M-Commerce Transaction Management with Multi-Agent Support

M. Younas, K-M Chao, R. Anane
School of Mathematical and Information Sciences,
Coventry University, United Kingdom
Email: {m.younas, k.chao, r.anane}@coventry.ac.uk

Abstract

Transaction management is a major issue in Mobile Commerce (M-commerce). It enables people to order goods and access information in an anywhere and anytime fashion. Given the nature of mobile computing, there is a need for a generic approach that adapts to the needs of M-commerce applications. In this paper we propose a new multi-agent approach that dynamically manages mobile transactions which adapt to the processing environment of M-commerce. We also present techniques that deal with the issues of handoff, disconnection, or system failures. Potential advantages of the proposed approach include enhanced flexibility, improved performance and fault tolerance.

1. Introduction

Recent advances in mobile computing devices have led to the emergence of M-Commerce. However, mobile computing devices are generally resource-scarce as they have limited bandwidth, less reliability and limited computational resources. These characteristics of mobile devices complicate the process of transaction management (TM) in M-commerce. Such TM may be quite diverse with probably many different transaction models and processing modes required. Some may follow traditional ACID (atomicity, consistency, isolation, durability) criteria while others may use extended transaction models (ETM) [1, 2]. ACID transactions follow the strict isolation policy, which does not permit the exposition of the intermediate results of transactions. Various ETM (such as open and closed nested, split and join transaction models) are defined to relax this strict policy [3].

Current solutions to M-commerce TM have limited scope, as they are applicable to specific applications and mobile devices. However, there is a need for TM techniques in which each transaction could be managed dynamically based on its behaviour and also on the behaviour of mobile devices [4]. In this context, the flexibility, adaptability, and high-level semantics afforded by multi-agents provide a useful framework for handling M-commerce applications. Agents are autonomous entities, which possess a high potential for practical applications in different areas such as air- traffic control, E-commerce, etc. The most common architecture of agents is the BDI (Beliefs, Desires and Intentions) [5]. In BDI model, *Beliefs* represent the external and internal informational state of an agent. *Desires* are agent's motivational state, that is, what the agent is trying to achieve. A typical BDI agent has a procedural knowledge constituted by a set of plans, which defines sequences of actions to be performed to achieve a certain goal or to react to a specific situation. The *Intentions* represents the deliberative state of the agent, that is, which plans, the agent has chosen for eventual execution. In this paper we propose a multi-agent approach that dynamically manages mobile transactions such that they adapt to the needs of M-commerce applications.

Section 2 describes the characteristics of mobile transactions. Section 3 reviews related work. Section 4 presents the proposed approach. Section 5 concludes the paper and identifies future work.

2. Characteristics of Mobile Transactions

The main characteristics of mobile transactions are illustrated through the architecture of mobile computing shown in Figure 1 [6, 7]. It comprises different components. *Fixed-hosts* (FH), *Base stations* (BS) or *mobile support stations* (MSS) — these are computer systems, which are inter-connected via wired network. BSs are capable of communicating with mobile units through wireless interface. Mobile units are portable computers or WAP-enabled mobile phones. BSs also act as an interface between mobile units and systems of wired-network. Operations of multiple BS are coordinated by *Base Station Controller* (BSC). BSC are in turn controlled by the *Mobile Switching Centre* (MSC), which is connected to the *Public Switching Telephone Network* (PTSN). Each BS covers a particular area, called a *cell*. Mobile units can move from one cell to another. This process of moving is referred to as a *handoff*.

The nature of mobile computing complicates the process of mobile TM as described below:

Proceedings of the17 th International Conference on Advanced Information Networking and Applications (AINA'03)
0-7695-1906-7/03 $17.00 © 2003 IEEE

Complexity of Mobility: In mobile computing, data, applications, and also the processing units (such as laptop) move. This mobility greatly complicates the process of TM. For example, transactions may move from one location to another as their originating devices move. Thus they may originate at one site and terminate at other site. Similarly, physical location of data may change as the mobile units carrying those data move from one location to another. Further, the handoff process also complicates TM, as handoff is generally unpredictable. Thus, it may affect the termination (i.e., commit or abort) process of a transaction. This is because the location of the desired mobile device associated with a transaction may not be immediately available for communication [7].

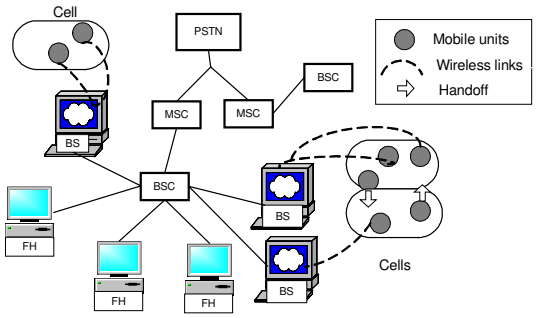

Figure 1: An Architecture of Mobile Computing

Low Reliability and Availability: Mobile units are frequently unavailable due to many reasons. First they have limited power supplies. This creates serious issues for TM. For example, a mobile unit having a flat battery may not communicate with other units involved in the processing of mobile transaction. Further, the mobile unit may run out of its disk space, or it may affect by break down, and so on. All these events would affect the functionality of mobile units, and consequently the processing of mobile transactions. Thus it is difficult to assume that a mobile unit is always available to participate in processing a mobile transaction [7].

Limited Bandwidth: Current mobile technology is associated with a limited bandwidth. Although the bandwidth is increasing, it is assumed that it will remain limited compared to the wired network. The consequence is that communication links may be overloaded with high volume of information exchange. This limitation creates complexities for mobile transactions. For example, transaction related messages cannot be promptly communicated between the participating systems.

3. Related Work

Various research studies have been conducted to examine issues of transaction management in mobile computing (see [8] for the detailed review of the related work). The Kangaroo Transaction (KT) model [9] takes into account the movement behaviour of mobile transactions such that they can hop from one base station to another as their mobile device moves. Such hoping of transactions is modelled through the use of split transaction model [10]. In KT model, a transaction splits only when it hops from one base station to another. Splitting is not allowed if transaction remains in the same cell. KT model aims at reducing message overhead as follows. When a mobile transaction moves to a new cell, the control of the transaction also moves to the new cell. If it remains at the originating site, however, messages would have to be sent from the originating site to the current base station whenever the mobile unit requests information. To avoid this message overhead, the transaction management function should move with the mobile unit. A pre-serialization technique for managing transactions in mobile multidatabase environment is proposed in [11]. This approach aims at maintaining isolation property of mobile transactions. The benefits claimed include supporting disconnection and maintaining transaction isolation. However, the main limitation of this approach is that these benefits are gained at the cost of extra overhead. The approach presented in [12] proposes a pre-write operation technique. The main objective of this approach is to increase data availability and reduce transaction processing on mobile devices. In this technique, transaction is executed on the mobile device using operations such as pre-writes, and pre-commit. Pre-write operation is performed on the local data items locked by the mobile device. Pre-commit is used to locally commit the transaction at the mobile device. After local commit, pre-writes of the data items are sent to the database, which makes them permanent and commits the mobile transaction. Moreover, there exist various approaches in which multi-agent techniques are applied to manage transactions in E-commerce or Web environment [13, 14]. However, these approaches do not consider mobile computing and are limited to the classical distributed computing with wired-network connections.

Although current approaches contribute to the mobile transactions research, they do have limitations. For example, these approaches do not consider the movement of data storage devices wherein a data may move from one cell to another along with the movement of their storage devices. However, such movement of data is considered as a key requirement for the success of M-commerce applications as in a ubiquitous database approach [15]. Further, current approaches (such as KT model) consider the movement behaviour of transactions. But if the originating-device of a transaction moves back to the previous base station (or cell), then the information related to the transaction management needs to be sent

back to the previous station. Thus it may not help in reducing communication overhead.

In summary, current approaches have limited scope as they are associated with specific applications, having fixed semantics, and fixed transaction correctness criteria. Thus there is a need to have transaction management mechanisms that adapt to the processing environment of mobile computing [4].

4. Agent-based Mobile Transactions

This section presents a TM model that relies on multi-agents in order to conform to the needs of M- commerce.

4.1 Basic Definitions

Definition 1*: A mobile transaction, T_m, is defined as a composition of site transactions each of which represents an activity (e.g., booking a flight).*

Each site transaction is a sequence of operations, which concludes with either an abort or a commit operation. In the proposed approach agents execute different site transactions, ST, so as to achieve the goal, e.g., buy a holiday package. Agents construct plans for the execution of site transactions to achieve a goal or react to a specific situation. Plans are executed according to the agent's intention. An agent processing a mobile transaction can be defined as follows:

Definition 2: *An agent is defined as ((ST, T_m), P, SD), where ST is a set of site transactions of a mobile transaction, T_m, and P is the plan for ST. SD is a set of structural dependencies.*

Agents execute specific operations, depending on the nature of processing environment. For example, an agent may execute a split operation to split a mobile transaction T_m into multiple independent transactions. Splitting a transaction is useful for long running mobile transactions so as to improve concurrency and recovery. Further, agents must determine the relationship between different mobile transactions and their site transactions. These relationships are determined using the structural dependencies (SD) [3]. Knowledge of structural dependencies is important for agents in order to dynamically manage mobile transactions. Some of these dependencies are defined as follows (see [3] for details]).

Definition 3 *(Execution)*: *Execution dependency (Ord, ST, P) defines the execution order, Ord, between the site transactions ST according to plan P. For example, ST_i and ST_j may execute in parallel or sequentially.*

Definition 4 *(Triggering): In triggering structural dependency (ST, P, E) agent may trigger a plan P if an event E occurs. For example, if ST_i, is aborted then agent needs to activate (trigger) an alternative site transaction, ST_j, to replace the aborted ST_i.*

4.2 The Execution Model

The execution model of the proposed approach is illustrated through the architecture shown in Figure 2. In the model, different agents cooperate with each other in order to execute a mobile transaction. These agents include *planning* agents, *execution* agents, and *recovery* agents. The responsibility of *planning* agent is to devise local plans so as to execute a site transaction. Planning agents delegate the plans to the *execution* agents, which executes plans for site transactions. *Recovery* agents provide recovery facilities in the case of failures.

General responsibilities of the different agents involved in processing the mobile transaction include: submitting site transactions to the databases or sending them to other agents, handling disconnection and migration of mobile units, logging information for recovery purposes, and enforcing the transaction correctness. In order to keep track of a mobile transaction agents must have all the necessary information such as IDs of a mobile transaction and its sites transaction, status of the mobile transaction, list and locations of the data sites, and the location of site transactions.

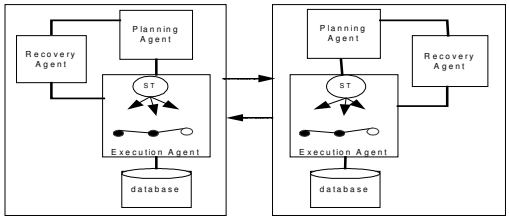

Figure 2. Architecture for Multi-Agents Mobile Transactions

When a user submits a mobile transaction to the respective agents using his/her mobile unit, the system devises a meta-plan based on the user's preferences and desire. The meta-plan comprises different possible plans that fulfill the user request. The meta-plan contains all the information required to achieve a goal, (e.g., book a holiday). Once the plan is initiated by the system, agents start gathering relevant information for the achievement of the goal. Agents use information of the meta-plan to generate an appropriate Applicable Plan List (APL), which is comprised of different possible plans. It is assumed that each plan corresponds to a site transaction. In order to maintain the application correctness agents must process all plans in a consistent way. That is, either all of them are successfully completed, or none of them is completed. In this case, all the plans for site transactions must accomplish their assigned tasks. If all the required

plans are successfully completed then the mobile transaction is committed, otherwise it is aborted.

Depending on the situation, agents may use different transaction models to process site transactions. For example, if a site transaction is compensatable, then open-nested transaction model [3] can be used. If a site transaction is non-compensatable then ACID criteria are suitable. Similarly, agent may split a mobile transaction into different independent transactions. Agents use structural dependencies to dynamically choose appropriate transaction model for processing mobile transactions. For example, if two site transactions have no dependencies then they can be split into two independent transactions enabling them to commit or abort independently.

To deal with issues of system failures, disconnection, or handoff we incorporate the following techniques:

Timeout: Timeout technique deals with uncertain situations where systems cannot communicate with each other due to failures [7]. We incorporate the timeout technique to deal with system failure or disconnection. In this technique, a timeout value is set by mutual agreement of the respective agents before the execution of a transaction. After the expiry of the timeout interval agents are free to perform the necessary actions (e.g., rollback, compensation, or replacement) so as to properly terminate mobile transaction. Timeout technique deals with the situation when the mobile device or the fixed host is not available during a specified time.

Disconnection: Due to resource scarcity frequent disconnection occurs during the processing of mobile transactions. The technique for dealing with disconnection is also based on the timeout. For example, agents on the mobile device may use this technique to redefine the timeout interval. If the disconnected agent can reconnect within the specified timeout interval, then it can continue execution of a transaction. Otherwise, the transaction is rolled back. Recovery agents deal with these issues.

Handoff: When a mobile unit moves from one cell to another, a handoff process is performed. During this process a mobile unit has to inform current base station the address of its subsequent new base station. After the connection is established with the new base station, mobile unit also needs to tell address of its previous base station. This ensures that both the new and previous base stations have all the required information of the mobile device and its associated mobile transaction.

5. Conclusion and Future Work

We have proposed a new multi-agent approach that dynamically manages mobile transactions in M-commerce. We have also presented techniques for dealing with issues of handoff, disconnection, and system failures. The proposed approach is partially implemented. Initial evaluation shows the following improvements: (i) Mobile transactions are dynamically managed using different transaction models. (ii) Fault tolerance and service availability is improved using alternative transactions. (iii) Concurrency is improved using open-nested and split transaction models, wherever appropriate. Our future work includes fuller implementation of the proposed approach in order to see its complete effectiveness.

6. References

[1] A.K.Elmagarmid "Database Transaction Models for Advanced Applications" *Morgan Kaufman, 1992*.

[2] M. Younas, B. Eaglestone, R. Holton "A Review of Multidatabase Transactions on the Web: From the ACID to the SACReD" *British National Conference on Databases (BNCOD),* Exeter, UK, Springer LNCS, 2000, pp. 140-152.

[3] P.K. Chrysanthis, K. Ramamritham "Synthesis of Extended Transaction Models using ACTA" *ACM Transaction on Database Systems*, Vol. 19, No. 3, Sept. 1994, pp. 450-491

[4] M.H. Dunham, V. Kumar "Impact of Mobility on Transaction Management" ACM Int. Workshop on Data Engg. for Wireless and Mobile Access, August 1999, Seattle, USA.

[5] N. R. Jennings "On Agent-Based Software Engineering" *Artificial Intelligence*, Vol. 117, No 2, 2000, pp. 277-296.

[6] A. Helal, B. Haskell, J.L. Carter, R. Brice, D. Woelk, M. Rusinkiewicz *"Any Time, Anywhere Computing: Mobile Computing Concepts and Technology"* Kluwer Publisher, 1999

[7] V. Kumar, N. Prabhu, M. Dunham, Y.A. Seydim,"TCOT - A Timeout-based Mobile Transaction Commitment Protocol", *IEEE Trans. on Computers*, Vol. 51, No. 10, October 2002.

[8] P. Serrano-Alvarado, C. Roncancio, M.E. Adiba "Analyzing Mobile Transaction Supports for DBMS" *4th Int. Workshop on Mobility in Databases and Distributed Systems, DEXA 2001*

[9] M.H. Dunham, A. Helal, S. Balakrishnan, "A Mobile Transaction Model That Captures Both the Data and Movement Behavior", *ACM/Baltzer Journal on Special Topics in Mobile Networks and Applications (MONET)*, Vol 2, 1997

[10] C. Pu, G.E. Kaiser, N. Hutchinson "Split Transactions for Open-Ended Activities" *VLDB* 1988.

[11] R.A. Dirckze, L. Gruenwald "A Pre-Serialization Transaction Management Technique for Mobile Multidatabases" *Mobile Networks and Applications* 5 (2000), pp. 111-321

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

[13] Q. Chen, U. Dayal *"Multi-agent Cooperative Transactions for E-Commerce" 7th Int. Conference on Cooperative Information Systems (CoopIS)*, Eilat, Israel, LNCS 1901, 2000

[14] M. Younas, N. H. Shah, K-M Chao "A Multi-Agent Approach to SACReD Transactions for E-Commerce Applications" *3rd Int. Conf. on Electronic Commerce and Web Technologies (EC-WEB)*, Sept. 2002, Aix en Provence, France

[15] K. Kuramitsu, K. Sakamura " Towards Ubiquitous Database in Mobile Commerce" 2nd ACM *International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE01),* May 20, 2001 in Santa Barbara, California, USA