Comparing transaction commit protocols for mobile environnements

Christophe Bobineau, Cyril Labbé, Claudia Roncancio, Patricia Serrano-Alvarado
LSR-IMAG laboratory
BP 72, 38402 St. Martin d'Hères, France
e-mail: FirstName.LastName@imag.fr

I. Introduction

The omnipresence of mobile devices and wireless networks lead to a growing interest in supporting a wide variety of applications in mobile environments. Numerous efforts in providing appropriate data managements for such environments are made [9]. Transaction supports have been revisited to propose adapted transaction models and properties [7], [8], [6], [3] (see [11] for a survey). Proposed algorithms and protocols try to optimize the use of mobile units resources and to overcome wireless network limitations (e.g. disconnections).

This work concerns protocols to commit transactions distributed over several mobile and fixed units. It is the first step of a comparative study of that kind of protocols. Such a study intends to provide a global picture of their characteristics, advantages and drawbacks. Qualitative and quantitative aspects should be analyzed with respect to mobile environment characteristics and application requirements. This paper does not provide such a comprehensive study but reports results on a rather quantitative comparison of some commit protocols.

We analyze three commit protocols: *Two Phases Commit* (2PC) [4], *Unilateral Commit Protocol for Mobile and Disconected Computing* (UCM) [1], [2] and *Combination of Optimistic approach and 2PC* (CO2PC) [12], [10]. 2PC and UCM guarantee atomicity whereas CO2PC insures semantic atomicity. These protocols are representative of current proposals but the study should be enhanced to other protocols like *Transaction Commit on TimeOut* (TCOT) [5].

2PC, UCM and CO2PC are briefly introduced in sections II, III and IV respectively. We present their state-transition diagram (see figure 1 for the notation) and their main characteristics. Section V compares some performance indices regarding to the number of mobile units involved in a transaction and the connectivity characteristics of the environment. The analysis focuses on the transaction validation phase. The impact of the protocols on the performances during a transaction execution itself is not yet considered. Section VI presents our main conclusions and future work.

II. 2PC Protocol

Description and Properties: The 2PC protocol insures atomic commit of transactions distributed on several databases. It has not been designed for mobile environments, however it is used by some transactional proposals for those environments. *Global transactions* involve several sub-transactions (called *local transactions)*, each executed on one of the *participant* databases – Fig. 2(B). Transaction validation is initiated by a message sent by the application to the 2PC's *coordinator* – Fig. 2(A). Participants vote commit/abort during a *voting phase*. The coordinator takes then a global decision during the *decision phase*. 2PC insures a correct application of the decision: if a global commit (respectively abort) is decided then all concerned local transactions commit (respectively abort).

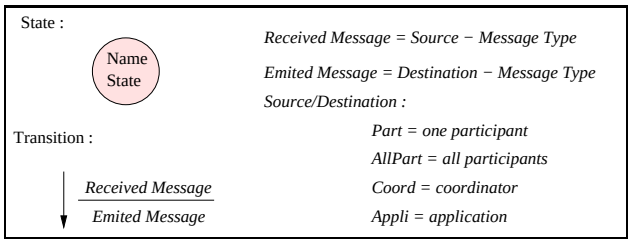

Fig. 1. State-diagrams' legend

Inaccurate global decision: A 2PC coordinator may decide a global abort in some cases when a global commit is eventually possible. This arises if the coordinator does not receive all the votes before its timeout expiration – *Wait Vote i* states in Fig. 2 (A).

Blocking situations: Two blocking situations can arise in the 2PC protocol. (i) The coordinator stalls until reception of all *Ack* messages from the participants – *Wait Ack Commit i* and *Wait Ack Abort i* states in Fig. 2 (A).
This situation isn't too damageable, since no data is blocked.
(ii) A participant stalls after voting commit until reception of the global decision. This situation may be constraining as participant's local resources remain locked during that time. Such a participant is not allowed to unilaterally terminate the local transaction.

III. UCM Protocol

Description and properties: UCM has been specifically designed for mobile environments. To insure atomicity, transaction operations and their acknowledgments are continuously logged. If a problem arises the global transaction is immediately aborted. Therefore, if a global transaction reaches the validation phase the global decision is *commit*. There is no risk of inaccurate global abort.

In UCM, validation is a single phase : the *decision phase*. It is initiated by the transaction's operation log

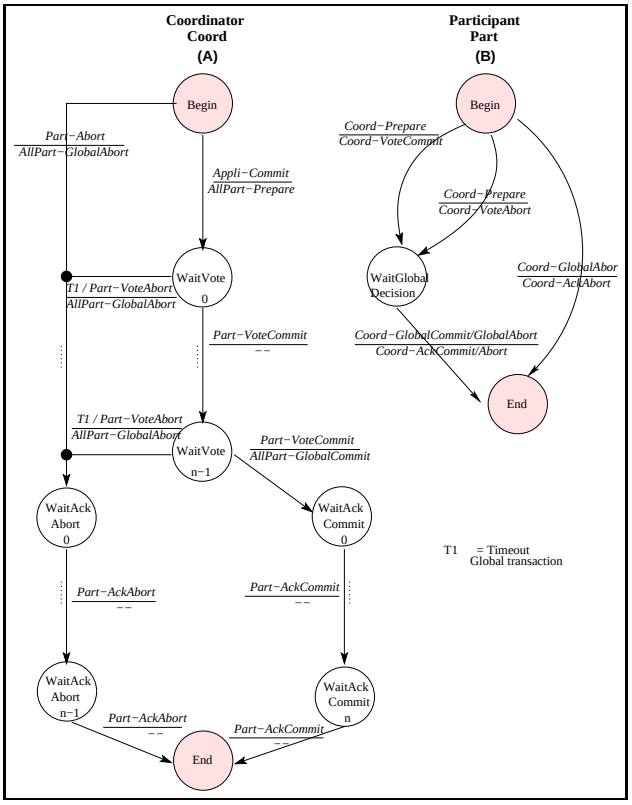

Fig. 2. State-diagram for the 2PC protocol

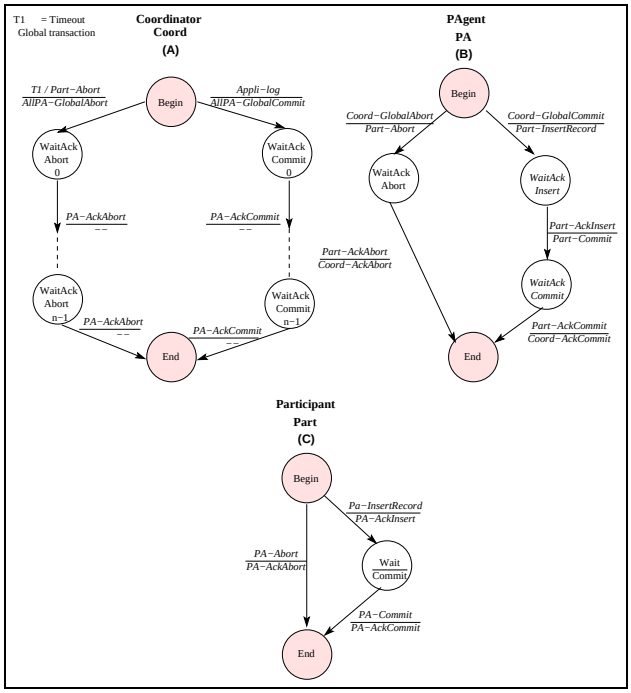

Fig. 3. State-diagram for the UCM protocol

transfer from the application to the coordinator. Database participants are represented by proxies to interact with the UCM coordinator – Fig. 3(B) and (C). This choice improves DB's autonomy and insures proper recovery in case of participant failure – message *InsertRecord* in fig. 3(B) and (C).

Blocking situation: As for 2PC, an UCM coordinator stalls if at least one *Ack* message is missing. But, as for 2PC, this situation is not damageable for the system.

IV. CO2PC Protocol

Description and properties: CO2PC was also specifically designed for mobiles environments. Global transactions may be composed by compensable local transactions and non-compensable ones[1]. CO2PC insures semantic atomicity. Compensable local transactions can commit/abort unilaterally in an anticipated manner, whereas non-compensable ones have to wait for the CO2PC coordinator's decision.

In CO2PC, participants (represented by proxies) send their votes (*commit* or *abort*) to the coordinator, who decides a global commit if there is unanimity and a global abort otherwise. The way participants behave wrt. to these message exchanges depends on the type of local transaction they execute (compensable or not).

Compensable local transactions: participants executing such transactions are represented by an *OptProxy* – fig. 4(B). It submits the local transaction to the underlying DBMS – called *OptPart* in fig. 4(C). Once the execution finishes, the local transaction is unilaterally committed or aborted. Then, the *OptProxy* sends the corresponding vote to the CO2PC coordinator. Later, if the global transaction has to be aborted, a compensation local transaction is executed on the *OptPart*.

Non-compensable local transactions: participants executing such transactions are represented by a *NonOptProxy* – fig. 4(D). It submits the local transaction to the underlying DBMS – called *NonOptPart* in fig. 4(E). Once the execution finishes, the *NonOptProxy* coordinates a 2PC protocol with the underlying DBMS as unique participant. The 2PC vote from the *NonOptPart* is propagated to the CO2PC coordinator. If the vote is abort, the *NonOptProxy* unilaterally decides the abortion of the local transaction (decision for the local 2PC). If the vote is commit, the *NonOptPart* enters the *prepare* state – *Wait Global Decision* state fig. 4(E). Once CO2PC coordinator's decision arrives to the *NonOptProxy*, it is transmitted to the *NonOptPart* where it is interpreted as the local 2PC decision.

Inaccurate global decision: The CO2PC coordinator might decide an inaccurate global abort in the same condition as 2PC does: at least one vote is missing at timeout.

Blocking situations: Participants and coordinator have timeouts to limit blocking. Nonetheless (i) the coordinator stalls, as for 2PC and UCM protocols, until it receives all *Ack*. (ii) Since a *NonOptPart* participates to a local 2PC protocol to validate its local transaction, this kind of participants inherits the blocking situation of 2PC: it may stall after a commit vote. An *OptPart* never stalls as it unilaterally terminates its local transaction in an eager manner.

[1]2PC and UCM protocols assume only non compensable local transactions

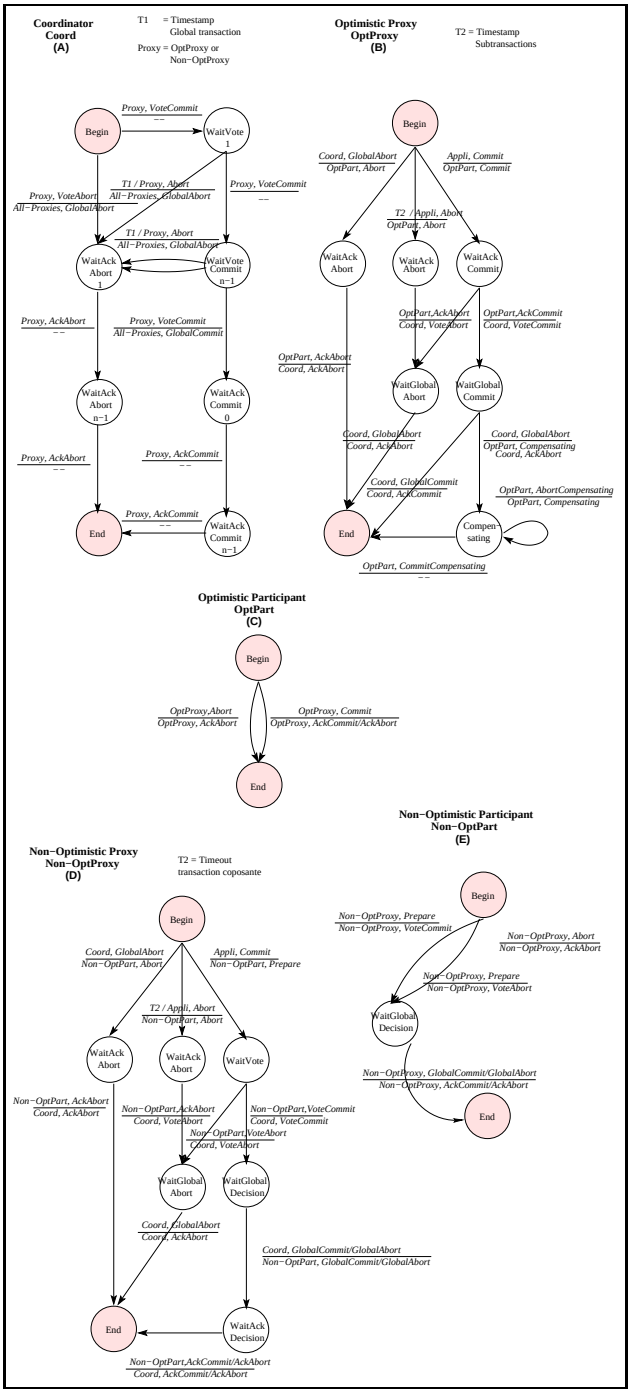

Fig. 4. CO2PC state-diagrams

V. Protocols comparison

A. Global conditions

This section compares some performance aspects of the mentioned commit protocols. Our study concerns the commit phase begining with the first message appearing in the diagrams presented before. In the case of CO2PC this supposes some kind of agreement of the participants to start the commit protocol. A more extensive study should eliminate this hypothesis. Furthermore, CO2PC authorizes participants executing compensable transactions to liberate their resources in an eager manner. Such advantage is not considered by our study and should be in future work.

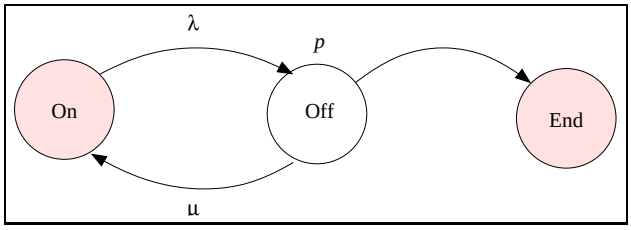

Fig. 5. Participant behavior : On/Off state automaton.

Performance Indices: This study considers the number of mobile units involved in a transaction and the connectivity characteristics of the environment. Three performance indices are evaluated: mean validation time, blocking probability of the coordinator and probability of inaccurate global abort. As seen in preceding sections, the last point concerns 2PC and CO2PC. Nevertheless such probability is tightly related to the choice of the timeout value. There is a tradeoff between the validation mean time and the probability of inaccurate global abort (short timeouts increase such probability). In our simulations timeouts are fixed by considering the minimum time to exchange required messages plus 50% of this time as security delay.

Network Hypothesis: Mobile participants are supposed to communicate with the coordinator using a not reliable network and are allowed to quit the system definitively without advertising the coordinator.
These facts are modelized by a two-states automaton, On/Off – fig. 5. For a participant, its connection time is randomly chosen according to a negative exponential distribution of parameter λ. Mean time in state On is $\frac{1}{\lambda}$. Disconnection times are also randomly chosen according to a negative exponential distribution of parameter μ. Mean time in state Off is $\frac{1}{\mu}$. When a participant leaves the state Off, it may either leave the system definitively with probability p, either return to state On.
Simulations reflect network latency by applying a constant delay to message transfers. A participant cannot send a messages if it's connection time (time before next disconnection) is not long enough.

Simulation platform: Compared protocols have been simulated using SimJava [13]. Simjava is a process based discrete event simulation package for Java. A simjava simulation is a collection of entities (named *sim_entities*) each running in its own thread. These entities are connected together by ports and can communicate with each other by sending and receiving event objects. A central system class controls all the threads, advances the simulation time, and delivers the events. The progress of the simulation is recorded through trace messages produced by the entities and saved in a file.
In our simulations a participant is represented by two entities. One for its validation process and one for its behavior. This last entity use two types of random number generator classes. The *Sim_uniform_obj* is used for uniformly distributed numbers generation. This to decide when a participant leave the system definitively. The *Sim_negexp_obj* is used for negative exponential distributed

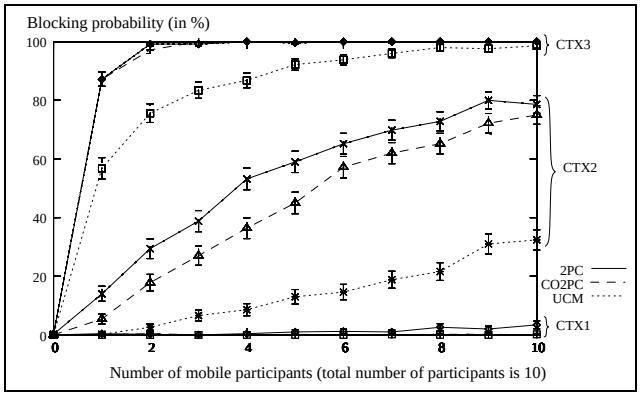

Fig. 6. Probability of blocking vs number of mobile participants.

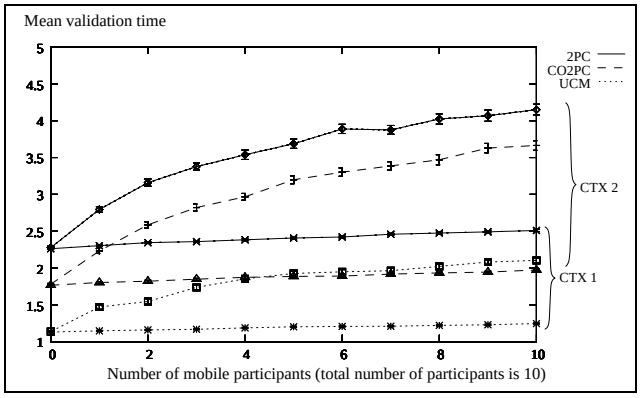

Fig. 7. Mean commit time vs number of mobile participants.

numbers generation that gives the time spend in states $On$and Off. The *sim_entity* that simulated the validation process of a participant sends *vote* and *ack* or *leave* events to the entity used for the coordinator simulation. This last entity drives the simulation, collects events and records statistics on blocking situations, mean validation time and inaccurate global aborts.

B. Mobile participants

Three contexts representing different degradated mobile environments were simulated. The probability of a participant to leave the system definitively was uniformly fixed to $p = 5\%$.[2] If a participant is connected to the system, in the first context (named *CTX1*) it has 90% chances of being connected (state On). Conditions of CTX1 are not excellent. Conditions are even more degradated in contexts 2 and 3 (*CTX2* and *CTX3*). The connection probability is respectively 50% and 10%. The three performance indices introduced before where evaluated for the three contexts by varying the number of mobile participants from 0 to 10 (the total number of participants is 10).

Probability of blocking: Fig. 6 shows the probability of blocking in the three contexts. The three compared protocols perform quite well in context CTX1. For UCM and CO2PC such probabilities are under 0.2%. 2PC is around 3%. This value is not negligible. The three protocols perform badly in the degradated contexts CTX2 and CTX3. In CTX2, UCM with 2 mobile participants has a blocking probability around 3%. In CTX3, for 2 mobile participants (among 10), the blocking probability exceeds 75% for the three protocols.

[2]This value is quite high.

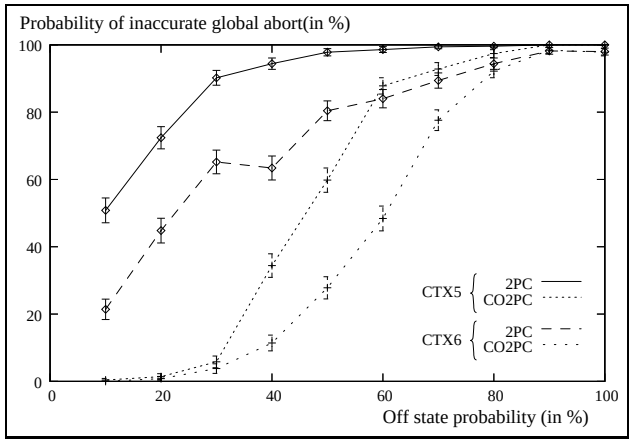

Fig. 8. Probability of inaccurate global abort vs probability of being disconnected (state Off).

Mean commit duration time: As the number of blocked commits is extremely high in CTX3, the estimation of the commit mean time makes no sense there. Fig. 7 reports mean commit time in contexts CTX1 and CTX2. In CTX1 the number of mobile participants has a week impact on the mean commit time and, once again, the three protocols perform well. The best performances are for UCM, followed by CO2PC and then 2PC. These results confirm intuition as commit duration depends mainly on messages sent over the wireless network. In CTX2, message expeditions are often delayed because the connection time is not long enough. This increases the mean commit duration time.

C. Highly degradated mobile environments

We consider here three new contexts – *CTX4*, *CTX5* and *CTX6* – where participants are supposed to be very instable. In CTX4, CTX5 and CTX6 the probability of a participant to leave the system definitively is respectively 5% , 15%, and 30%. So CTX6 presents the worst conditions. The probability of a participant to be disconnected (state Off) ranges between 10% and 100%. The three performance indices introduced before have been evaluated. In the following we report the main results.

Probability of inaccurate global abort: Fig 8 shows, for 2PC and CO2PC, that the probability of inaccurate global abort grows with the degradation of the environment. In CTX5 and CTX6 such probability decreases when the probability of a participant to leave definitively the system grows. This result, somehow surprising, is explained by the fact that global aborts are more and more justified when participants leave the system rapidly.

Mean commit duration time: Fig. 9 shows mean commit duration time for the three protocols (blocking situations are discarded) in CTX4. The duration time grows until system collapses. Values fall drastically because most

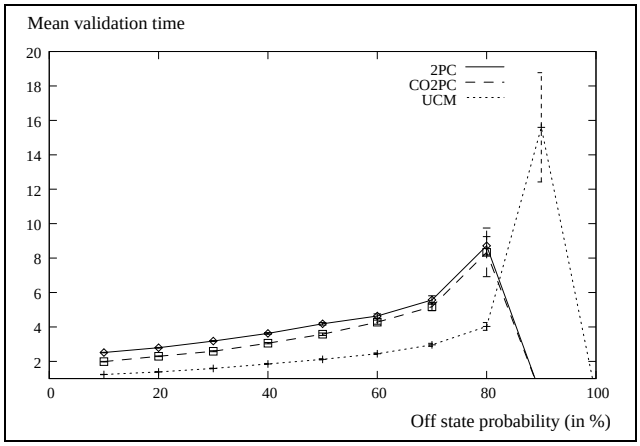

Fig. 9. Mean commit duration time vs probability of being disconnected (state Off).

transactions are blocked and discarded in the mean commit time evaluation. Once again these results show that protocols reducing the use of wireless network are less affected by context degradations.

VI. Conclusion and future work

We presented a performance analysis of the 2PC, UCM and CO2PC commit protocols used to validate transactions distributed over several mobile and fixed units. Protocols were simulated and tested under rather pessimistic conditions (e.g. high probability of disconnections). Results allow us to conclude that these three protocols behave well in mobile environments with poor connectivity. As expected, this study confirms that the less the protocol uses the wireless network the better it supports context degradations. A net degradation of performances is noticed when increasing disconnection frequency and duration. The order from the less affected to the most affected protocol is UCM, CO2PC and then 2PC. However the two last protocols allow to choose appropriate timeouts and this facility has not been used in our study. A longer timeout decreases the probability of inaccurate global abort but increases the average commit time.

This study has to be enhanced is several manners. 1) The whole transaction – not only the commit phase – have to be considered. This is important as some commit protocols impact the performances of the transaction execution (before its commitment). 2) Other commit protocols have to be simulated and added to the comparison. TCOT, for example is based on timeouts and allows participants to renegotiated their value during transaction executions. The extension of our analysis to the performance evaluation of the whole transaction allows a fair comparison with such kind of protocols. 3) More performance indices should be analyzed. For example the blocking probability for participants and its impact. Furthermore, qualitative aspects should be discussed. For example the properties the protocol insures, its impact on recovery and the requirements concerning underlying DBMS (autonomy, heterogeneity in concurrency control approach).

References

[1] C. Bobineau. *Gestion de transactions en environnement mobile*. PhD thesis, Université de Versailles, France, 2002.

[2] C. Bobineau, P. Pucheral, and M. Abdallah. A Unilateral Commit Protocol for Mobile and Disconnected Computing. In *12th Conf. on Parallel and Distributed Computing Systems (PDCS'00)*, Las Vegas, US, August 2000.

[3] S. Cuce, A. B. Zaslavsky, B. Hu, and J. Rambhia. Maintaining Consistency of Twin Transaction Model Using Mobility-Enabled Distributed File System Environment. . In *MDDS 02*, France, September 2002.

[4] J. Gray. Notes on database operating systems , operating systems, an advanced cours. *LNCS 60, Springer-Verlag*, 1978.

[5] V. Kumar, N. Prabhu, M. H. Dunham, and A. Y. Seydim. TCOT- A Timeout-Based Mobile Transaction Commitment Protocol. *IEEE Transactions on Computers*, 51(10), 2002.

[6] M. Lee and A. Helal. HiCoMo: High Commit Mobile Transactions. *Parallel and Distributed Database Systems*, 11(1), 2002.

[7] S. K. Madria and B. Bhargava. A Transaction Model for Improving Data Availability in Mobile Computing. *Kluwer Academic Publishers Distributed and Parallel Databases (DAPD)*, 10(2), 2001.

[8] E. Pitoura, P. K. Chrisanthis, and K. Ramamritham. Characterizing the Temporal and Semantic Coherency of Broadcast-Based Data Dissemination. In *Int. Conf. on Database Theory (ICDT)*, volume 2572 of *LNCS*, Siena, Italy, January 2003.

[9] E. Pitoura and G. Samaras. *Data Management for Mobile Computing*. Kluwer Academic Publishers, 1998.

[10] P. Serrano-Alvarado. *Transactions adaptables pour les environnements mobiles*. PhD thesis, Université J. Fourier, France, 2004.

[11] P. Serrano-Alvarado, C. L. Roncancio, and M. Adiba. A survey of mobile transactions. *Int'l Journal on Distributed and Parallel Databases (DAPD)*, à paraître 2004.

[12] P. Serrano-Alvarado, C. L. Roncancio, M. Adiba, and Labbé. Adaptable mobile transactions and environment awareness. In *BDA'03, 19èmes Journées Bases de Données Avancées*, Lyon, France, Octobre 2003.

[13] SimJava. http://www.dcs.ed.ac.uk/home/hase/simjava/.