Mobile Information Systems 2 (2006) 3–19
IOS Press

A novel data replication and management protocol for mobile computing systems

J.H. Abawajy[a], M. Mat Deris[b] and M. Omar[c]
[a]*Deakin University, School of Information Technology, Geelong, VIC, 3217, Australia*
E-mail: jemal@deakin.edu.au
[b]*Tun Hussein Onn University of Technology, Faculty of Information Technology & Multimedia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia*
[c]*University of Malaya, Institute of Science Mathematics, 50603 Kuala Lumpur, Malaysia*

Abstract. Mobile computing has enabled users to seamlessly access databases even when they are on the move. Mobile computing environments require data management approaches that are able to provide complete and highly available access to shared data at any time from any where. In this paper, we propose a novel replicated data protocol for achieving such goal. The proposed scheme replicates data synchronously over stationary sites based on three dimensional grid structure while objects in mobile sites are asynchronously replicated based on commonly visited sites for each user. This combination allows the proposed protocol to operate with less than full connectivity, to easily adapt to changes in group membership and not require all sites to agree to update data objects at any given time, thus giving the technique flexibility in mobile environments. The proposed replication technique is compared with a baseline replication technique and shown to exhibit high availability, fault tolerance and minimal access times of the data and services, which are very important in an environment with low-quality communication links.

1. Introduction

The proliferation of wireless networks and portable computing devices has led to the emergence of the mobile computing paradigm that enables data and resources to be accessed from anywhere and at anytime. As mobile computing devices become more and more common, mobile databases are also becoming popular [2,13,17–19,22]. For example, empowering the travelling salesperson to greet customers at the door and assist them throughout the store with mobile applications to check inventory, access product information, process transactions, and issue receipts or rain checks. Another example is the stock trading application, where brokers might read the prices of multiple stocks from the database for the computation of composite index before they decide to buy any stock.

However, reliable storage of data with concurrent read/write accesses is an ever recurring issue in distributed environments. The problem becomes even more challenging in mobile computing settings, as applications in mobile environments are confronted to limitations imposed by wireless networks and mobile hosts, which includes poor and variable bandwidth, frequent disconnections, high communication prices and limited battery autonomy. These limitations lead to several potential failures that affect data management (e.g., queries, replication, and transactions). For example, it may be hazardous to certain applications such as mobile stock trading where a buy/sell trade will be triggered to exploit the temporary pricing relationships among stocks. Effects of mobile transactions committed during disconnection should be incorporated into the database while guaranteeing data and transaction correctness upon reconnection. Further, the handoff process complicates transaction management, as handoff is generally unpredictable and it may affect the termination (i.e., commit or abort) process of a transaction.

One way to hide this variability from users while providing a responsive and highly available data management service is by replicating service state and user data at multiple locations. The importance of such techniques is increasing as collaboration through wide-area and mobile networks becomes popular [1]. Replication is a critical enabling technology of distributed services, improving both their availability and performance. Availability is improved by allowing access to the data even when some of the replicas are unavailable. Performance improvements concern reduced latency, which improves by letting users access nearby replicas and avoiding remote network access, and increased throughput, by letting multiple computers serve the data. Replication techniques have been widely studied in traditional distributed systems [19–21]. However, the traditional techniques cannot be used directly in mobile database systems as they are built for environments in which communication cost is symmetric, hosts have well known location and no power restrictions on the hosts. In contrast, the inherent characteristics of wireless and mobile environments such as mobility and disconnection make it very difficult to guarantee one-copy serializability [4]. Frequent disconnections coupled with high transaction rates can also lead to deadlocks and reconciliation rate that will experience a cubic growth [4]. The problem of protocol for maintaining replicated data in mobile computing environments has also been widely studied [6–9]. We argue that existing replica management would not cope well with the likely behaviour of mobile clients and a replica management support algorithm that can scalable, fast and have low overhead is still one of the outstanding issues to be resolved.

In this paper, we consider the challenging problem of providing serializability along with good performance and strong consistency guarantees to mobile applications issuing read-write transactions. The key contributions of this paper can be summarized as follows: we present a new replication scheme that replicates data synchronously in a manner of logical three dimensional grid structure on the fixed network while data is asynchronously replicated on mobile network based on commonly visited sites for each user. This combination enables the proposed protocol to operate with less than full connectivity, easily adapt to changes in group membership, and make few assumptions about the underlying network topology. Second, the proposed solution provides fast and scalable implementation of both read and write operations as well as avoids the single point of failure that characterizes primary commit approaches [7]. Third, we describe in detail the consistency levels provided by the proposed protocol and outline formal correctness proofs. The proposed approach has good performance and strong consistency as commitment agreement is accomplished without the need for a plurality quorum of replica servers to be simultaneously accessible, as happens with dynamic voting schemes [8]. Last but not least, the proposed scheme is compared with the baseline replication technique [6] and shown to require less communication cost for an operation as well as providing higher data availability.

The rest of the paper is organized as follows: In Section 2, the background to the problem addressed in this paper is given. Also, we review relevant replica control techniques and show the similarities and differences with the proposed approach. In Section 3, the system model used in this paper is described. In Section 4, the proposed replica management technique is described. We also present the correctness of the proposed replication scheme. In Section 5, we compare the proposed replica approach with the baseline approach [6] and show that the proposed approach requires less communication cost for both read and write operations as well as providing higher data availability. Finally, the conclusion and future directions are given in Section 6.

2. Background and related work

Mobile database is characterized by frequent sharing with requirements of one-copy serialzability [2]. We define a transaction as a sequence of read and write operations on data items in mobile database systems. We distinguish between queries (i.e., read-only transactions) and update transactions. Both types of transactions execute entirely locally. However, queries are light weight in that a query can commit immediately after it successfully finishes its execution. Update transactions, on the other hand, must participate in a distributed commitment process after finishing execution. Two transactions are said to conflict if their common read items have the same version numbers and at least one of the transaction's read items overlaps with the other's update items.

An important feature of mobile database systems is their ability to allow disconnected mobile devices to continue updating local copies of the data items. The key problem to this approach is the reconciliation problem (i.e., the problem of serializing potentially conflicting updates from disconnected clients on all replicas of the database). Reconciliation of conflicting updates is especially critical for disconnected databases where disconnected updates are allowed. Roam [7] employs optimistic replica control mechanism and ensures an eventual convergence to a total order among causally related replica updates. Nevertheless, Roam's consistency protocol does not regard any notion of update commitment, which means that it cannot assert whether the replica values accessed by applications are strongly consistent or not. Haddock-FS [7] employ a primary commit strategy, which centralizes the commitment process in a single distinguished primary replica that establishes a total commit order over the updates it receives. This approach is able to rapidly commit updates, since it suffices for an update to be received by the primary replica to become committed, provided that no conflict is found. However, should the primary replica become unavailable, the commitment progress of updates generated by replicas other than the primary is inevitably halted. Also, it is not scalable as the primary copy can be the hot spot. In contrast, our replication approach is scalable and eliminates the single point of failure that characterizes primary commit approaches [2].

In [6], a replication scheme called Transaction-Level Result-Set Propagation (TLRSP) is discussed. Each fixed and mobile units will store replica of the data. When the data in both mobile and fixed nodes are consistent, a mobile host is said to be operating in consistent state. At each point when the mobile host updates the local replica, it said to be in accumulating state. In this state, the mobile host can also be disconnected while the host continues to access and modify the local copies of the objects. Locally committed transactions are logged at the mobile host. When the mobile host is reconnected to a host in fixed network, it is said to be in reconciliation state. In this state, it is assumed that no transactions are running, instead the mobile host sends the locally committed transactions to the fixed host for conflict detection. The fixed host updates those transactions that passed the validation test and the recently updated copies of the objects are forwarded to the mobile host to refresh its local copies.

The read-one write-all technique is used for managing data replicas in the system. In this technique, a logical read operation on a replicated data item is converted to one physical read operation on any one of its copies, but a logical write operation is translated to physical writes on all of the copies. Although read-one write-all technique provides read operations with high degree of availability at low cost but severely restricts the availability of write operations since they cannot be executed at the failure of any copy. In contrast, we propose a new quorum-based technique for maintaining replicated data that can provide both high data availability and low response time. In addition, our strategy allows transactions to commit with the approval of only a few quorum sites rather than requiring the vote of all sites. Also, our approach does not require all sites to be connected together and 'agree' to update data objects at any given time, thus giving the proposed technique flexibility required in a mobile computing environment.

Several quorum-based protocols have also been proposed for mobile computing environments. Quorum-based protocol is characterized by a read (write) quorum and an intersection requirement between read and writes operations, which eliminates the single point of failure of primary commit approach. A combination of weighted voting, epidemic information flow and versioning strategies is used in Dino [8]. Database states are tracked by associating a version number with each database item. The items in the local copy of the database are modified and their version numbers incremented only when update transactions commit. It requires one entire election round to be completed in order to commit each single update, if only non-commutable updates are considered. This is acceptable when applications are interested in knowing the commitment outcome of each tentatively issued update before issuing the next one.

As noted in [3], in some usage scenarios users and applications are interested in tentatively issuing multiple, causally related tentative updates before acknowledging their commitment. In such situations, the commitment delay imposed by Deno's voting algorithm becomes unacceptably higher than that of a primary commit approach. To address this problem, an epidemic weighted voting algorithm based on version vectors is discussed in [3]. In case where one host is not accessible, the processing of an object is noted in the partial commit state, and resolved it after some time delay. This will increase the response time, which is one of the major performance parameter in replicated systems, and therefore decreases the performance of the system. Also, versioning can lead to heavy reprocessing overhead in many circumstances [6]. Unfortunately, the very construction of these quorums is not a trivial task, as their outcome is strongly subject to membership changes. Also, through the way quorums and in particular their intersections are formed, one can trade fault tolerance (reliability) against protocol efficiency (overhead). In contrast, objects in our approach are replicated at a sub-set of servers. We also use time-based conflict resolutions as opposed to [3] as such eliminating version vector comparison for all but recently changed objects. Our asynchronous implementation is suitable for mobile network since it does not require all sites to be connected together and 'agree' to update data objects at any given time. While on the fixed network, the data objects is replicated synchronously to all sites based on three dimensional grid structure technique. This technique allows transactions to commit with the approval of only a quorum sites rather than requiring the vote of all sites, thus giving the technique flexible in a mobile computing environment.

In summary, the issue of data replication for mobile environments has been addressed by a number of projects, with the common intent of offering high data availability. Basically, there are three main issues in implementation of replication protocols. The first issue is how data is replicated over the nodes (both mobile and fixed) in the network. The second issue is how database consistency is maintained. Third, scalability and reliability are very important properties of the replica management approaches for mobile database systems [6]. However, none of the existing techniqueus discuss how data is replicated both on the fixed and mobile hosts. Also, these approaches do not address the issue of low-cost read operations. In contrast, the proposed replication strategy replicates an object synchronously over stationary sites based on three dimensional grid structure techniques, while for the mobile sites; an object is replicated asynchronously based on commonly visited sites for each user. However, existing protocols are designed primarily to achieve high availability by updating a large fraction of the copies which provides some (although not significant) load sharing. Also, existing approaches achieve high availability and consistency at the expense of higher storage cost and processing cost. Moreover, the behavior of the replica management protocol in the face of multiple replica failures has not been discussed in the literature.

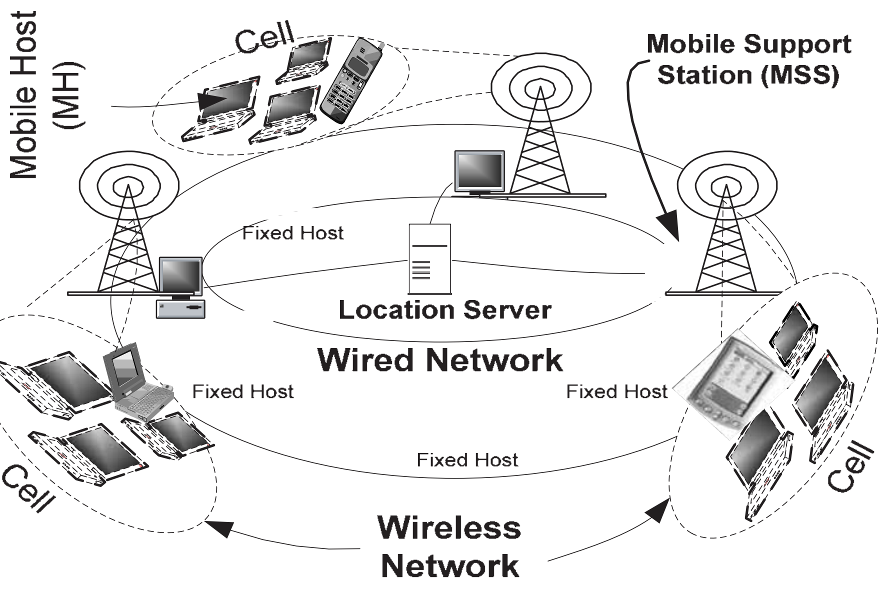

Fig. 1. Architecture of the replicated mobile database model.

3. Mobile computing system model

Figure 1 shows the model of the mobile database system, which is similar to that described in [6,7,14], on top of which we define our replica control and management model. The system consists of two basic components: fixed network component and mobile network component. In this paper, we assume an asynchronous system in which servers can only fail silently and communication links may fail to deliver messages. Combinations of such failures may lead to partitioning failures where sites in a partition may communicate with each other, but no communication can occur between sites in different partitions. Failures are not permanent and can be recovered from. No assumptions are made regarding the speed or reliability of the network.

3.1. Fixed network infrastructure

The fixed infrastructures are used as a backbone for mobile computing system. It consists of Wired Network (WN) and Fixed Host (FH) units. A FH is a computer in the wired network and includes the location server that maintains location information of the mobile hosts and Mobile Support Stations (MSS) units. Fixed hosts can communicate with each other through the fixed network. With the exception of MSS, FHs are not capable of connecting to a mobile unit.

MSSs are are equipped with a wireless interface and capable of connecting with a mobile host that are within its coverage area (a cell). MSSs act as an interface between mobile units and systems of wired-network. Each MSS covers a particular area, called a cell and acts as an interface between mobile computers and fixed hosts. Its responsibilities include keeping track of the execution status of all mobile transactions concurrently executing (or previously executed at this site but not yet committed), logging recovery information, and performing needed checkpointing.

3.2. Mobile network infrastructure

The mobile component consists of Wireless Network (WN) and Mobile Host (MH) units. MHs are portable computers that vary in size, processing power, and memory. A MH is capable of connecting to the fixed network via a wireless link. Mobile units can move within a cell or between cells, effectively disconnecting from one MSS and connecting to another. This process of moving is referred to as a handoff. At any point in time, a MH can be connected to only one MSS.

An involuntary disconnection can occur in a mobile computing environment when there is a temporary impediment to communication. This can be caused by limitations such as short range, inability to operate underground and in steel-framed buildings, or line-of-sight constraints. A voluntary disconnection can occur when a user deliberately operates isolated from a network. This may happen because no networking capability is available at the location of a mobile computer, or to avoid use of the network for cost or power consumption reasons.

4. Three dimensional grid replication technique

In this section, we describe the proposed quorum-based replication protocol that we refer to as a three dimensional grid structure (TDGS). The quorum for an operation is defined as a set of copies whose number is sufficient to execute that operation. The selection of a quorum is restricted by the quorum intersection property to ensure one-copy equivalence (data always in a consistent state): For any two operations o[x] and o'[x] on an object x, where at least one of them is a write, the quorum must have a non-empty intersection.

4.1. Data replication in fixed network

Given N copies of a data object in the system, we logically organize these copies into a box-shape structure with four planes (i.e., $\alpha_1, \alpha_2, \alpha_3$, and α_4) as shown in Fig. 2. Each copy of the object (circles in Fig. 2) is located at x, y, z coordinate ($C_{x,y,z}$) in a given plane (e.g., $C_{0,0,0}$, $C_{0,0,1}$, ..., $C_{l-1,l-1,l-1}$). A pair of copies that can be constructed from a hypotenuse edge in a box-shape structure is called hypotenuse copies. For example, copies $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$, $\{C_{0,0,l-1}, C_{l-1,l-1,0}\}$, $\{C_{0,l-1,l-1}, C_{l-1,0,0}\}$, or $\{C_{l-1,0,l-1}, C_{0,l-1,0}\}$ in Fig. 2 are hypotenuse copies. In the following subsections, we describe how the read and write operations are handled by the three dimensional grid structure protocol.

4.2. Transaction processing in fixed network

In this section, queries are light weight in that a query can commit immediately after it successfully finishes its execution. Update transactions, on the other hand, must participate in a distributed commitment process after finishing execution. Two transactions are said to conflict if their common read items have the same version numbers and at least one of the transaction's read items overlaps with the other's update items. Read operations on an object are executed by acquiring a read quorum that consists of any hypotenuse copies. In Fig. 2, copies $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$, $\{C_{0,0,l-1}, C_{l-1,l-1,0}\}$, $\{C_{0,l-1,l-1}, C_{l-1,0,0}\}$, or $\{C_{l-1,0,l-1}, C_{0,l-1,0}\}$ are hypotenuse copies any one pair of which is sufficient to execute a read operation. Since each pair of them is hypotenuse copies, it is clear that, read operation can be executed if one of them is accessible, thus increasing the fault-tolerance of this protocol.

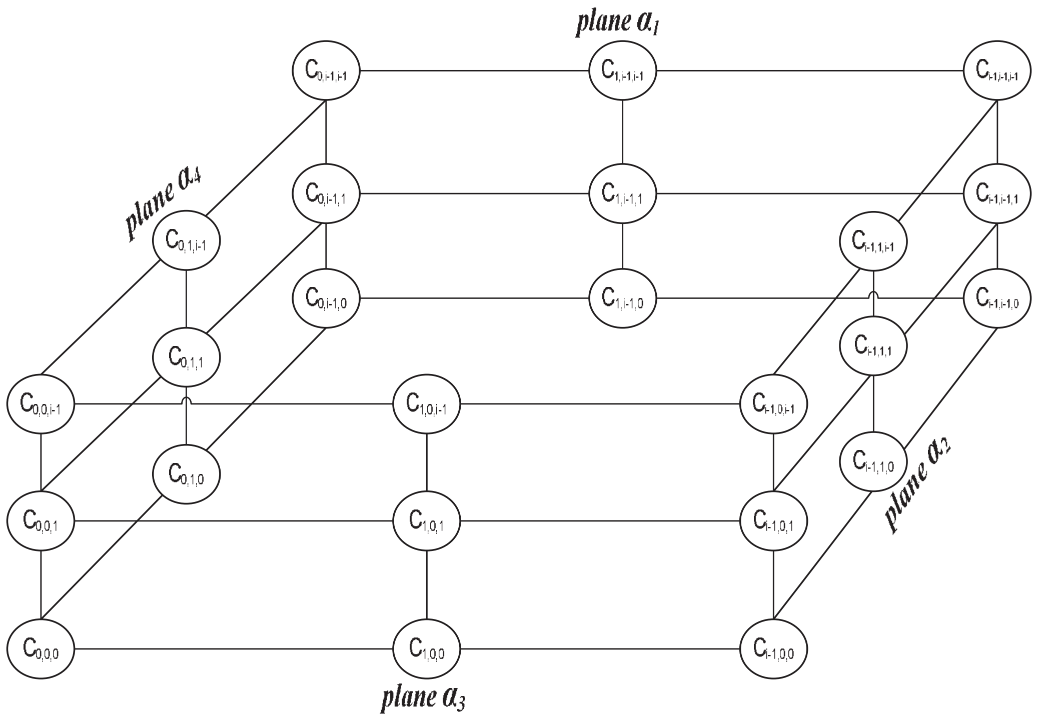

Fig. 2. The organization of the fixed networks with four planes: α_i denotes planes, circles represent a copy of an object replica at location $C_{x,y,z}$.

Note that since read operations do not change the value of the accessed data object, a read quorum does not need to satisfy the intersection property. However, the write operation must satisfy the quorum intersection property to ensure one-copy equivalence. For example, suppose that a site $C_{(i,j,k)}$ initiates a transaction to write its data object. For all accessible data objects, a three dimensional grid structure transaction attempts to access a three dimensional grid structure quorum. Write operations are executed by acquiring a write quorum from any plane that consists of hypotenuse copies and all vertices copies. For example, if the hypotenuse copies, say $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$ are required to execute a read operation, then copies $\{C_{0,0,0}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$ are sufficient to execute a write operation, since one possible set of copies of vertices that correspond to $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$ is $\{C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$. Other possible write quorums are $\{C_{0,0,0}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{l-1,0,l-1}, C_{l-1,0,0}\}$, $\{C_{l-1,l-1,l-1}, C_{0,0,0}, C_{0,0,l-1}, C_{l-1,0,l-1}, C_{l-1,0,0}\}$, $\{C_{l-1,l-1,l-1}, C_{0,0,0}, C_{0,0,l-1}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$, etc. If a three dimensional grid structure transaction gets a three dimensional grid structure write quorum without non-empty intersection, it is accepted for execution and completion, otherwise it is rejected.

We use an example to explain how transactions are managed under the three dimensional grid structure scheme. Assume that there are 16 copies of data A, B, C, D, E, F, G, H, R, S, T, U, V, W, X, and Y in the system (with full replication of data object) as shown in Fig. 3. Table 1 shows the set of read quorum and the set of write quorum for the example of Fig. 3. Let us consider R initiates a write transaction. The execution of the transaction processed as follows: First, R asks the three dimensional grid structure write quorum whether it can be constructed or not. If the three dimensional grid structure

Table 1
Three dimensional grid structure quorums of read and write sets for the example of Fig. 3

Read Quorum Set	Write Quorum Set
{{A, H},{C,F},{G,B},{E,D}}	{{A,H,B,C,D}, {A,H,B,D,F}, {A,H,E,F,G}, {A,H,C,E,G}, {C,F,A,B,D}, {C,F,B,D,H}, {C,F,A,E,G}, {C,F,E,G,H}, {G,B,A,C,D}, {G,B,D,F,H}, {B,G,E,F,H}, {B,G,A,C,E}, {E,D,A,B,C}, {E,D,B,F,H}, {D,E,F,G,H}, {D,E,A,C,G}}

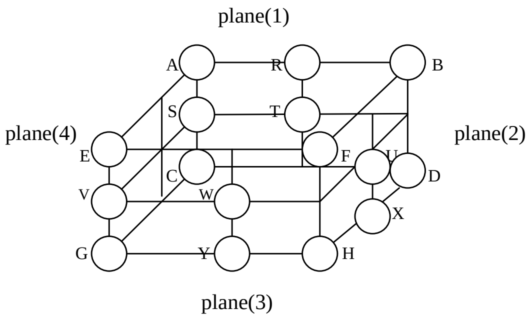

Fig. 3. A three dimensional grid structure organization with 16 copies.

write quorum can be constructed (copies under three dimensional grid structure (TDGS) write quorum return an 'OK' messages for such execution), then R returns 'OK' to the transaction manager. If the transaction manager requests a commit, then in the second step, R asks all copies to commit the execution. If the three dimensional grid structure write quorum cannot be constructed, then R returns a 'FAIL' to the transaction manager and asks all copies to abort the operation in the second phase.

4.3. Data replication in mobile network

An object is replicated asynchronously at only one site based on the most frequently visited site, χ, which is defined as the most frequent site that requested the same data from the fixed network. This site will replicate the data asynchronously, therefore it will not be considered for the read and write quorums. How to adaptively determine χ is itself an interesting problem, which is our on-going follow up work. The basic idea is that χ can be given either by a user or selected automatically from a log file/database at each center.

4.4. Transaction processing in mobile network

A transaction $\boldsymbol{T}$, at each host MHi, executes locally under a local concurrency control mechanism, such as two-phase locking. Each member maintains a vector clock that captures the causal order of precommit transactions. It also keeps a two-dimensional time table as defined in [16], which corresponds to MHi's most recent knowledge of the events at all members. At a point of commit, a precommit record is written to an event log. That record is sent to the nearest fixed site where the three dimensional grid structure quorum technique is applied to update and replicate to the other fixed sites. The precommit record for a transaction from MHi that is stored in the event log, contains the readset, writeset, the values written, and a precommit timestamp. The timing information is used to determine when a given site is aware of a set of events. Based on the information, transactions can be committed or aborted from the event log.

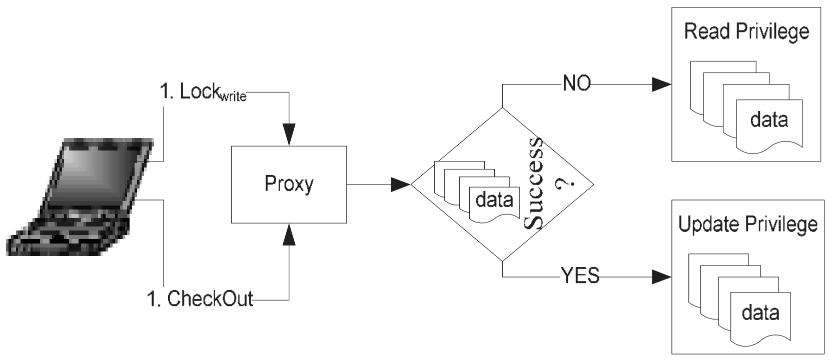

Fig. 4. Check-out process.

If a given host, MH_d, wants to disconnect and be able to write a particular data object, it declares its intention to do so before disconnection and "check-out" (i.e., takes) the object for writing. This can be accomplished by obtaining a lock on the item before disconnection. An object can only be checked out to one site at a time. Let $I(MH_d)$ be a set of objects that MH_d wants to update during disconnection. The general procedure that MH_d follows to acquire a write lock on $I(MH_d)$ during disconnection is shown in Fig. 4. First, MH_d informs the nearest site "proxy" from the fixed network to check-out $I(MH_d)$. At the same time, MH_d initiates a pseudo-transaction to obtain write locks on the items in $I(MH_d)$. If the pseudo-transaction is successful, MH_d disconnects with update privileges on the items in $I(MH_d)$. MH_d has complete and unlimited access to $I(MH_d)$ while other mobile hosts have complete access to the objects other than $I(MH_d)$ while the remaining connected hosts in the system have read/write access on objects other than $I(MH_d)$. The objects not write locked by transaction at disconnect time are treated read-only by the mobile host during disconnect. However, if the pseudo-transaction fails to obtain write locks on the items in $I(MH_d)$, the objects are treated as read-only by MH_d.

When MH_d is reconnected, it contacts the proxy from the fixed network. It then transfers the pre-commit transaction to the fixed proxy server. When the proxy receives the pre-committee transaction, it applies the three dimensional grid structure technique to replicate to the fixed sites. When MH_d receives OK from the proxy, it then releases the corresponding lock on $I(MH_d)$ and clears its log. Finally, the proxy replicates the $I(MH_d)$ objects to the commonly visited sites other than MH_d, that were reconnecting to the fixed network.

We now give an example to illustrate how the proposed scheme works when transaction executions are interleaved while mobile hosts are connected and disconnected from the mobile database. Figure 5 shows an example of transaction execution for MH_d and MH_j mobile hosts respectively. In the figure, t_i denotes a transaction while X_i indicates version of object X written by transaction t_i. The dash line under MH_d denotes transactions performed while the MH_d is disconnected. MDBMS is the mobile database running on the server while LDB is the local database used by each mobile hosts.

In the example of Fig. 5, MH_j remains connected while MH_d is disconnected after acquiring a write lock on data item X with pseudo-transaction. MH_d performs t_1 and t_2 transactions while it is disconnected from the mobile database (MDBMS) and performs t_5 after getting connected to the database. In contrast, MH_j remains connected and executes transaction t_3 and t_4. Notice that MH_d can read any database items while it is disconnected (e.g. Y_0) without getting read locks before disconnection on those items that existed locally at disconnect time. Also note that all of MH_d's transactions will be reading versions of the data items that existed at disconnect time. Finally, MH_d performs t_5 after reconnecting to the server.

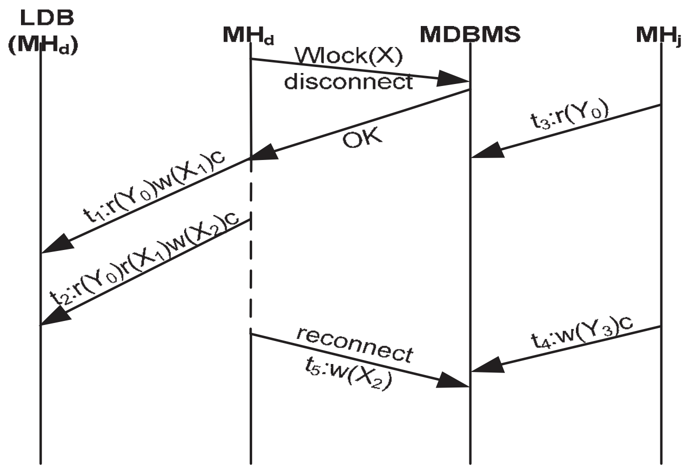

Fig. 5. Check-out mode with mobile read. X_i indicates version of object X written by transaction t_i.

The example of Fig. 5 will guarantee serializability because each transaction at a disconnected host respects two phase locking. First, only the objects that a mobile host successfully write locked before disconnection can be modified while the host is disconnect. Second, the objects which are write locked at disconnect time can neither be read nor written by other site. Finally, the objects not write locked by transaction at disconnect time are treated read-only by the mobile host during disconnect.

4.5. Correctness proof

The correct criterion for replicated database is one-copy serializable. After listing some definitions, we proof that the HRP is serializable. We use the standard serialization graph (SG) approach to show the correctness of HRP.

Definition 1. *Coterie.* Let $\mathbf{U}$ be a set of *groups* that compose the system. A set of *groups* $\mathbf{T}$ is a coterie under $\mathbf{U}$ iff

i) $G \in \mathbf{T}$ implies that $G \neq \varnothing$ and $G \subseteq \mathbf{U}$.
ii) If $G, H \in \mathbf{T}$ then $G \cap H \neq \varnothing$ (intersection property)
iii) There are no $G, H \in \mathbf{T}$ such that $G \subset H$ (minimality).

Definition 2. Let $\mathbf{R}$ be a set of read quorums which consists of *groups* of hypotenuse copies, that is sufficient to execute read operations, and $\mathbf{W}$ be a set of write quorums which consists of *groups* that are sufficient to execute write operations under three dimensional grid structure technique. Then from Fig. 2.

$$\mathbf{R} = \{\{C_{0,0,0}, C_{l-1,l-1,l-1}\}, \{C_{0,0,l-1}, C_{l-1,l-1,0}\}, \{C_{0,l-1,l-1}, C_{l-1,0,0}\}, \{C_{l-1,0,l-1}, C_{0,l-1,0}\}\},$$

and

$$\mathbf{W} = \{\{C_{0,0,0}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\},$$
$$\{C_{0,0,0}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{l-1,0,l-1}, C_{l-1,0,0}\},$$

$\{C_{l-1,l-1,l-1}, C_{0,0,0}, C_{0,0,l-1}, C_{l-1,0,l-1}, C_{l-1,0,0}\}$,

$\{C_{l-1,l-1,l-1}, C_{0,0,0}, C_{0,0,l-1}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$,

$\{C_{l-1,l-1,0}, C_{0,0,l-1}, C_{0,0,0}, C_{l-1,0,l-1}, C_{l-1,0,0}\}$,

$\{C_{l-1,l-1,0}, C_{0,0,l-1}, C_{0,0,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$,

$\{C_{l-1,l-1,l-1}, C_{0,0,0}, C_{0,0,l-1}, C_{l-1,0,l-1}, C_{l-1,0,0}\}$,

$\{C_{l-1,l-1,l-1}, C0,0,0, C_{0,0,l-1}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$,

$\{C_{0,l-1,l-1}, C_{l-1,0,0}, C_{l-1,0,l-1}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}\}$,

$\{C_{0,l-1,l-1}, C_{l-1,0,0}, C_{l-1,0,l-1}, C_{0,0,l-1}, C_{0,0,0}\}$,

$\{C_{l-1,0,0}, C_{0,l-1,l-1}, C_{0,l-1,0}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}\}$,

$\{C_{l-1,0,0}, C_{0,l-1,l-1}, C_{0,l-1,0}, C_{0,0,l-1}, C_{0,0,0}\}$,

$\{C_{0,l-1,0}, C_{l-1,0,l-1}, C_{l-1,0,0}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}\}$,

$\{C_{0,l-1,0}, C_{l-1,0,l-1}, C_{l-1,0,0}, C_{0,0,l-1}, C_{0,0,0}\}$,

$\{C_{l-1,0,l-1}, C_{0,l-1,0}, C_{0,l-1,l-1}, C_{l-1,l-1,l-1}, C_{l-1,l-1,0}\}$,

$\{C_{l-1,0,l-1}, C_{0,l-1,0}, C_{0,l-1,l-1}, C_{0,0,l-1}, C_{0,0,0}\}\}$

By definition of coterie, then **W** is a coterie, because it satisfies all coterie's properties. The next definition will formally define the read and the write quorums.

Definition 3. Let η be a group of hypotenuse copies and ω be a group of copies from any plane that consists of hypotenuse copies and all copies which are vertices as shown in Fig. 2. A set of read quorum, (R_{quorum}) and a set of write quorum ($\mathbf{W}_{\mathrm{quorum}}$) can be defined as follows:

$$\mathbf{R}_{\mathrm{quorum}} = \{\eta_i | \eta_i \cap \eta_j = \varnothing, i \neq j\}, \text{ and}$$
$$\mathbf{W}_{\mathrm{quorum}} = \{\omega_i | \omega_i \cap \omega_j, i \neq j, \text{ and } \omega_i \cap \eta_j \neq \varnothing \text{ for } \eta_j \in \mathbf{R}\}$$

Since read operations do not change the value of the accessed data object, a read quorum does not need to satisfy the intersection property. While a write quorum needs to satisfy read-write and write-write intersection properties.

Definition 4. Serialization Graph

A serialization graph (SG) is a directed graph, G = (N, E), that consists of a set of N = $\{T_1, T_2, \ldots, T_N\}$ sites and a set of E = $\{e_1, e_2, \ldots, e_N\}$ directed edges. There is one site in the graph of the form ($T_i \rightarrow T_j$), $1 \leqslant i \leqslant N$, $1 \leqslant j \leqslant N$, $i \neq j$, where T_i is the initial site of e_i and T_j is the end site of the e_i. Such an edge is created if one of the operations in T_i appears in the schedule before some conflicting operation in T_j. In other words, SG (S), where S is a schedule, does not contain any cycle.

Definition 5. Let T_i and T_j be two committed transactions in a schedule S produced by the Hybrid Replication technique. If there is an edge $T_i \rightarrow T_j$, in SG(S), then timestamp, $TS(T_i) < TS(T_j)$.

There are two possible cases to be considered:

- *Case 1*: T_i read before T_j write at fixed proxy ($r_i[x] \rightarrow w_j[x]$)
  When T_j requests for the write-lock at fixed proxy where T_i is already hold the read-lock, the timestamp for T_j has to be adjusted by adding a sufficiently small value to it. Thus, $TS(T_i) < TS(T_j)$.
- *Case 2*: T_i write before T_j read at fixed proxy ($w_i[x] \rightarrow r_j[x]$)
  In this case, it is clear that T_j will be aborted if T_j requests a read-lock while T_i hold the write-lock to update x. The T_j will obtain a read-lock after T_i release the write-lock. Thus, $TS(T_i) < TS(T_j)$.

Theorem: If S is a committed schedule produced by hybrid replication technique, then S is serializable.

Proof: Consider there is an edge in SG (S), $T_i \rightarrow T_j$, then there is a conflicting operations, $O_i[x]$ and Oj[x] occur in S, such that $O_i[x]$ precedes $O_j[x]$. Hence, by Definition, $TS(T_i) < TS(T_j)$. If a cycle $T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_n \rightarrow T_1$ existed in SG(S), then by induction, $TS(T_1) < TS(T_1)$. This is a contradiction. Since the three dimensional grid structure technique is also serializable, then S produced by HR is serializable.
□

5. Comparative analysis

In this section, we discuss the simulation results of the proposed replica control protocol (i.e., HR) as compared to the primary copy (PC) and the Transaction-Level Result-Set Propagation (TLRSP) protocols. We use cost-based and simulation-based methods are used to compare the two approaches.

5.1. Cost-based performance analysis

In this section we analyse and compare the availability of the HR and TLRSP protocols. We assume that failures are independent and that the probability of a site being up is p. We did not include the PC protocol in this experiment as there is only one copy of the replica.

5.1.1. Transaction processing overhead

Let $C_{TDGS,,R}$ and $C_{TDGS,W}$ be the read and write costs incurred by the three dimensional grid structure (TDGS). £The size of a read quorum in TDGS is hypotenuse copies which is 2 as shown in Fig. 2. Thus, the cost of a read operation in TDGS is, $C_{TDGS,,R} = 2$. The cost of a write operation, $C_{TDGS,,W}$, can be represented as:

$$C_{TDGS,,W} = \ + \alpha_{copies,V} - \alpha_{copies,}$$

where is the hypotenuse copies, $\alpha_{copies,V}$ denote all copies of vertices in a plane; and $\alpha_{copies,}$ is the hypotenuse copy in the same plane. Thus:

$$C_{TDGS,,W} = 2 + (4 - 1) = 5.$$

For example, if hypotenuse copies is $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$, then all copies of vertices in plane α_1 that correspond to $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$ is $\{C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\}$. Therefore,

$$\begin{aligned} C_{TDGS,W} &= |\{C_{0,0,0}, C_{l-1,l-1,l-1}\}| + |\{C_{l-1,l-1,l-1}, C_{l-1,l-1,0}, C_{0,l-1,l-1}, C_{0,l-1,0}\}| \\ &\quad - |\{C_{l-1,l-1,l-1}\}| \\ &= 2 + (4 - 1) = 5 \end{aligned}$$

5.1.2. Fault-tolerance

In TDGS, a write quorum can be constructed as follows: Let $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ be a set of planes in the TDGS technique as shown in Fig. 3. Let i = $\{C_{0,0,0}, C_{l-1,l-1,l-1}\}$, be the hypotenuse copies, then write availability that consists of hypotenuse copies i, W_i, can be represented as:

Probability $\{C_{0,0,0}$ is available$\}$* [φ available]
$+$ Probability $\{C_{l-1,l-1,l-1}$ is available$\}$* [ϕ available]
$-$ Probability $\{C_{l-1,l-1,l-1}$ and $C_{0,0,0}$ are available$\}$* [(φ and ϕ) are available]
where,

$$\varphi = \Omega(\alpha_1) + \Omega(\alpha_2) - \Omega(\alpha_1 \cap \alpha_2),$$

$$\phi = \Omega(\alpha_3) + \Omega(\alpha_4) - \Omega(\alpha_3 \cap \alpha_4),$$

and $\Omega(\alpha_i)$ = Probability of plane α_i available.

Without loss of generality, we assume that a non-vertex copy, say $C_{1,l-1,l-1} \in \alpha_1$ is a primary copy. The probability of α_1 available, $\Omega(\alpha_1)$, can be represented as:

Probability $\{$all copies of vertices from α_1 and primary copy are available$\}$ $+$ Probability$\{$ (all copies of vertices and primary copy $+1$ copy) from α_1 are available$\}$ $+ \ldots +$ Probability $\{$all copies from α_1 are available$\}$

$$= p^5 \sum_{j=0}^{m-5} \binom{m-5}{j} p^j (1-p)^{m-5-j} = p5$$

However, the probability of α_i, i = 2, 3, 4 available, $\Omega(\alpha_i)$, can be represented as:

Probability $\{$all copies of vertices from α_i are available$\}$ $+$ Probability $\{$(all copies of vertices i $+$ 1 replica) from α_i are available$\}$ $+ \ldots +$ Probability $\{$all copies from α_i are available$\}$

$$= p^4 \sum_{j=0}^{m-4} \binom{m-4}{j} p^j (1-p)^{m-4-j} = p4$$

where m is a number of copies in each plane. Thus $\Omega(\alpha_1) = p^5$, and $\Omega(\alpha_i) = p^4$, for i = 2, 3, 4.

Figure 7 shows the availability (vertical axis) while the aliveness of replicas are varied (horizontal axis) when the number of replicas are set to 10 and 16 respectively (i.e., N = 10 and N = 16). As expected, the HR protocol has much higher availability compared to the TLRSP protocol. This is because HR does not have to access all replicas. First, the HR protocol allows us to construct a write quorum even if three out of four planes are unavailable as long as the hypotenuse copies are accessible. To show this, consider the case when only one plane which consists of four copies of vertices and hypotenuse copies are available, e.g., the set $\{C_{l-1,l-1,l-1}, C_{0,0,0}, C_{0,0,l-1}, C_{l-1,0,l-1}, C_{l-1,0,0}\}$ is available as shown in Fig. 2. A HR transaction can be executed successfully by accessing those copies in a HR quorum. Hence the write quorum in HR protocol is formed by accessing those available copies. Read operations, on the other hand, need to access the available hypotenuse copies. Thus the HR protocol enhances the fault-tolerance in write operations compared to the TLRSP protocol. Moreover, HR protocol ensures that read operations have a significantly lower cost, i.e., two copies, and have a high degree of availability, since they are not vulnerable to the failure of more than three quarter of the copies.

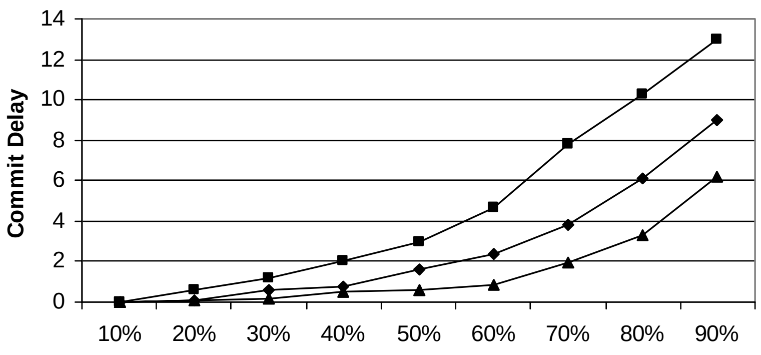

Fig. 6. Commit delays for the three protocols.

5.2. *Simulation-based performance analysis*

The testbed environment used in this paper and the parameters for the system were obtained from the work reported in [3,6,7]. We also used the results reported in [3,6,7] to validated the results reported in this section. We used communication overhead, commit delays and availability as performance metrics to compare the three protocols using a simulator written in C#. A total of 1000 experiments were run for each protocol and the average of the experiments was considered for each performance metric. In the experiments, we generated the disconnection time using a uniform distribution in the range of 1 sec to 2 min. Transaction arrival rates are set to 20/sec.

5.3. *Commit delay*

The commit delay is the time between the initiation and successful completion of the commitment of the transactions. We set the read and write transactions equal (i.e., 50% each) in the experiments described in this section. Also, the number of replicas (only for HR and TLRSP protocols) is set to 10 while the read and update transaction percentages are varied.

Figure 6 shows the commitment delays (vertical axis) as a function of the update transaction percentages (horizontal axis). The results of the experiments show that the TLRSP protocol is inferior to both PC and HR protocols while PC protocol outperforms both HR and TLRSP protocols. This is because there is only one copy in PC protocol whereas there are multiple copies to be consulted in HR and TLRSP protocols. However, HR performs substantially better than the TLRSP protocol as it also consults less replicas compared to the TLRSP.

5.4. *Communication overhead*

In this section we study the message overhead of the different protocols. This is important as replication requires the participating sites to coordinate their activities by exchanging messages. In practice, this can have a significant impact on the overall behavior of the protocol. Figure 7 shows the number of messages as the function of the number of replicas used in HR and TLRSP protocols respectively.

From the figures, it is apparent that HR has the lowest cost for write operation compared to TLRSP protocol. This is because of the fact that in the TLRSP protocol, an update operation needs to access all the replicas of the file in the system. Thus, the communication cost of an update operation in TLRSP protocol is directly proportional to the number of replicas. In contrast, the HR protocol has a better performance in terms of communication costs when compared to TLRSP. This is because it needs 5 copies at most, which results in significantly lower communication costs for comparable data availability.

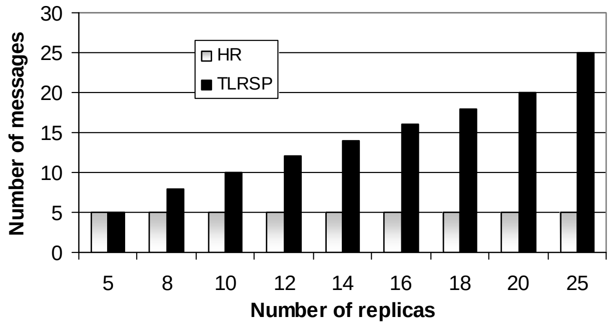

Fig. 7. Communication overhead.

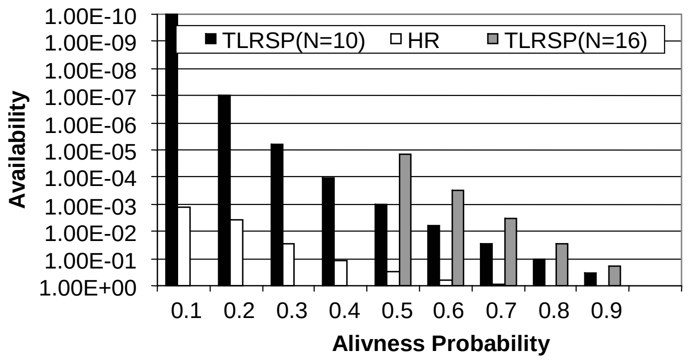

Fig. 8. Comparison of the write availability between HR and TLRSP (N is the number of replicas).

6. Conclusions and future directions

One of the main objectives of mobile computing systems is to provide users with the opportunity to access information and services regardless of their physical location or movement behavior. However, this new infrastructure presents tremendous challenges for data management technology, including huge-scale, variable, and intermittent connectivity; bandwidth, power, and device size limitations. In the presence of frequent disconnection failures, data availability and accessibility from anywhere at any time is not easy to provide. One way to cope with this problem is through data and service replications. To this end, we proposed a new replication technique to manage the data replication for mobile computing environments. The proposed replica management approach offers a number of highly desirable features for applications running in mobile environments. First, it provides high data availability in the face of node and link failure as well as volunteer and involuntary disconnections.

Second, it is also scalable and have low overhead as compared to TLRSP protocol.

Our future work includes fuller implementation of the proposed approach in order to see its complete effectiveness. Also, we plan to study the performance of the proposed replica management system under a non-uniform disconnection time. How to adaptively determine the most frequently visited site is itself an interesting problem, which is part of our on-going follow up work. The transaction size at mobile clients is usually not very large. It is expected that long mobile transactions will be relatively slow and vulnerable to data conflicts with transactions at the server. Nevertheless, we would like to compare the performance of both algorithms by experimenting with a wide range of mobile transaction size.

References

[1] Y. Saito and M. Shapiro, Optimistic replication, *ACM Computing Surveys* (*CSUR*) **37**(1) (2005), 42–81.
[2] P. Serrano-Alvarado, C. Roncancio and M. Adiba, A Survey of Mobile Transactions, *Distributed and Parallel Databases* **16**(2) (2004), 193–230.
[3] J. Barreto and P. Ferreira, Optimistic Consistency with Version Vector Weighted Voting, Technical Report RT/004/2004, INESC-ID/IST, 2004.
[4] J. Holliday, D. Agrawal and A. El. Abbadi, Disconnection Modes for Mobile Databases, *Journal of Mobile Network, Kluwer* **8** (2002), 391–402.
[5] M. Mat Deris, J.H. Abawajy and H.M. Suzuri, An Efficient Replicated Data Access Approach for Large-Scale Distributed Systems, *Proc. IEEE International Symposium on Cluster Computing and the Grid* **2660** (2004), 223–229.
[6] D. Zhiming, M. Xiaofeng and W. Shan, A transactional asynchronous replication scheme for mobile database systems, *Journal of Computer Science and Technology* **17**(4) (2002), 389–396.
[7] J. Barreto and P. Ferreira, *A replicated file system for resource constrained mobile devices*, in Proceedings of IADIS International Conference on Applied Computing, 2004.
[8] U. Cetintemel, P.J. Keleher, B. Bhattacharjee and M. J. Franklin, Deno: A decentralized, peer-to-peer object replication system for mobile and weakly-connected environments, *IEEE Transactions on Computer Systems* (*TOCS*) **52**(7) (2003), 943–959.
[9] P. Reiher, D. Ratner and G. Popek. Roam: a scalable replication system for mobility, *Journal of Mobile Networks and Applications* **9**(5) (2004), 537–544.
[10] P. Serrano-Alvarado, C. Roncancio and M. Adiba, A Survey of Mobile Transactions, *Distribbuted Parallel Databases* **16**(2) (2004), 193–230.
[11] M. Denny and M.J. Franklin, Edison: Database-Supported Synchronization for PDAs, *Distributed and Parallel Databases* **15**(2) (2004), 95–116.
[12] J.M. Milan-Franco, R. Jiménez-Peris, M. Patiño-Martínez and B. Kemme, *Adaptive middleware for data replication*, Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, 2004, 175–194.
[13] S.H. Phatak and B. Nath, Transaction-centric reconciliation in disconnected client-server databases, *Journal of Mobile Networks and Applications* **9**(5) (October 2004), 459–471.
[14] B.S. Nishio and M. Tsukamoto, Data management issues in mobile and peer-to-peer environments, *Data & Knowledge Engineering* **41**(2–3) (June 2002), 183–204.
[15] R. Jiménez-Peris, M. Patiño-Martínez, G. Alonso and B. Kemme, Are quorums an alternative for data replication? *ACM Transactions on Database Systems* (*TODS*) **28**(3) (September 2003), 257–294.
[16] A.B. Waluyo, B. Srinivasan and D. Taniar, Efficient Broadcast Indexing Scheme for Location-Dependent Queries in Multi Channels Wireless Environment, Journal of Interconnection Networks, *Special Issue on Advanced Information Networking: P2P Systems and Distributed Applications* **6**(3) (2005), 303–321.
[17] A.B. Waluyo, B. Srinivasan and D. Taniar, Research in Mobile Database Query Optimization and Processing, *Mobile Information Systems* **1**(4) (2005).
[18] A.B. Waluyo, B. Srinivasan and D. Taniar, Research on Location-Dependent Queries in Mobile Databases, *International Journal of Computer Systems Science & Engineering* **20**(3) (2005), 77–93.
[19] S. Goel, H. Sharda and D. Taniar, Replica Synchronization in Grid Databases, *International Journal of Web and Grid Services* **1**(1) (2005), 87–112.
[20] D. Taniar and J.W. Rahayu, A Taxonomy of Indexing Schemes for Parallel Database Systems, *Distributed and Parallel Databases: An International Journal, Kluwer* **12**(1) (2002), 73–106.
[21] M. Serrano, C. Calero and M. Piattini, An Experimental Replication With Data Warehouse Metrics, *International Journal of Data Warehousing and Mining* **1**(4) (2005), 1–21.
[22] J. Jayaputera and D. Taniar, Query Processing Strategies for Location-Dependent Information Systems, *International Journal of Business Data Communications and Networking* **1**(2) (2005), 17–40.

Dr. J.H. Abawajy is a senior lecturer of computer science at Deakin University, Department of Science and Technology, School of Information Technology. Dr. Abawajy received his PhD in computer science from Ottawa-Carleton Institute of Technology (Canada), a MSc computer science from Dalhousie University (Canada) and BSc computer science from St. F. X university (Canada). His research interests are in the area of high-performance Grid and cluster computing, data management for large scale applications and mobile systems. Dr. Abawajy has guest edited several journals and served as a program committee of numerous international and national conferences. He also chaired several special sessions and workshops in conjunction with international conferences. Dr. Abawajy worked as a software engineer, UNIX systems administrator and database administrator for many years.

Prof. Mustafa Mat Deris received the B.Sc. from University Putra Malaysia, M.Sc. from University of Bradford, England

and Ph.D. from University Putra Malaysia. He has published more than 50 papers in Journals and Conferences. He was appointed as a reviewer of a special issue on International Journal of Parallel and Distributed Databases, Elsevier, 2004, a special issue on International Journal of Cluster Computing, Kluwer, 2004, IEEE conference on Cluster and Grid Computing, held in Chicago, April, 2004, and Malaysian Journal of Computer Science, Technical Committee of International Association of Science and Technology, IASTED, on Database, and Technical Committee of 2nd International Workshops on Grid and Peer-to-Peer Computing, Singapore, May 2005. Currently he is the head of the Department of Computer Science at his university.

Mohammed bin Omar received his PhD in Operational Research from Univ. of Exeter, England in 1998. Currently, he is an Associate Professor at the Department of Mathematics, Faculty of Science, University of Malaya. He has published more than 20 papers in journals and proceedings. His research interests are Mathematical Modeling, Applied Operations Research, and Intelligent Transportation Systems.