Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment

L. H. Yeo & A. Zaslavsky

School of Computing & Information Technology, Frankston, Monash University
McMahons Road, Frankston, Victoria 3199, Australia

Abstract

In a multidatabase environment with mobile computers involved, the nature of computing is such that the user may not wait for the submitted global transaction to complete before disconnecting from the network. In this paper, a basic architectural framework to support transaction management in multidatabase systems is proposed and discussed. A simple Message and Queuing Facility is suggested which provides a common communication and data exchange protocol to effectively manage global transactions submitted by mobile workstations (MWS). The state of global transactions is modelled through the use of transaction sub-queues. The proposed strategy allows an MWS to submit global transactions and then disconnect itself from the network to perform some other tasks thereby increasing processing parallelism and independence.

1 Introduction

With the decentralisation of business enterprises, there is a shift in the processing requirements: from online processing environment accessing a centralised host computer system in the 1980's to that of a distributed processing environment in the 1990's. The main driving forces that influence the emergence and adoption of distributed computing technologies are: (1) information systems are now being decentralised and are developed utilising heterogeneous computer hardware and software platforms; (2) widespread implementation of local area networks and wide area networks. These have given rise to a distributed heterogeneous computing environment with multi-hardware, multi-software and multi-network protocols with widespread sharing of computer resources such as communication facilities, files, printers and databases. Remote access to critical corporate data resources by end-users has also become an important strategic requirement for strengthening the business competitiveness of the enterprise. Consequently, new application systems executing in such a decentralised environment may require access to distributed data resources and exchange information with each other. The autonomy of each site also implies that different concurrency control and recovery facilities are implemented. Thus, the management of distributed transactions across the heterogeneous multidatabase environment has become an important research issue.

Advancements in wireless communications (such as cellular communications and Cellular Digital Packet Data) as well as in portable and mobile computers (or mobile workstations), such as laptops and Personal Digital Assistants (hand held pen-based computers) have enabled telecommuting to become a reality. In an ever increasing degree, people are now working from home and are connected to their offices via telecommunications lines. The communication network also provides the basic infrastructure that facilitates the use of mobile computers including data transfer and message exchange. Consequently, mobile computing paradigm will have a significant impact on both hardware and software design including database systems. From the database systems' perspective, several research areas need to be revisited such as query processing, transactions processing, and security [1]. In the context of transaction management, the transaction model needs to include long-duration transactions and Sagas [7]. While the connection of mobile computers to some coordinating site may be only for a limited duration, the synchronisation and recovery of global transactions submitted will form a new area of challenges to database researchers.

Cooperative processing in a distributed multidatabase environment has also become an emerging technology. In a distributed cooperative processing environment, part of the processing is carried out at the workstation while the partner application is executing at the host computer system. This also assumes that the applications are to be developed and processed on a suitable platform. Such platform should allow transparent access by users so that they are not concerned with the location of resources. In this respect, cooperative processing is more specialised and is a subset of a distributed processing [2, 11]. Therefore, there are two distinct characteristics in a cooperative processing environment: (1) there exists a distributed database system to support and manage transactions and data resources; (2) there are workstations that initiate transactions with their subtransactions being processed at one or more sites. In such a scenario, a global transaction is initiated by an end-user using a mobile computer. Therefore, one can observe that mobile computing can be considered as part of the distributed processing environment [1]. We believe that transaction management in a cooperative processing environment utilising distributed multidatabase systems (MDS) with mobile computing is an important area of research.

ACID (Atomicity, Consistency, Isolation and Durability) properties have been widely used and studied in the formulation of concurrency control and recovery mechanisms used in database management systems. At the same time, new emerging computing environment based on cooperative processing and distributed heterogeneous database systems adds new dimensions to database problems. Firstly, there is a requirement for autonomy of local database systems to be preserved. Secondly, the mobile workstations may be disconnected from the network most of the time. Therefore, a distinction should be made between a logical disconnection of a mobile workstation and a failure in the network. In the former case this is planned and well prepared in advance. Hence, in the context of transaction management, such distinction must be considered in its design. Thirdly, the end user may not wait for the transaction to complete before disconnecting from the system. Therefore, transaction management in such an environment offers new challenges both at the workstation level and at the MDS level.

To facilitate our research, five main goals have been set. Our first goal is to provide a full-fledged transaction management framework so that the users and application programs will be able to access data across multiple sites transparently. Our second goal is to attempt to enhance database concurrency and data availability through the adoption of a distributed concurrency control and recovery mechanism that preserves local autonomy, that is, the local DBMS has full control over the execution of transactions (both global subtransactions and local transactions). The third goal is to implement the concept of extensibility to support various database systems in our framework so that our components can coexist with a relational or an object-oriented database system. The fourth goal aims at providing an environment where the proposed transaction processing component operates independently and transparently of the local DBMS. This also implies that a local database system can participate or detach itself from the MDS without having to be concerned with the implementation complexity of the proposed framework. This means that the MDS must ensure that all outstanding global transactions must be completed successfully before a detachment process can occur. Finally, we incorporate the concept of mobile computing through the use of mobile workstations into our model. Therefore, our model addresses both the management of distributed transactions and mobile computing.

The main aim of this paper is three folds. Firstly, to present a brief overview of a generalised architectural framework that supports mobile computing in a cooperative multidatabase processing environment. Secondly, to present a Message and Queuing Facility (MQF) to manage global transactions submitted by mobile workstations thereby improving transaction parallelism and independence in a distributed MDS environment. Thirdly, to discuss management of mobile workstations within the proposed framework.

2 Related Work

Transaction management in an MDS environment has some inherent problems which are a result of the requirement that the underlying participating DBMSs should have complete autonomy over the execution of local transactions. These problems have been extensively studied and numerous strategies have been proposed [10].

In general, there are three main requirements that may cause problems in designing a transaction management scheme in an MDS environment [6]. Firstly, due to the heterogeneity of the underlying database management systems that are being implemented, the Global Transaction Manager (GTM) must have the capability of dealing with different transaction managers. Secondly, the underlying DBMSs have the ability to implement their own concurrency control mechanisms. Thirdly, the Local Transaction Manager (LTM) may not communicate information regarding local concurrency control back to the GTM. As such, the GTM is unaware of any conflicts between local transactions and the global subtransactions. The concept of serializability has been used as a correctness criteria for concurrent execution of multiple transactions in a centralised and decentralised database environment. However, in an MDS environment, ensuring global transaction serializability is complicated by the fact that the LTM is autonomous and it can also implement different concurrency control mechanism. One of the main problems encountered in maintaining global transactions serializability is how to ensure the execution order of global transactions by the local database systems. This is due to the fact that indirect conflicts between global subtransactions and local transactions can occur, thereby causing the execution order of these global subtransactions to be altered [8].

One of the approaches is to restrict the global transaction type [4]. The main rationale is to disallow any two global transactions to access the same site. Such a scheme does provide some site autonomy. However, the underlying DBMS must implement strict two-phase locking protocol. Moreover, the MDS is designed to be a centralised system which is subjected to site and network failures. Another approach is to force global transactions to obtain a ticket (logical timestamp) stored in the local database system [8]. Such an action would cause an additional conflict to occur between global subtransactions thereby maintaining their execution order. Other approaches relax the serializability as the correctness criteria. For instance, the concept of Sagas had been introduced by Garcia-Molina and Salem [7] for long-lived transactions. Another approach is to use quasi-serializability [5] where no value dependencies exist among databases so that indirect conflicts between global subtransactions and local transactions cannot occur.

3 Motivation

Consider a scenario of a hypothetical wheat marketing agency using cooperative processing in an MDS environment shown in Figure 1.

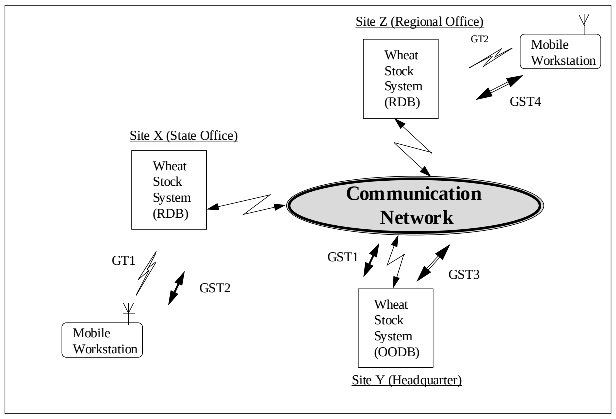

Fig.1. Distributed Processing Environment of a Wheat Marketing Agency

Consider a global transaction GT1, where a wheat marketing officer may be negotiating a contractual agreement on wheat procurement from a wheat grower using mobile workstation to access information at some remote host computer located at Site X via wireless communication facilities. Information such as the stock level of a particular grade of wheat (GST1) may be obtained from a central host computer located at Site Y. Information on grain availability and transport facilities (GST2) may also be obtained from its regional office at Site X simultaneously. The wheat marketing officer may then disconnect from the network and perform some other tasks. He may then reconnect to Site X to query the status of GT1 at some point in time. The results obtained from GT1 may then be scrutinised by a knowledge-based or a decision support application so that the wheat marketing officer can then present various price and payment options available to the wheat grower. Once the contractual agreement has been signed, the agreed price and delivery details must be reflected in their respective databases.

Now consider another global transaction GT2 which is a wheat sale initiated at Site Z. Information on wheat stock (GST3) obtained from Site Y together with the information on location of wheat silos and transport arrangements (GST4) obtained from Site Z may be used to obtain the best location to collect the wheat grains. The prices and location of delivery may depend on whether GT1 has been committed or not as the stock level will affect the prices and delivery schedules.

In this scenario, a decision support processing system is depicted where the structure of the environment is dynamic and the transaction rates are low. The information used, for example wheat stock, represents only a snapshot of its value at some moment in time. Discrepancies of data can therefore occur, however, these can be resolved quite easily. For example, we can change the application semantics such that if the wheat stock availability is above a certain prescribed threshold level, then transaction GT1 may be allowed to proceed. Alternatively, another extraction of the wheat stock data can be performed and then be used to contrast with the previous extraction so that a reconciliation between these two sets of data can be carried out. Another possible solution would be to introduce time-constraint into the execution of global transactions such that if the deadline is not met, then the global transactions should not be executed at all. Hence, one could design a customised concurrency control mechanism and recovery management scheme specific to this operating environment based on the semantic requirements of such application needs.

This example clearly illustrates that the design of a transaction management scheme in a cooperative processing environment involving heterogeneous and autonomous database systems is complex and difficult. Disconnection of mobile workstations from the network is an important issue as a mobile workstation may disconnect itself from the network after the submission of a global transaction. Two problems can arise from such an implementation: (1) the differentiation between a connection failure and an orderly disconnection from the network; (2) the management of global transactions submitted to the host computers by the users of these mobile workstations after the disconnection. Therefore, these problems offer new challenges in the design of a transaction management scheme both at the global level and at the local level. The following section discusses the design and conceptual framework of the proposed Multidatabase Transaction Processing Manager (MDSTPM) architecture.

4 Overview of MDSTPM System Architecture

An MDS is an integrated distributed database system consisting of a number of autonomous component database management systems. Each of the underlying component database systems is responsible for the management of transactions locally. To facilitate the execution of global transactions, an additional layer of software must be implemented which permits the scheduling and coordination of transactions across these multiple component database management systems. Several questions can be raised that will influence the overall design of MDSTPM architecture, namely:

- How to build a layered transaction management architecture that would be implemented over existing DBMSs and mobile workstations?
- Given that mobile workstations will be disconnected for a considerable period of time, what is the impact of mobile computing on transaction processing?
- What is the most appropriate transaction model to be used?
- How to preserve the autonomy of local DBMSs?
- What is the correctness criteria to be used for such a transaction management scheme?
- What type of concurrency control mechanism should be used?
- What is the crash/recovery strategy to be implemented?
- How to support an effective and efficient distributed processing in an MDS environment?
- If a mobile workstation is a part of an MDS, and a global transaction requires an information kept there, what would be the most appropriate strategy to manage transactions given that the workstation is disconnected for a considerable period of time?
- How to notify a disconnected mobile workstation that a global transaction has been submitted which requires some data kept there ?

The proposed MDSTPM architecture incorporating mobile computing is depicted in Figure 2. This layered architecture is discussed in more detail in [13].

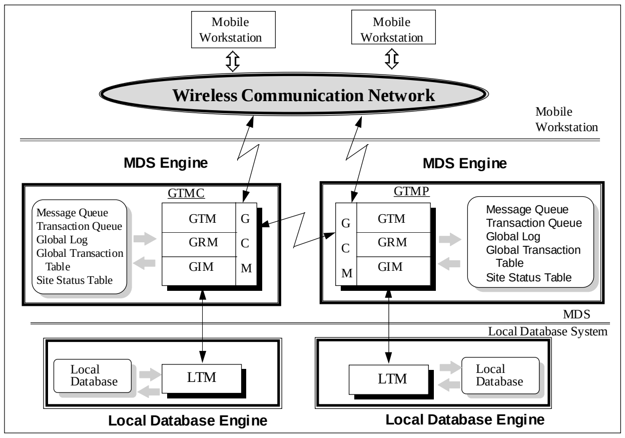

Fig.2. MDSTPM Architecture

The MDSTPM consists of the following components:

- The *Global Communication Manager* (GCM) is responsible for the generation and management of message queues within the local site. In addition, it also communicates, delivers and exchanges these messages with its peer sites and mobile workstations in the network.
- The *Global Transaction Manager* (GTM) coordinates the submission of global subtransactions to its relevant sites. The Global Transaction Manager Coordinator (GTMC) is the site where the global transaction is initiated. All participating GTMs for that global transaction are known as GTMPs. The GTM can be categorised into: Global Scheduling Submanager (GSS) and Global Concurrency Submanager (GCS). The GSS is responsible for the scheduling of global transactions and global subtransactions. The GCS is responsible for the acquisition of necessary concurrency control requirements needed for the successful execution of a global transactions/subtransactions. The global transaction manager (GTM) is responsible for the scheduling and commitment of global transactions and global subtransactions while the local transaction manager (LTM) is responsible for the execution and recovery of transactions executed locally.
- The *Global Recovery Manager* (GRM) coordinates the commitment and recovery of global transactions and global subtransactions after a failure. It ensures that the effects of committed global subtransactions are written to the underlying local database or none of the effects of aborted global subtransactions are written at all. It also

uses the write-ahead logging protocol so that the effects to the database are written immediately without having to wait for the global subtransaction to complete or commit.

- The *Global Interface Manager* (GIM) coordinates the submission of request/reply between the MDSTPM and the local database manager which can be executing in a relational database system (RDB) or an object oriented database system (OODB). This component provides extensibility functions including the translation of an SQL request to an object-oriented query language request.

5 Management of Distributed Global Transactions and Mobile Workstations

Our approach to the management of mobile workstations and the global transactions submitted is to have these mobile workstations to be part of the MDS during its connection with their respective coordinator node. Once a global transaction has been submitted, the coordinating site can then schedule and coordinate the execution of the global transaction on behalf of the mobile workstation (it is beyond the scope of this paper to discuss the concurrency and recovery strategies to be implemented). A detailed discussion of these can be found in [12,13]. The main rationale for this strategy is: (1) the user of the mobile workstation may disconnect from the network and perform some other tasks without having to wait for the global transaction to complete; (2) the host computers are connected to each other with an existing reliable communication networks and are thus less susceptible to network failures.

One of the mechanisms that has been used extensively for interprocess communication in a distributed computing environment is Remote Procedure Call (RPC), eg, the Open Software Foundation's Distributed Computing Environment RPC model. In the RPC paradigm, an application program requests services from another application executing in a remote node. This strategy is analogous to a subroutine call by a main program with parameters being passed to the processing node to direct processing requirements. Such an implementation would imply that events are occurring synchronously as the caller would have to wait for the control to be returned back before continuing its processing.

An alternative approach termed Message and Queuing Facility (MQF) is proposed to facilitate the implementation of the overall strategy. A mobile workstation (*mws*) sends a request message (together with the information required for processing) to its pre-assigned coordinating node for processing. Messages are thus handled asynchronously enabling the mobile workstation to disconnect itself from the network to perform some other tasks leaving the coordinating node to coordinate the execution of the global transactions submitted on its behalf. Moreover, as these mobile workstations may have totally different reliability characteristics compared to host computers located at the various remote sites, their connection period may be intermittent and short. Because of the nature of mobile computing, the MQF strategy would be most appropriate in this implementation as: (1) it is simple to manage the delivery and recovery of messages; (2) it is time independent in that mobile workstations may be disconnected from the network for an unbounded period of time while the global transactions submitted by these mobile workstations are being coordinated and executed by their respective coordinating node; (3) the ability of each workstation to query the status of its global transactions at its convenience. The constructs for MQF and the procedural flow will be discussed further on.

5.1 Management of Message Queues

In the proposed MQF, messages can be classified into three types: (1) Request Message; (2) Acknowledgment Message; (3) Information Message.

Typical examples of a *Request message* are Req_Reconnect (Request for Reconnection), or Req_Mws_Status (Request Mobile Workstation Status).

Acknowledgment messages are created in response to a request message and might include, for example, Ack_Reconnect_Mws (Acknowledge Reconnection to mobile workstation).

Information messages include among others Info_Msg_Queue (Message Queue status) or Info_Mws_Status (Mws/Site status information).

For each of the mobile workstations there exists a message queue and a transaction queue. Below we discuss the management of mobile workstations through the use of these queues.

5.2 Management of Mobile Workstations

To make a connection to the coordinating node, the mobile workstation will perform the following:

- Checks its *mws* status table for its last connection status, where the status_table={node_id, mws_id, mws_timestamp, mws_status}.
- Sends a Req_Connect to the designated GTMC with the following {node_id, mws_id, mws_timestamp, action}, where action={connect}.
- Writes the connect request into the mobile workstation log file.
- Updates the *mws* status table.

Upon receipt of this request, the GCM will perform he following:

- Logs the connect request into its log file and then acknowledges the connection by replying Ack_Connect back to the mobile workstations. It also checks and updates the workstation status table that is kept in storage.
- Scans the output queue for any outstanding output and routes this information to the mobile workstation if necessary.

To make a disconnection from its coordinating node, the mobile workstation will perform the following:

- Sends a Req_Disconnect request to the designated GTMC with the following {node_id, mws_id, mws_timestamp, action}, where action = {disconnect}.
- Writes the disconnect request record into the mobile workstation's log file.

Upon receipt of this request, the GCM will perform he following:

- Writes the disconnect request into its log file and then acknowledges the disconnection by replying Ack_Disconnect back to the mobile workstation. It then updates its *mws* status table that is kept in storage.

The overall procedural flow in the management of mobile workstations is shown in Figure 3.

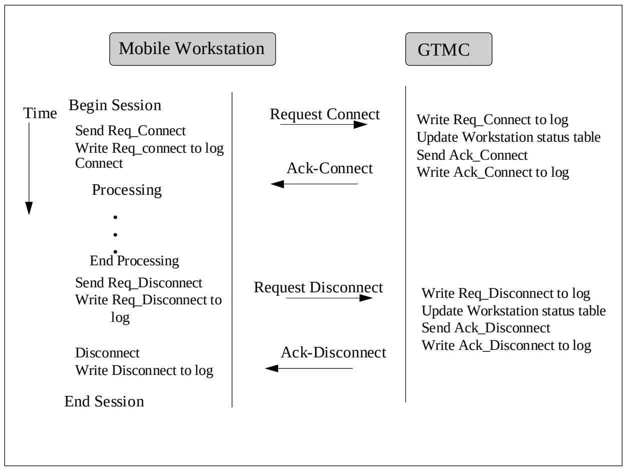

Fig.3. Workstation Connection Procedural Flow

Assuming that during a session with the host computer, an *mws* is disconnected from the network, for example, when a network failure is encountered. During its subsequent connection to the host computer, the *mws* would have detected from its *mws* status table stored locally that it has not performed an orderly disconnection. Hence, the *mws* would send a Req_Reconnect to the coordinating site together with the last global transaction submitted (if any).

Upon receipt of Req_Reconnect, the GCM can then verify from its Global Log that the last global transaction submitted (if any) is correct. The GCM would send the following messages back to the *mws* :

- Ack_Reconnect_Mws
- Info_Msg_Queue
- Info_Queue_Status
- Info_Mws_status

If there is a total failure of the mobile workstation, during subsequent connection to the coordinating site, its GCM would have detected that the *mws* is still in session according to the *mws* status table. The GCM would send the following messages back to the *mws* :

- Ack_Reconnect_Mws
- Info_Msg_Queue
- Info-Queue_Status
- Info_Mws_status

Based on this information returned, the end-user can then decide the follow-up actions to be taken.

The second type of failure that can arise, concerns the MDSTPM software itself and is analogous to a site failure. Such a failure will affect both the execution of global transactions and global subtransactions. Recovery from site failures in the proposed scheme can be handled properly. Upon recovery from a site failure, the GTM would perform the following:

- Recovers the site status table from the shadow site status table.
- Re-establishes the site status table by sending a probe packet (Req_Site_Status) to all participating sites.
- Rebuilds the temporary virtual global MDS schema.
- Reads the global log file and repeatedly submits those global subtransactions which are in the prepared state to their respective LTMs for execution if their global transactions have already been committed.

The management of global transactions/subtransactions through the use of transaction queues will be discussed next.

5.3 Transaction Management

To manage the transactions submitted by mobile workstations, a simple but effective global transaction queuing mechanism is proposed. The basic principle behind our queuing mechanism is the concept of finite state machines. This is because we can clearly define a set of possible state and transitions from one state to another during the *life span* of a global transaction, where by life span we mean actions or subtransactions within the primitives BEGIN_GLOBAL_TRANSACTION and END_GLOBAL_TRANSACTION. Hence the key to implement the proposed model is to design a queuing facility that maps exactly each of these states. Altogether, there are five transaction sub-queues that are used to manage global transactions/subtransactions submitted to the local site by the mobile workstation.

- *Input Queue*. This queue contains all the global transactions/subtransactions that first arrived at the coordinator node. It is in monotonically increasing order based on transaction arrival time.
- *Allocate Queue*. Global transactions/subtransactions will be selected for execution based on a first-in-first-out (FIFO) basis or a priority based scheduling algorithm. All the locks required for the global transactions / subtransactions will be acquired during this stage.
- *Active Queue*. This queue contains all currently active global transactions/subtransactions.
- *Suspend Queue*. This queue contains all global transactions/subtransactions that have completed the first phase of the two phase commit protocol.
- *Output Queue*. This queue contains all completed global transactions.

When a global transaction is being submitted by a mobile workstation, the GCM will place the global transaction into the *Input Queue*. Periodically, the GSS will scan the *Input Queue* and select a global transaction for execution. Once selected, the global transaction will be transferred to the *Allocate Queue* where all required locks will be acquired by GCS. The global transaction will then be transferred to the *Active Queue* where global subtransactions to be executed at other sites will be despatched by the GCM. Once the global transaction has completed its first phase of the two phase commit protocol, it is then placed into the *Suspend Queue*. Upon completion of the two-phase commit, the global transaction is then placed into the *Output Queue*.

Once a transaction has been selected for execution by the GSS, the necessary concurrency control requirements are then acquired and managed by the GCS.

To ensure the consistency of database objects from interleaved execution of multiple concurrent transactions, concurrency control mechanism is used to isolate a transaction from the effects of all the other concurrently executing transactions. Serializability is the usual correctness criterion used for the design of concurrency control mechanism [3].

One of the main problems encountered in maintaining serializability of global transactions is how to ensure that the execution order of global transactions is preserved by the LTM. Georgakopolous, Rusinkiewicz and Sheth [8] have shown that local conflicts between global subtransactions and local transactions may change the order of execution of global transactions. A ticketing method is used to resolve this problem. All global subtransactions are forced to obtain a ticket first thereby causing additional conflicts among themselves consequently preserving their execution order [13].

As the use of mobile workstations becomes a new computing paradigm, transaction management is of more and more importance, especially in a cooperative multidatabase processing environment. We have seen that preserving local database autonomy in an MDS environment is difficult and has given rise to numerous problems in the design of transaction management scheme. We have also examined the various approaches to transaction management in a distributed MDS environment. However, there are tradeoffs made by these various approaches that can be classified into two broad categories. First, the level of autonomy is relaxed which results in changes being made to the local DBMSs. Second, some of the ACID properties are compromised and alternative correctness criteria are suggested. However, it is important to note that all these approaches assumed a failure-free system. We have also discussed that the use of mobile computing has given rise to a new set of challenges to the database researchers.

6 Summary and Future Work

In this paper, we have introduced a generalised MDSTPM architecture that provides an autonomous transaction management strategy in a cooperative processing environment across heterogeneous database systems. The main focus of this architecture is to provide an underlying framework to support mobile computing that can achieve a high degree of distributed transaction parallelism and independence. The fundamental feature of our approach is that it decomposes the operating environment into two separate entities: the mobile workstation level and at the host computer or stationary computer level.

A Message Queuing Facility is proposed to manage mobile workstations effectively in a distributed MDS environment. As messages are being handled asynchronously, the mobile workstations can perform some other tasks while their pre-assigned coordinating nodes coordinate and execute transactions on their behalf. However, there are several issues that require further investigations including the implementation of the MQF and the development of a prototype system to support the execution and recovery of multidatabase transactions. It is also important to note that for an MDS to function correctly, it is necessary to establish an

MDSTPM component software at each site in order to facilitate the integration.

In conclusion, the two computing environments - multidatabase systems and mobile computing - can be integrated in harmony to provide a rich solution to emerging requirements for cooperative processing computing.

Acknowledgments

Liu Hui Yeo wishes to thank the Australian Wheat Board and especially Mr Howard Smith for the support of this research programme. Authors are thankful to Professor Phillip Steele for the provided opportunities, encouragement, advice and necessary facilities for this research. The views and opinions expressed in this paper are solely that of the authors and do not represent that of the Australian Wheat Board. Authors would like to thank the referees for their useful comments that helped to improve this paper.

References

[1] Alonso,R. and Korth,H.F. "Database System Issues in Nomadic Computing", *ACM SIGMOD*. 22:2, 1993, p.388-392.

[2] Atre,S. *Distributed Databases, Cooperative Processing, & Networking*. New York:USA. McGraw-Hill, 1992.

[3] Bernstein,P.A., Hadzilacos,V. and Goodman,N. *Concurrency Control and Recovery in Database Systems.* Reading, Mass : Addison-Wesley, 1987.

[4] Breitbart,Y., Silberschatz, A. and Thompson,G. "An Update Mechanism for Multidatabase Systems". *Database Engineering*. (Eds.) W.Kim, M.Carey, S.Sarin and C.Zariolo. IEEE Computer Society Press, 1990, 150-156.

[5] Du,W., and Elmagarmid,A.K. "Quasi Serializability : A Correctness Criteria for Global Database Consistency in InterBase", in *Proceedings of the Fifteenth International Conference on Very Large Databases*, 1989, 347-356.

[6] Elmagarmid,A.K. and Rusinkiewicz,M. "Critical Issues in Multidatabase Systems". *Information Sciences*. 57-58, 1991, 403-424.

[7] Garcia-Molina,H. and Salem,K. "Sagas" in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1987, 249-259.

[8] Georgakopolous,D., Rusinkiewicz,M. and Sheth,A. "On Serializability of Multidatabase Transactions Through Forced Local Conflicts" in *Proceedings of the Seventh International Conference on Data Engineering*, 1991 Kobe Japan, p. 314-323.

[9] Rowe,L.A. and Stonebraker,M. "The POSTGRES Data Model" in *Proceedings of the 13th International Conference on Very Large Databases*. Brighton, England. Palo Alto:California. Morgan Kaufmann, 1987, 83-96.

[10] Sheth,A.P. and Larson,J.A. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys,* Vol.22, No.3, 1990, 183-236.

[11] Voss,F.W. "APPC/MVS Distributed Application Support". *IBM Systems Journal*. 31:2, 1992, 381-408).

[12] Yeo, L.H. and Zaslavsky, A. "Transaction Management in Multidatabase Systems" *Technical Report Number 93-02*. SCITF, Monash University, Melbourne, Australia, 1993.

[13] Yeo,L.H. and Zaslavsky,A. "Layered Approach to Transaction Management in Multidatabase Systems" in *Proceedings of the 5th International Hong Kong Computer Society Database Workshop: Next Generation Database Systems,* 1994, 179-189.