. K. Chrysanthis. Pro-motion: Management of mobile transactions. In *Proceedings of the 11<sup>th</sup> ACM Symposium on Applied Computing*, 1997.
- [3] M. Faiz and A. Zaslavsky. Database Replica Management Strategies in Multidatabase Systems with Mobile Hosts. In *Proceedings of the 6th International Hong Kong Computer Society Database Workshop*, Mar. 1995.
- [4] P. Reiher, J. Heidemann, D. Ratner, G. Skinner, and G. Popek. Resolving File Conflicts in the Ficus File System. In *Proceedings of the Summer USENIX Conference*, pages 183–195, June 1994.
- [5] J. Holliday, D. Agrawal, and A. El Abbadi. Exploiting Planned Disconnections in Mobile Environments. In *RIDE 2000, Proc. of the 10th Int. Workshop on Research Issues in Data Engineering: Middleware for Mobile Business Applications and E-Commerce*, February 27–28, 2000, San Diego, CA, USA, pages 25–30, IEEE Computer Society Press, 2000.
- [6] B. Bhargava, "Concurrency Control in Database Systems," *IEEE Trans. Knowledge and Data Engineering*, vol 11, no. 1 (1999), pp. 3-16.
- [7] P.A. Bernstein and N. Goodman, "An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases," *ACM Trans. Database Systems*, vol 9, no. 4 (1994), pp. 596-615.
- [8] Ziyad Tariq Abdul-Mehdi, Ali Bin Mamat, Hamidah Ibrahim and Mustafa. M.Dirs, "Check-Out Planned Disconnection Mode with New Transaction Management in Mobile Database", *proceeding of M2USIC 2005 MMU International Symposium on Information and Communication Technologies*. pp. TS12 – 2(5-8).