Envelope Transaction Mechanism for a Cluster of Peers

doi:10.3991/ijim.v3s2.953

Sushil Kulkarni[1] and Mukund Sanglikar[2]
[1] Jai Hind College, Mumbai, India.
[2] Mithibai College, Mumbai, India.

***Abstract*—A new approach for mobile transaction processing is presented in a cluster of peers. A cluster is a dynamic collection of mobile hosts called peers that are clustered around a single mobile host called a centroid to form a temporary work group for processing and exchanging information. All peers are connected to the centroid using a short range wireless network. All peers and the centroid are members of a peer connected set. The centroid is chosen in such a way that it has a strong connection to the mobile support station. After collecting data from a server, the centroid is free to disconnect from the server and ready to create its own cluster of peers. A dynamically configurable data processing space, called envelope repository, is created to process various transactions at the centroid. The processed data is kept in an envelope at a private work space of the centroid and is brought to the envelope repository whenever required. Peer initiates a jumping transaction to fetch data objects to the centroid. The jumping transaction initiates an envelope transaction to acquire the necessary locks on data objects at envelope repository. After getting the locks, pseudo transaction processes the data objects to give the result stored in an envelope. Envelope pseudo write protocol is designed to make jumping transactions globally serialized.**

***Index Terms*—Envelope repository, Envelope P-write protocol, Expected and average expected cost, Peer connected set.**

I. INTRODUCTION

In the domain of telecommunications, there is an increase in the emergence of portable devices, which makes mobile computing a reality. However, many challenging issues are faced by users to take part in mobile computing while moving in an efficient and quasi-transparent manner. All mobility applications tend to have a large central server and use mobile platforms only as caching devices. We want to increase the role of mobile hosts to allow the mobile user to process the data independently or with servers.

A mobile environment is a geographical territory and is a collection of areas called closed spheres. Wireless communications in each closed sphere is provided by a single low-power transmitter-receiver [4]. There might be some areas in mobile environments in which wireless communication is not available like concrete tunnels. Thus, a mobile environment is a collection of closed spheres that are separated or overlapped with each other. Every closed sphere contains a mobile support station [30] for communication between mobile hosts or fixed database servers. Mobile support stations are connected via a wired (fixed) network. Mobile hosts or database servers with the mobile support stations are connected via wireless networks. Compared to wired networks, wireless networks are characterized by: lower bandwidth, unstable, disconnections, and ad-hoc connectivity [4]. A wireless network does not have the same capacity as a wired network. For example, a wireless network has bandwidth of the order of 10Kbps or a wireless local area network (WLAN) has bandwidth of 10 Mbps [4].

Mobile hosts like laptops, PDAs, and cellular phones are portable mobile computing devices [30] which have the capability to cache and process a limited amount of information. Database servers are stationary computers and are connected via high speed wired-networks, and play roles as permanent data storage repositories.

Mobile hosts move in mobile environments and carry out tasks called mobile tasks. While being in a closed sphere, a mobile host can be either connected or disconnected with the mobile support station of this closed sphere. A set of mobile hosts of a closed sphere are linked together on demand using wireless (ad hoc) networks. At any given time a subset of mobile hosts are disconnected or a new subset of mobile hosts can be added in the ad hoc network [27]. Communication between a set of mobile hosts and a server is through wireless network.

A peer-to-peer (P2P) network is an ad hoc network and can be built on the wireless network. In P2P, server and mobile hosts are called peers. A server and set of mobile hosts is called a peer group. In a peer group, the server is always a member but mobile hosts keep on adding or deleting. Mobile hosts communicate via short-range wireless technologies such as IEEE 802.11, Bluetooth, UMTS, or UWB.

With such communication mechanisms, mobile hosts receive information from their neighbors or from remote objects by multi-hop transmission relayed by intermediate mobile hosts. Peer groups or a cluster of peers can provide services to their member peers but are also accessible by peers of other peer group in the P2P network.

Mobile work in a mobile environment is an active research field [20, 22]. A transaction in a mobile environment is different from distributed databases in many ways [8]. To support mobile computing, the process of transaction should support the limitations of mobile processing, disconnections and power supply. Operations on shared data should guarantee transactions to commit at servers and mobile hosts. Mobile transactions must provide local atomicity to allow the transaction to commit on mobile hosts despite disconnections [12].

In this paper, we propose and design a comprehensive model for a transaction processing system for a cluster of

peers. This system has the ability to support mobile data sharing and cope with the dynamic changes happening in mobile environments.

Section 2 briefly provides a survey of a few transaction processing models. Section 3 discusses the behavior of a mobile host. Section 4 gives dynamic creation of a cluster and its properties related to mobile environment. Section 5 is devoted to explaining a mobile data processing space called the envelope repository. Section 6 and 7 explain mobile transaction processing and the envelope transaction mechanism for a cluster of peers. Section 8 gives the different cost measures of our envelope transaction mechanism and finally we give how it is implemented in Section 9.

II. RELATED WORK

In this section, we will briefly overview some of the previous work done for designing mobile transactions in mobile environment.

The Report and co-transaction model [2] is based on a reporting transaction RT shares its partial results to top level transaction TT by delegating its operations. The delegation process can take place any time during the execution of transaction RT. A co-transaction is a reporting transaction, but it cannot continue executing during the delegation process. This model does not support mobility of mobile host from one cell to another and disconnection is not supported. [6, 10] is nested transaction model, which focuses on disconnected transaction processing with client-server architecture. Top-level transactions are executed at fixed hosts, and sub transactions are executed at mobile hosts. The execution of sub-transactions at the mobile host is archived using the concept of compact object. The model does not discuss mobility of the mobile host. The model requires high capacity resources at mobile hosts and distributed transaction processing is not supported. For each data object, there is a master copy and several replicated copies. There are two types of transactions: Base and Tentative. Base transactions operate on the master copy, while tentative transactions access the replicated copy version. A mobile host can cache either the master or the copy versions of data objects. While the mobile host is disconnected, tentative transactions update replicated versions. The model does not support the mobility of transactions. The Weak-Strict transactions model [14] defines two types of transaction: weak and strict. These transactions are carried out within the clusters that are a collection of connected hosts which are connected via high-speed and reliable networks. In each cluster, data that is semantically related is locally replicated. Mobility of transaction is not discussed in detail in this model. The distributed transaction processing among mobile hosts in a cluster is not discussed. [16, 18] model aims to increase the data availability at mobile hosts. This is achieved by allowing a transaction on a mobile host to submit pre-write operations that write the updated data values, and then issue a pre commit state to the mobile support station. After that, the rest of the mobile transaction can be carried out and finally committed at fixed hosts. A mobile host does not play any role in the execution of the transaction. The molex transaction model [1] is based on top of multi-database and split-join transactions. A Mofix transaction is accompanied with success and failure transaction dependency rules. In the Adaptable Mobile Transaction model [3], a mobile transaction service is proposed to support the adaptability of mobile transaction execution with three-tier client/agent/server architecture. The Kangaroo Transaction Model [5] is designed to capture the movement behavior and the data behavior of transactions when a mobile host moves from one mobile cell to another. This transaction model is built based on the concepts of global and split transactions in a heterogeneous and multi-database environment. The global transaction is split when the mobile host moves from one mobile cell to another and the split transactions are not joined back to the global transaction. The Kangaroo transaction model assumes that the mobile transactions may start and end at different locations.

III. BEHAVIOR OF MOBILE HOSTS

Behavior of mobile host (M) in mobile environment is:

(a) Stable state of M: M is in stable state if jumping condition, $| S - M | < r$, where r is the radius of a closed sphere and M is static or moving inside a closed sphere with no disconnection of M with the mobile support station. A mobile host is said to be in static state either when its movement velocity is zero, or when the location of the mobile host is not considered changing within a period of time. For example, bus stops at a bus-stop to pick up passengers, a salesman is selling products at a shopping centre, or two mobile hosts are always moving close to each other. A mobile host is in a moving state if the velocity is greater than zero or the location of M is moving over a time and change direction of movements of M. For instance, a passenger is travelling by bus; a post man delivers letters in a closed sphere.

(b) Restrictions of M: M possesses restrictions because mobile computers have a limited energy supply, less storage capacity, and limited functionality compared to stationary computers. The storage capacity of a mobile computer (i.e., hard disks or memory) is much less than a stationary computer and is harder to be expanded. Therefore, a mobile host may not be able to store the necessary data that is required for its operations in disconnected mode. If disconnection occurs at M, then the state of M is called isolation state. In this state, the connection to other M is established but not using mobile support station (MSS). Disconnection may occur because of different factors like, M moves out of the wireless communication range, network services are not available, or M is running out of its energy. The isolation state is further refined to an autonomous and idle state. In an autonomous state, M can process the transaction on data objects available at M. If M can not do any operation or if there is a delay the operation, then M is in idle state.

IV. A CLUSTER OF PEERS

A cluster of peers is a dynamic group of mobile hosts that form a temporary workgroup for processing and sharing information as well as support each other. A cluster is not predefined, but contains one or more mobile hosts. A cluster has a centroid, which is a mobile host with a strong connection to server.

The key idea that we propose in this section is to create a cluster of peers using the concept of neighborhood defined not in traditional distance-based approach but using relations like reflexive and symmetric. Secondly, instead of counting the number of peers in a cluster, we use other measure to define the cardinality of neighborhood.

A. Neighbors of stable state mobile hosts

A mobile host is a neighbor of another mobile host if a relation is reflexive and symmetric i.e. p, q are the objects of a set mobile hosts say D and R is a relation such that $\forall p, q \in R$; $(p, p) \in R$ and if $(p, q) \in R$, then $(q, p) \in R$. Then p is a neighbor of q.

Thus, we define the neighborhood of any mobile host as follows:

Definition1. (Neighborhood of q): Let R be a relation on D, which is reflexive and symmetric. For any $p \in D$, there exists q as a mobile host from D such that $(p, q) \in R$. A neighborhood of q is denoted by $N^R(q)$ and given by $N_R(q) = \{p \in D / (p, q) \in R\}$

$N_R(q)$ is also called a cluster of peers with q as centroid and mobile hosts p clustered around q are called peers. The centroid is, in fact, a static mobile host as compared to peers. After the peers become members of a cluster, they may be disconnected from a server. The definition of a cluster in [7] is restricted to the special case of a distance based neighborhood $N_\varepsilon(q) = \{p \in D / |p - q| \leq \varepsilon\}$, where $\varepsilon > 0$. In our context, this definition of $N_\varepsilon(q)$ is not appropriate because of mobility of p. Peers, with the exception of the centroid, can participate in more than one cluster for sharing information. A Peer moves out of a cluster if it is disconnected from a centroid directly or indirectly. This might happen because of a disconnection of wireless network, low battery energy or a peer moving out of communication range of a cluster.

There are different types of peers: (i) peers inside the cluster are called core peers (ii) peers, on the boundary of one cluster but are core peers of another cluster are called boundary peers. (iii) a mobile host not in any cluster is called noise.

Figure1 shows clusters with centroid 1 with 2, 3 and 5 as core peers. On the other hand, 7 is the only core peer of a cluster with centroid 6. Peer 4 is on the boundary of both clusters. Mobile hosts 8 and 9 are the noise for both clusters. We assume that centroids are connected to each other if required.

Before forming a cluster of peers, centroids cache data objects from the server and disconnect from the server. To become a dynamic member of a cluster, a short range wireless channel is used. In other words, peers of a cluster are connected using a P2P communication network with the server as mobile host called a centroid. In continuation, we discuss a method to form $N_R(q)$ using a short range wireless network.

Assume that q is a static mobile host and taken to be a centroid of a cluster $N_R(q)$. Let stime and ctime be starting and completion time of activities by centroid q then there exists a mobile host p which may be in static or isolation state, and wants to join $N_R(q)$ at time t_i such that $stime \leq t_i(p) \leq ctime$. Let r_q be a full range of a centroid to reach mobile hosts and r_p is a full range of mobile hosts wishing to reach centroid or mobile hosts using short range wireless network. Let R be a relation 'reachable from' defined as the maximum of r_p and r_q i.e. $R = \{(p, q) / p, q \in D; \max\{r_p, r_q\}\}$. R is reflexive and symmetric. Therefore, $p \in N_R(q)$ i.e. p is reachable from q and p is a peer of a cluster $N_R(q)$. Following are a few cases:

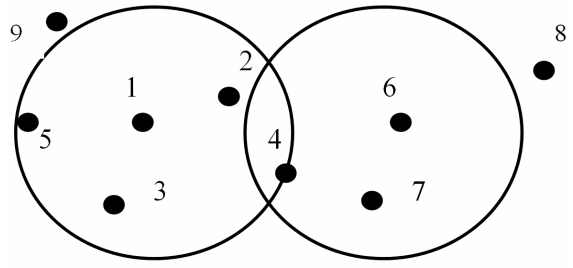


Figure 1. Centroid 1, 6. Core 2, 3, 5, 7. Boundary 4, Noise 8, 9

(i) p is a core peer of $N_R(q)$ iff $\max\{r_p, r_q\} = r_q$ and $stime(q) \leq t_i(p) \leq ctime(q)$. (ii) p is a boundary peer of $N_R(q)$ iff $\max\{r_p, r_q\} = r_p = r_q$; $stime(q) \leq t_i(p) \leq ctime(q)$ and $\max\{r_p, r_s\} = r_s$; $stime(s) \leq t_i(p) \leq ctime(s)$. This means p is on the boundary of $N_R(s)$ but is a core peer of $N^R(s)$. (iii) p is a noise iff $\max\{r_p, r_q\} = r_p$ and $stime(q) \leq t_i(p) \leq ctime(q)$. (iv) If p, s are core peers in $N_R(q)$ then it is not always true that p is a core peer of $N_R(q)$ i.e. $p, s \in N_R(q)$ does not imply $p \in N_R(s)$ as R is not symmetric.

B. Capacity of $N_R(q)$

Capacity is the cardinality of $N_R(q)$ and can be determined by defining a function pCap – peer capacity from a power set D to positive real numbers; pCap: $2^D \rightarrow R^+$ given by pCap(S) =x, where S is the set of centroids and is a subset of D. The value of x can be thought as (i) Number of peers getting services from a centroid at a specific time (ii) Number of peers using a specific service provider or (iii) Number of clusters to be formed. We assume minCap -minimum capacity of a cluster and the centroid condition to form a cluster as follows:

Definition 2. (Centroid condition): Let pcap: $2^D \rightarrow R^+$ be a function given by pCap(S) =x, where S is the set of centroids and is a subset of D. Let minCap be a + ve real number. Then the boolean predicate minPeer is defined to be true iff $pCap(S) \geq minCap$.

For example, if S = {q, q'} with pCap ({q}) = 4 and pCap ({q'}) =6. Let minCap=1 then minPeer ($N^R(q)$) and minPeer ($N_R(q')$) satisfies centroid condition.

C. Peer Connectivity

To connect all peers and centroids of clusters, we define a peer connected set analogous to the definition of density based clusters [9]. The given definition fails as there are two kinds of peers in a peer connected set besides the centroid; the core peer (inside the centroid) and boundary peer (on the fence of the centroid). The value of pCap for the boundary peer is taken to be low as compared to core peer of a cluster so minCap value is to be set low to add as many peers as possible in a peer connected set. All objects of peer connected sets are to be connected directly or indirectly to each other. The following definitions are useful for connecting peers using $N^R(q)$ and minPeer where q is taken as centroid:

Definition 3. (Direct Peer Reachable): A peer p is a direct peer reachable from another peer q with respect to (R, minPeer) if (i) $p \varepsilon N_R(q)$, (ii) minPeer ($N_R(q)$) = true.

Obviously, a direct peer reachable is symmetric for all core peers or boundary peers from the centroid. In general, however it is not symmetric. For instance two peers of the

same cluster need not be direct peer reachable from each other.

Definition 4. (Peer Reachable): A peer p is a peer reachable from another peer q with respect to (R, min-Peer) if there is a chain of peers $p_1, p_2, \ldots p_n$,where $p_1 = q$ and $p_n = p$ such that for all i = 1 to n, p_{i+1} is directly peer reachable from p_i with respect to (R, minPeer)

This definition is an extension of direct peer reachability. The easiest way to have peer reachability between two core peers is by connecting them with a centroid. Peer reachable is transitive but not symmetric, in general.

Both the definitions are related to a single cluster and all peers are reachable from one another. On the other hand, the following definition gives the connectivity between peers from different clusters (adjacent clusters)

Definition 5 (Peer Connected): Peer p_1 of cluster C_1 is peer connected to another peer p_2 of adjacent cluster C_2 with respect to (R, minPeer) if there is a peer (Core or boundary) O such that p_1 is peer reachable from O and O is peer reachable from p_2 with respect to (R, minPeer)

Obviously, Peer connected is reflexive, symmetric and transitive. Thus, finally the peer connected set is the set of peers which may be peer connected or reachable.

Definition 6 (Peer Connected Set): A peer connected set $C \subseteq D$ with respect to (R, minPeer) satisfies following conditions:

1. Connectivity: $\forall p,q \in D$; p is peer connected to q with respect to (R, minPeer).

2. Reachability: $\forall p,q \in C$; if $q \in C$ and p is peer reachable from q with respect to (R, minPeer) then $p \in C$.

Note that C contains at least one core peer as C contains at least one peer p; which is peer reachable to it self. Following are the advantages of making a cluster of peers:

(i) A mobile host with a strong connection with MSS can initiate cluster formation. A number of mobile hosts can join or leave as a peer of a cluster at any given time depending on the completion of sharing activities.

(ii) Mobile computing [29] must handle the inherent characteristics of mobile environments [11]. In particular, mobile applications have to face periods of disconnection that may arise due to economic factors, unavailable connectivity or the application model. To allow mobile users to continue their work even in these periods, a cluster is useful. A disconnected mobile user, who is peer from a cluster, can be connected to centroid using peer connectedness.

Now, we propose an environment at the centroid to process various transactions called envelope repository.

V. ENVELOPE REPOSITORY

The Envelope Repository (ER) is a dynamically configurable mobile processing space located at the centroid of a cluster of peers. This mobile processing space provides various envelopes to be processed using transactions while peers are moving. An envelope is a container having a virtual list of data objects obtained by processing a transaction. Envelopes are stored in the private work space [21, 23, 25] of a centroid. When a centroid reconnects to the data base server, envelopes are propagated into the server. Information at the envelopes is temporary inconsistent at the ER. We assume in this paper that the ER at the centroid is capable of sharing and processing envelopes using transactions but the ER at each peer can process only information located at that peer and send the corresponding envelope to the ER of the centroid for further evaluation. Following are the advantages of envelopes:

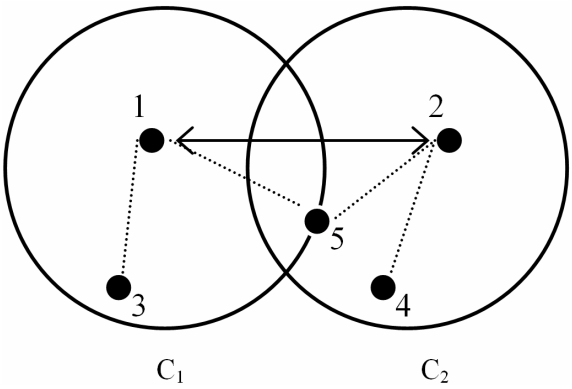


Figure 2.

(a) Sharing dynamic data by mobile hosts: An envelope is created when more than one transaction is processing data objects available in an envelope. A jumping transaction JT_1 initiated by peer p_1 creates an envelope at the ER of the centroid and reaches a synchronous point, before another jumping transaction JT_2 initiated by p_2 otherwise JT_2 can create an envelope and other jumping transactions can join the ER to process an envelope. After all jumping transactions are processed successfully the envelope is destroyed but the ER is not disturbed.

(b) Persistent temporary ER: Envelopes are persistent temporary storage for sharing data objects from private work spaces and are integrated into the server when the centroid re-connects to the server. Jumping transactions are changes the state of data objects at the centroid. When the centroid reconnects to the server, the state of data objects at the server is modified. The envelope has partially committed results obtained from the jumping transaction.

(c) Envelopes are distributed for processing jumping transactions: Envelopes are created and distributed for centroids for processing transactions. For example, 1 and 2 are centroids of two clusters C_1 and C_2. Assume that 3 and 4 are core peers and 5 is boundary peer. All are peers connected to each other. ENV_2 at the ER_2 of C_2 is entirely used by 5 or distributed by 4 and 5 of C_2. As 4 is peer connected to centroid 1 of C_1, 4 can process ENV_1 of C_1 at envelope repository ER_1. [See figure 2]

(d) Low battery of the centroid: Low battery of the centroid may cause disconnections to peers of a cluster. In such a situation, a centroid can be shifted to another mobile host called a proxy peer of a cluster and the ER is reallocated to a new centroid so that envelopes can still be processed. A candidate centroid must satisfy the following conditions: (i) The candidate centroid must have strong and stable connectivity. (ii) The distance between the candidate centroid and centroid is less than the distance between peers and the centroid (iii) All peers are able to connect to the candidate centroid using a short range wireless network.

VI. MOBILE TRANSACTION PROCESSING

Transactions in mobile environments are used to process data objects with protections. They also allow access and modify single or multiple data objects as single atomic operations. A transaction in traditional data bases

cannot be considered as flat transactions as they do not allow partial results to be committed or aborted. Such a situation arises many times because of various limitations of mobile computing [29] and the long transaction. Hence, if the process terminates during transaction processing, everything is to be restored to the point just before the transaction started. To overcome this problem, a mobile nested transaction can be considered.

A mobile nested transaction is constructed from a number of sub transactions. Each sub transaction is called a jumping transaction. The top level jumping transaction may have children that run in parallel with one another on different centroids. Each of these children may also execute one or more jumping transactions of its own children. Jumping transactions are executed at the centroid where location data exists. All jumping transactions are connected to each other to form a tree-like structure called Directed Parse Tree [13, 15, 31]. Nodes in Directed Parse Tree (DPT) correspond to execution of operations at a particular level of abstraction in a layered system. The DPT has the same height, which is equal to the number of levels in the underlying system architecture. The edges in DPT represent the implementation of an operation that is invoked at level L(i) by a sequence of operation executions at the next low level L(i+1) (for i=0 to n-1 in bottom up order). Execution of jumping transactions (JT) can be location dependent; i.e. at level L(i), two JTs may be invoked on different data objects or at L(i) and L(i+1), JTs can be executed on same data objects but initiated from different location. Jumping transactions are executed sequentially in an envelope repository like JT_1, [JT_2: JT_{21}, JT_{22}, JT_{23}], [JT_3: JT_{31}, JT_{32}, JT_{33}], JT_4. Where JT_1 and JT_4 are executing on a data object from an envelope but JT_2 and JT_3 are operating on data objects from different envelopes.

Each node of a DPT corresponds to a jumping transaction and has the following components:

(i) Order of processing is denoted by the JT_i

(ii) Jumping action gives unary or binary action like retrieval, updating, deletion, joining, etc.

(iii) An identification number of a centroid of a cluster where jumping transaction is processed. Initially there is no identification number in a DPT. Values are filled when jumping transactions are executed.

(iv) A pointer is divided into two parts, right (y) and left (x), and stores the index i of the executed JT of connected nodes. A pointer may have an NIL value. Jumping actions are identified by pointers. If a pointer is (NIL, x), the action is unary and if it is (y, x), then it is binary. If the pointer is (NIL, NIL), then an action is submitted to a fixed server and a copy of the data objects are brought to the private work space of the centroid to process the JT. After data objects are brought the centroid is free to disconnect from the server.

VII. ENVELOPE TRANSACTION MECHANISM

In this section, a flexible processing mechanism is designed to support processing of jumping transactions initiated by a mobile host and joining an appropriate cluster. Existing methods like delegation operations [24, 23, 28], inter-process interactions [26] or for data processing do not have the capacity to process data in a mobile environment. The proposed mechanism can be carried out in both a synchronous and an asynchronous manner. Each jumping transaction is initiated from a private work space and after completion, the message is received. We differentiate two types of transactions: *Envelope and Pseudo transactions*. They are used as a coupling to process JTs at the ER. Envelope transactions are used to gain locks on data objects and then pseudo transactions are executed. Pseudo transactions are used to read or update data objects from an envelope at the ER. In concurrent JTs, processing envelope transactions of two jumping transactions share the lock and the associated pseudo transaction is processed at ER independently. For example, if ET_1 and ET_2 are envelope transactions of jumping transactions JT_1 and JT_2 then process JT_2, ET_1 transfers the lock to ET_2 and executes the corresponding pseudo transaction.

Before we discuss various tasks of envelope and pseudo transactions, we use the following notation: JTi^{p} denotes i^{th} jumping transaction initiated by a peer p, $JT_{iE}{}^{p}$ is the envelope transaction and the $JT_{iI}{}^{p}$ is the pseudo transaction initiated by $JT_i{}^{p}$. In our discussion, a peer may be taken as mobile host.

The tasks of envelope and pseudo transactions are given below:

Envelope Transaction (ET): For processing an envelope, envelope transactions are used to acquire locks on data objects of the envelope and various actions using pseudo transactions. A lock may be read or written. The role of envelope transaction the $JT_{iE}{}^{p}$ is to support JT_i and (i) share the envelope with the other JTs; (ii) transfer the locks for processing the JTs from L(i) to L(i+1) of the same DPT; (iii) transfer the locks for processing the JT's from the associated DPT at any level; (iv) save partial or updated results into envelopes. This will avoid loss of work due to peer failure; (v) Each JT initiates at least one envelope transaction, depending on the need of sharing data with other JT's.

Pseudo Transactions (PT): Pseudo transactions are the local transactions of a centroid and are used to read or update the data objects from an envelope. Pseudo transactions use different actions depending upon the state as follows:

Following actions are performed on consistent data objects at a centroid.

(a) A *read* action reads data objects where there are no conflicts between the locks on data objects. This action examines a set of attributes or set of tuples from consistent data bases.

(b) A *write* action makes modifications to data objects where there are no conflicts between the locks on data objects. This action modifies the set of attributes or set of tuples from a consistent database.

These actions are performed when the centroid has a connection with the server so that consistent data objects can be fetched. In a DPT, a node with a pointer (NIL, NIL) always fetches consistent data objects.

Following actions are performed on data objects located in an envelope at a centroid.

(a) A *pseudo write* action modifies data objects from an envelope. This action makes modification in set of attributes or set of tuples from an envelope.

(b) A *pseudo commit* (P-commit) action commits the JT and creates an envelope and places it in the private work space of the centroid. This is the final action of the

JT. Also an A *pseudo abort action* aborts the JT after successfully undoing actions carried out by the JT and destroys the envelope.

(c) A *pseudo read* action reads data objects from the envelope. This action reads attributes or a set of tuples from an envelope.

(d) Mobile transaction is *committed (aborted)* if all jumping transactions are pseudo commit (aborted).

In an asynchronous process, the JT is initiated from a user's private work space and triggers the ET at the ER followed by the PT at the run time in the ER of a centroid. In other words, the ET provides an appropriate lock to process the PT and the JT is P-committed. [See figure 3.]

An ET satisfies all ACID properties when one jumping transaction is P-committed in the ER at a time. On the other hand, the isolation property is relaxed when jumping transactions are processed in a synchronous manner. This means the data objects can be viewed before P-commit of the JT.

In a synchronous process, if two concurrent jumping transactions JT_1 and JT_2 want to update x from an envelope and the envelope transaction ET_1 of JT_1 has a required lock, then ET_1 transfers the lock on x to envelope transaction ET_2 of JT_2 on demand to process x from an envelope. Finally we define an envelope as follows:

Definition7. (Envelope): An envelope is a virtual list of data objects and is a triple (L, E, D) produced by the pseudo transaction. Here L is a mobile identification number of a peer, who has created an envelope using a JT, E is the identification number of a centroid, where the JT is pseudo committed to produce a list of data objects D. Envelopes are store in the private work space of the centroid until the transaction is committed.

A. *Pseudo Commit Features*

If all JT's are pseudo committed then the transaction is committed at the centroid. When a centroid reconnects to the server, these transactions have to synchronize with transactions from other centroids. If conflicts occur, they will be resolved and the modification of data objects is made permanent at the server; otherwise the transaction is aborted. To execute all pseudo transactions, envelopes are brought from the database server to the ER and modified according to the DPT guidelines. Modified envelopes are kept in a private local work space. The transaction is finally committed according to the order of the pseudo commit jumping transactions mentioned in DPT.

In a DPT, if all JTs are intentionally committed at L (i) then JTs at L (i+1) will be processed and can be intentionally committed. Envelope transactions are used to transfer the necessary locks from L (i) toL (i+1) to process the pseudo transactions.

[17] defined the ACTA transactional framework for reasoning about and synthesizing the dependencies among transactions. Here we reuse the commit dependency and abort dependency rules from the ACTA transactional framework where at least one JT is P-committed from the DPT at the ER of the centroid:

(a) JT P- commit means the ET received an appropriate lock to execute the PT. JT is P- aborted means the ET may not get a required lock to process the PT or the ET got a lock but the PT failed.

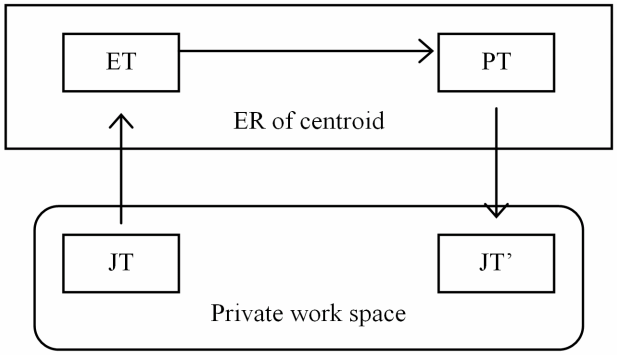


Figure 3.

(b) If the JT is initiated but the demanded lock to process the PT is not issued by the ET, the JT is aborted. However, processing the PT can be delayed until the ET acquires a lock from an associated ET and hence the JT can be P-committed.

(c) Suppose the JT is initiated and the corresponding ET acquires a lock on the data object to process the PT. If the PT is not P-committed, then the PT is P-aborted.

(d) In the process of P-commit of the JT, if the peer goes into an autonomous or idle state, then the remaining processing can be carried out by another peer of the same cluster until peers establish a connection to centroid.

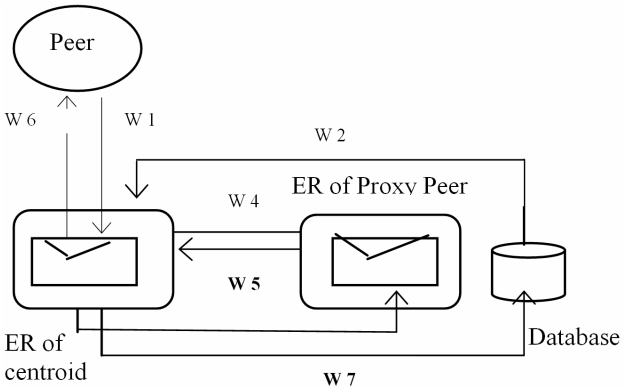
B. *Envelope P- write protocol*

In this section, we implement a consistency protocol for envelopes to make transactions globally serialized. The non-blocking primary protocol [19] is modified for envelopes, created for data object x, to coordinate the write operation on x at the ER.

In the envelope P-write protocol, all read and written operations are performed on an envelope brought from a database server to the ER of a centroid. A primary copy of an envelope is also kept at one of the peers called proxy peer. It is not used by any of the users but works as a centroid when the centroid is not functioning. All the updates are made to the proxy peer from a centroid, but block operations on envelopes are made to update from the proxy peer to the data base server. The advantages are: (i) In the disconnected mode of a centroid, a proxy peer carries out all the operations as a centroid. Later on, when connecting again, updates are propagated from the proxy peer to the centroid. (ii) All the updates also remain at the database server. The protocol works as follows:

Suppose a JT wants to modify data object x, and if the envelope is not at the ER of a centroid, then it is created at a private work space of a centroid and brought to the ER of a centroid. A copy of an envelope is placed at the ER of the proxy peer. Updates are fetched by the proxy peer from the centroid and acknowledgment is sent to a centroid. A P-commit action is used to inform a peer for completion of JT using DPT.

Block updates are sent to the database server by specifying time (See Figure 4). The same way pseudo read actions can be performed on the envelope.



W1: P-Write request
W2: Move envelope to ER of a centroid
W3: Copy of an envelope to a proxy peer.
W4: Updates an envelope at ER of proxy peer
W5: Acknowledgment to envelope at ER of centroid
W6: Pseudo commit message to peer with DPT
W7: Blocked updates at a given time data base

Figure 4.

C. *Envelope processing features*

The jumping transaction process at the ER by two consecutive jumping transactions of the same DPT initiated by a peer is carried out in an asynchronous manner on an envelope and the resulting envelope is stored at the private work space of the centroid. For instance, let $JT_1^{\ p}$ and $JT_2^{\ p}$ be consecutive JTs of the same DPT initiated by p. If $JT_1^{\ p}$ is pseudo committed and creates an envelope ENV_1, then an envelope transaction $JT_{2E}^{\ p}$ connects to $JT_{1E}^{\ p}$ and requests a lock to process $JT_{2I}^{\ p}$ on an envelope ENV_1 changed by $JT_1^{\ p}$ to create a new envelope ENV_2.

Processing of jumping transactions of the same DPT initiated by p is carried out on different envelopes in a synchronous manner.

(a) Let $JT_x^{\ p}$ and $JT_y^{\ p}$ be jumping transactions of the same DPT and simultaneously connected to carry out a synchronous process on different envelopes in the ER, the envelope and pseudo transactions $JT_{xE}^{\ p}$, and $JT_{xI}^{\ p}$ of $JT_x^{\ p}$ are initiated and executed in parallel with $JT_{yE}^{\ p}$ and $JT_{yI}^{\ p}$ of $JT_y^{\ p}$ on separate envelopes ENV_x and ENV_y in the ER.

(b) If the ER is physically distributed among different centroids of a cluster, then jumping transaction processing is carried out in synchronous manner.

Processing of $JT_x^{\ p}$ and $JT_y^{\ p}$ continues in a synchronous manner at the ER until a asynchronous point is reached and a new envelope ENV_z is created in the ER. On the other hand, if the ER is distributed among different centroids of a cluster, then ENV_x is brought to the ER where ENV_y resides and then makes a join or other way.

The envelope sharing process at the ER is carried out by jumping transactions from different DPTs, initiated by mobile hosts p and o in synchronous manner if $JT_x^{\ p}$ and $JT_y^{\ o}$ are connected to each other simultaneously. In other words, envelope transactions $JT_{xE}^{\ p}$ and $JT_{yE}^{\ o}$ are initiated and executed in parallel with pseudo transactions $JT_{xI}^{\ p}$ and $JT_{yI}^{\ o}$.

In mobile environments, disconnections can happen any time during the processing of an envelope. The mechanism should have the ability to recover from disconnections to ensure that the jumping transaction processing is carried out correctly. Suppose $JT_x^{\ p}$ and $JT_y^{\ o}$ are disconnected. The connectivity can be established by linking $JT_x^{\ p}$ to ER and sharing an envelope with $JT_y^{\ o}$ using envelope transaction $JT_{xE}^{\ p}$. Furthermore, a jumping transaction $JT_y^{\ o}$ can be connected to the ER to obtain data objects of an envelope using envelope transaction $JT_{yE}^{\ o}$. This is possible because $JT_{xE}^{\ p}$ is connected to $JT_{yE}^{\ o}$ for envelope sharing in the ER. This process is completed in asynchronous manner.

VIII. CONNECTED COST MODEL

In this section, we analyze our work using probabilistic analysis. This section is divided into two subsections. In the first section, we probabilistically study the static data object allocation method and in the second section, we study the dynamic data allocation method. In each of these subsections, we derive the expected cost first and then average expected cost and then compare these methods.

In this study, the communication cost is taken as 1 unit if cost is required, otherwise 0 units are taken. We assume that pseudo actions (read or write) follow a Poisson distribution with parameter λ_r for pseudo read and λ_w for pseudo write. Denote $\theta = \frac{\lambda_w}{\lambda_r + \lambda_w}$. At any point, the action θ is the probability of pseudo write and 1- θ is the probability of pseudo read because the Poisson distribution is memoryless. The following two measures are used to do the analysis.

1. Suppose X is the data allocation method and λ_r, λ_w are pseudo read and write distributions, respectively. We denote $Exp_X(\theta)$, the expected cost of a relevant action.

2. Suppose θ varies over time with equal probability of having any value between 0 and 1. Then we define the average expected cost per action, denoted AVG_X, to be the mean value of $Exp_X(\theta)$ for θ ranging between 0 and 1, namely: $AVG_X(\theta) = \int_0^1 Exp_X(\theta)\, d\theta$

The cost depends on the existence of a copy of c at the centroid (C) and the proxy peer (P). A cost is determined as follows:

1. If there exists a copy at C and P, then pseudo write cost is that cost which is required to make modifications at C and send to P. A pseudo read cost is taken to be zero. This means, that if there is a copy of data objects at C and P, then the cost depends only on pseudo write actions.
2. If no copy exists at C, then we assume that the cost of pseudo read as one because data objects are to be brought from the DB server to C and P. A pseudo write cost is taken to be zero as there is no data object, so pseudo write actions cannot be performed. This means that if there is no copy of data objects, then the cost depends on the pseudo read action. Here, a schedule must contain the pseudo read action followed by the pseudo write action.

A. *Probabilistic analysis of static methods*

A static method is used to determine the cost of a transaction when a copy of data objects may or may not exist at C and P. Following are two types of static methods:

Method SA_1: A method where copies of data objects do not exist at C and P. The pseudo write cost is zero and the pseudo read cost is one.

Method SA_2: A method where copies of data objects exist at C and P. The pseudo write cost is one and the pseudo read cost is zero.

From above, $\text{Exp}_{SA_1}(\theta)$ and $\text{Exp}_{SA_2}(\theta)$ have qual probability for pseudo read and write actions respectively. Thus, $\text{Exp}_{SA_1}(\theta) = 1 - \theta$ and $\text{Exp}_{SA_2}(\theta) = \theta$.

Average expected costs are obtained as

$$\text{AVG}_{SA_1}(\theta) = \int_0^1 (1-\theta)\, d\theta = \frac{1}{2} \text{ and}$$

$$\text{AVG}_{SA_2}(\theta) = \int_0^1 \theta\, d\theta = \frac{1}{2}$$

B. Probabilistic analysis of dynamic method

A dynamic method DA executes k pseudo write actions fired by n users. The probability of the existence of a copy at C and P is denoted by α_k and it is probable that the majority among preceding k pseudo read actions and the probability that the number of preceding k pseudo write actions for n users. Assume that k > n. Thus,

$$\alpha_k = \sum_{j=0}^{n} {}^{k}C_j\, \theta^j (1-\theta)^{k-j}$$

Theorem 1. Expected cost of a DA is

$\text{Exp}_{DA}(\theta) = \alpha_k \theta + (1-\alpha_k)(1-\theta)$ for all k and θ.

Proof: Let p be an action. When there is a copy at C and P then the expected cost of p is equal to the probability that p is pseudo write and is θ. When there is no copy at C and P, then the expected cost of p is equal to the probability that p is a pseudo read action and is 1- θ. Thus, the expected cost of p is the probability that p is a pseudo write action under the condition that there exists a copy of data objects or the probability that p is a pseudo read action under the condition that there is no copy of data objects. Hence we get $\text{Exp}_{DA}(\theta) = \alpha_k \theta + (1-\alpha_k)(1-\theta)$.

Now we see the relationship between SA_1, SA_2 and DA.

Theorem 2. For all k and θ,

$$\text{Exp}_{DA}(\theta) \geq \min\{\text{Exp}_{SA_1}(\theta), \text{Exp}_{SA_2}(\theta)\}$$

Proof: We have,

$$\begin{aligned}\text{Exp}_{DA}(\theta) &= \alpha_k \theta + (1-\alpha_k)(1-\theta) \\ &= \alpha_k \text{Exp}_{SA_1}(\theta) + \\ &\quad (1-\alpha_k)\text{Exp}_{SA_2}(\theta)\end{aligned}$$

Thus $\text{Exp}_{DA}(\theta) \geq \min\{\text{Exp}_{SA_1}(\theta), \text{Exp}_{SA_2}(\theta)\}$

This shows that the expected cost of dynamic methods is greater than that of static methods. The following theorem calculates the average cost of the DA and shows that it is less then the minimum of static methods.

Theorem 3. Let the DA execute k pseudo write actions fired by n users such that k > n. Then

$$\text{AVG}_{DA}(\theta) = \frac{1}{2} + \frac{(n+1)(n-k)}{(k+1)(k+2)}$$

Proof: Using theorem 1, after some algebraic simplifications, it can be shown that,

$$\begin{aligned}\text{AVG}_{DA}(\theta) &= \int_0^1 (\alpha_k \theta + (1-\alpha_k)(1-\theta))\, d\theta \\ &= \int_0^1 (1 - \theta - \alpha_k + 2\alpha_k \theta)\, d\theta \qquad (1)\end{aligned}$$

To find the above integral following identity is used for positive integers a and b :

$$\int_0^1 (x^a + (1-x)^b)\, d\theta = \frac{a!\,b!}{(a+b+1)!}$$

Using above identity :

(i) $\int_0^1 \alpha_k\, d\theta = \frac{n+1}{k+1}$ and

(ii) $\int_0^1 \alpha_k \theta\, d\theta = \frac{(n+1)(n+2)}{2(k+1)(k+2)}$

Substitute in (1) and after few simplifications, we obtain,

$$\begin{aligned}\text{AVG}_{DA}(\theta) &= \frac{1}{2} + \frac{(n+1)(n+2)}{(k+1)(k+2)} - \frac{n+1}{k+1} \\ &= \frac{1}{2} + \frac{(n+1)(n-k)}{(k+1)(k+2)}\end{aligned}$$

Corollary 1. The average expected cost of static methods decreases when pseudo write actions k increase and

$$\text{AVG}_{DA}(\theta) < \min\{\text{AVG}_{SA_1}(\theta), \text{AVG}_{SA_2}(\theta)\}$$

Proof: From the above theorem and as k> n, we see that $\text{AVG}_{DA}(\theta)$ decreases when pseudo write action k increases.

From the above, we conclude that if the number of users n increases and the number of pseudo write actions is greater than n, then the average expected cost decreases.

IX. IMPLEMENTATION

The implementation of a jumping transaction mechanism is done as follows:

All the related information regarding transactions in clusters of peers is described by an XML document. The specification of a submitted transaction is converted into an internal SQL query representation via XML parser. We used Xerces2 parse, which is the existing parser [34] to support the transformation of transaction specification.

The transaction execution manager takes an SQL query as input. When an SQL query is received, the transaction execution manager will submit this to be executed in server via standard JDBC connection. If an envelope transaction is received, the transaction execution manager will carry out the execution of pseudo transactions via write () or read () method of the Java Transaction API.

The envelope repository is designed and implemented with Jini and JavaSpace [32, 33]. An envelope repository is created by the transaction execution manager when an envelope transaction is initiated by the jumping transaction. An envelope repository is allocated at one computer due to the limitation of the JavaSpace technology.

In this implementation, the MySQL locking model is used with the standard commit functionality. This does not contrast with switching off the auto-commit functionality in the mobile locking system.

The performance of the jumping transaction mechanism with the support of envelope repository has been tested by assuming that data objects and corresponding locks exist

at a centroid and we got significant improvement in system throughput.

X. CONCLUSION

In this work, we presented a jumping transaction mechanism to process data objects at the centroid of a cluster of peers. After reviewing related works, the concept of a peer connected set and cluster of mobile hosts was introduced. We then discussed various features and related concepts required to process transactions at the centroid of a cluster initiated by peers. Envelopes are kept at a private local work space and are brought to an envelope repository to process until the transaction is committed. Pseudo committed data objects are visible to other transactions before committing the transaction. This relaxes the isolation property. A proxy peer is kept for processing transaction as standby if the centroid fails in pseudo write protocol. Various features related to sharing an envelope by jumping transactions are also discussed.

Further research will have to consider the following issues. Firstly, proper lock management is to be developed. Secondly, a data conflict awareness mechanism is to be developed to support mobile transactions to show conflicts among database operations in mobile environments.

REFERENCES

[1] K.-I. Ku and Y.-S. Kim: *Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems*, Research Issues in Data Engineering (RIDE), 2000, pp 39- 46.

[2] P. K. Chrysanthis: *Transaction Processing in Mobile Computing Environment*, IEEE Workshop on Advances in Parallel and Distributed Systems, 1993, pp 77-83.

[3] P. Serrano-Alvarado: *Defining an Adaptable Mobile Transaction Service*, Extending Database Technology (EDBT) Workshops, 2002, pp 616-626.

[4] R. Schneiderman: *The Mobile Technology Question and Answer Book A Survival Guide for Business Managers*, American Management Association, 2002.

[5] M. H. Dunham, A. Helal and S. Bal Krishnan: *A Mobile Transaction Model Tha Captures Both the Data and Movement Behavior.* Mobile Networks and Applications (MONET), 2(2), 1997, pp 149-162. (doi:10.1023/A:1013672431080)

[6] G. D. Walborn and P. K. Chrysanthis: *Transaction Processing in PROMOTION* ACM Symposium on Applied Computing (SAC), 1999, pp 389-398.

[7] Ester M., Kriegel H.-P., Sander J. and Xu X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, OR, 226-231, 1996.

[8] P.K. Chrysanthis: *Transaction processing in a mobile computing environment* proceeding of IEEE work shop on Advanced in Parallel and Distributed Systems , pp 77-82,Oct. 1993

[9] Ester M., Kriegel H.-P., Xu X: *A Database Interface for Clustering in Large Spatial Database,* Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining. Montreal, Canada, 94-99. 1995.

[10] D. Ratner, P. L. Reiher, G. J. Popek and G. H. Kuenning: Replication Requirements in Mobile Environments, Mobile Networks and Applications (MONET), 6(6), 2001, pp 525-533. (doi:10.1023/A:1011862121702)

[11] Satyanarayanan.M. *Fundamental Challenges in Mobile Computing* In Proceedings of the 15 th ACM Symposia on Principles of Distributed Computing, 1996.

[12] P.A. Bernstein, V. Hadzilacos, Goodman: *Concurrency control and recovery in database systems,* Addison Wesley 1997

[13] Sushil Kulkarni, Dr. M.A. Sanglikar: *Spatio-Temporal Schema Model And Carrier and Courier mobile query processor for Spatio-Temporal database* Urban Planning and Environment: Strategies and Challenges international Conference , Elphinstone College, Mumbai, pg.116-117, January 30th & 31st, 2007

[14] E. Pitoura and B. K. Bhargava: *Data Consistency in Intermittently Connected Distributed Systems*, IEEE Transactions on Knowledge and Data Engineering (TKDE), 11(6), 1999, pp 896-915. (doi:10.1109/69.824602)

[15] Sushil Kulkarni: *Envelope Model: Data Access from any where, International* Conference by IPSI - Montenegro, September 24-29, 2005

[16] S. K. Madria and B. K. Bhargava: *A Transaction Model for Mobile Computing*, International Database Engineering and Application Symposium (IDEAS), 1998,pp 92-102.

[17] P. K. Chrysanthis and K. Ramamritham: *Synthesis of Extended Transaction Models Using ACTA*, ACM Transactions on Database Systems (TODS), 19(3), 1994, pp 450-491. (doi:10.1145/185827.185843)

[18] S. K. Madria and B. K. Bhargava: *A Transaction Model to Improve Data Availability in Mobile Computing*, Distributed and Parallel Databases, 10(2), 2001, pp 127-160. (doi:10.1023/A:1019232412740)

[19] Budhiraja N, Manzullo K. Schneider E. and Toueg S: *The Primary Backup Approach*, In Mullender S(ed.), Distributed Systems,pp 199-216. Wokingham, Addison Wesley, 2 nd edition 1993.Cited on page 297.

[20] D. Bottazzi, A. Corradi and R. Montanari: *A context-aware group management middleware to support resource sharing in MANET environments*, International Conference on Mobile Data Management (MDM), 2005, pp 147-151.

[21] J. Holliday, D. Agrawal and A. E. Abbadi: *Disconnection Modes for Mobile Databases*, Wireless Networks, 8(4), 2002, pp 391-402. (doi:10.1023/A:1015542723791)

[22] J. Liu, D. Sacchetti, F. Sailhan and V. Issarny: *Group management for mobile Ad Hoc networks: design, implementation and experiment*, International Conference on Mobile Data Management (MDM), 2005, pp 192-199.

[23] H. Ramampiaro: *CAGISTrans: Adaptable Transactional Support for Cooperative Work*, Dr.ing thesis, Norwegian University of Science and Technology (NTNU), 2001.

[24] P. K. Chrysanthis: *Transaction Processing in Mobile Computing Environment*, IEEE Workshop on Advances in Parallel and Distributed Systems, 1993, pp 77-83.

[25] G. P. Picco, A. L. Murphy and G.-C. Roman: *Developing mobile computing applications with LIME*, International Conference on Software Engineering (ICSE), 2000, pp 766-769.

[26] N. Prabhu, V. Kumar, I. Ray and G.-C. Yang: *Concurrency Control in Mobile Database Systems*, International Conference on Advanced Information Networking and Applications (AINA), 2004, pp 83-86.

[27] C. Chen, W. Wu and Z. Li, Multipath Routing Modeling in Ad Hoc Networks, *Proc. of ICC' 05*, Volume 5, May 2005, pp. 2974 – 2978

[28] A. Brayner and J. d. A. M. Filho: *Sharing Mobile Databases in Dynamically Configurable Environments*, 15th International Conference on Advanced Information Systems Engineering (CAiSE), 2003, pp 724-737.

[29] Istanbullu, A. (2008). Mobilim: Mobile Learning Management Framework System for Engineering Education. *Int. Jour. Ed.Educ. Vol. 24, No. 1, p.32-39.*

[30] Economides, A.E. and Nikolaou, N. (2008). Evaluation of Handheld Devices for Mobile Learning. *Int. Jour. Ed. Educ. Vol.24. No.1. p. 3-13.*

[31] Sushil Kulkarni, Dr. M.A. Sanglikar: Envelope Transaction Mobile Model; Second International Conference on Informatics, Universiti Malaya, Kuala Lumpur, Malaysia. 26, 27 Nov 2007.

[32] Jini Specifications and API Archive, Sun Microsystems, http://java.sun.com/products/jini/.

[33] F E. Freeman, S. Hupfer and K. Arnold: Java Spaces: principles, patterns, and practice, Addison-Wesley, 1999.

[34] http://xml.apache.org/xerces2-j/

AUTHORS

Sushil Kulkarni is with Mathematics Department, Jai Hind College affiliated to University of Mumbai at Mumbai, India (e-mail: sushiltry@gmail.com).

Mukund Sanglikar is with Mathematics Department, Mithibai College affiliated to University of Mumbai at Mumbai, India. He was formerly Head of the Department of Computer Science at University of Mumbai. (e-mail: masanglikar@rediffmail.com).

Submitted 30 May 2009. Published as resubmitted by the authors on 9 October 2009.