Adaptable Mobile Transactions and Environment Awareness

Patricia Serrano-Alvarado [1] **— Claudia Roncancio — Michel Adiba — Cyril Labbé**

LSR-IMAG Laboratory
BP 72, 38402 St. Martin d'Hères, France
e-mail: Firstname.Lastname@imag.fr
http://www-lsr.imag.fr/Les.Groupes/STORM/

ABSTRACT. *Mobile environments are characterized by high variability (e.g. variable bandwidth, disconnections, different communication prices) as well as by limited mobile host resources. Such characteristics lead to high rates of transaction failures and unpredictable execution costs. This paper introduces an Adaptable Mobile Transaction model (AMT) that allows defining transactions with several execution alternatives associated to a particular context. The principal goal is to adapt transaction execution to context variations. An analytical study shows that using AMTs increases commit probabilities and that it is possible to choose the way transactions will be executed according to their costs. In addition, the middleware TransMobi is proposed. It manages environment awareness and implements the AMT model with suitable protocols.*

RÉSUMÉ. *Les environnements mobiles sont caractérisés par une grande variabilité (bande passante variable, déconnexions, prix de communication différents, etc.) ainsi que par des unités mobiles à ressources limitées. Ces caractéristiques entraînent un nombre important de défaillances transactionnels et des coûts d'exécution imprévus. Cet article introduit un modèle de transactions mobiles adaptables (AMT) permettant de définir des transactions avec plusieurs alternatives d'exécution. Le principal objectif est d'adapter l'exécution des transactions aux variations du contexte. Une étude analytique montre que les AMT augmentent la probabilité de validation et qu'il est possible de choisir le type d'exécution en fonction de son coût. Nous proposons également l'intergiciel TransMobi gérant la perception de l'environnement et implantant le modèle AMT à l'aide de protocoles appropriés.*

KEYWORDS: *Mobile transactions, ACID properties, environement awareness.*

MOTS-CLÉS : *Transactions mobiles, propriétés ACID, perception de l'environnement.*

1. Supported by the CONACyT scholarship program of the Mexican Government.

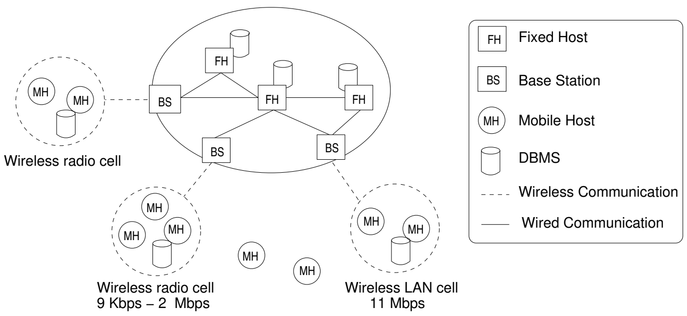

Figure 1. *Mobile environment global architecture.*

1. Introduction

The omnipresence of mobile devices such as cell phones, PDAs, smartcards, sensors and laptops, together with the development of different kinds of networks (local, wireless, ad-hoc, etc.) lead to a true mutation in the use, design and development of future information systems. Our work is related to the topics of ubiquitous and pervasive computing where technological improvements allow users to access data and perform transactions from any type of terminal and from anywhere using a wired or a wireless network. Applications that we have in mind cover a wide area. They might be personal ones, where clients want to access public data (e.g. weather forecast, stock exchange, road traffic) or professional ones, where mobility is inherent (e.g. mobile vendors/clients, health services, mobile offices, transport). We consider that mobile and fixed hosts can be clients or servers.

We consider a mobile computing environment with a network consisting of fixed and mobile hosts (FH, MH), see Figure 1. MHs could be of different nature ranging from PDAs to personal computers. Shared data are distributed over several database servers running, generally, on FHs. MH may run database management system (DBMS) modules and may provide some services to other hosts. While in motion, an MH may retain its network connection through a wireless interface supported by some FHs which act as Base Stations (BS). The geographical area covered by a BS is a cell. Each MH communicates with the BS covering its current cell. The process during which an MH enters into a new cell is called *hand-off*.

We make no specific assumptions about the database model (relational, object) but we place ourselves in a multidatabase environment assuming that data are managed by autonomous and possible heterogeneous DBMSs.

Applications in mobile environments are confronted to particular characteristics and limitations imposed by hardware such as: low and variable bandwidth, frequent disconnections, high communication prices, variable hardware configuration (due to plug-in components), limited display, battery autonomy, processing power and data storage. These limitations/variations lead to a lot of potential failure modes that affect data management process (e.g. queries, replication, caching, transactions, etc.).

In this paper, we are particularly interested on mobile transaction management. Focusing on this notion, we adopt a quite general definition: *a mobile transaction is a transaction where at least one mobile host takes part in its execution*. Mobile transactions are considered as long lived ones because of the probability of disconnections.

As an example, let us introduce an e-shopping application that allows people to browse products in an e-mall, to select, to book and to buy items. We assume also that secured e-payment is available based on credit cards or *e-money*. Application execution – as a set of transactions – will not be the same if they are launched from a (fixed) terminal office, from a PDA while traveling in a train or from home using a laptop. Thus, under this context:

– transactions may succeed but with different execution times (bandwidth capacity is highly variable) and communication prices (prices vary among wireless network providers/technologies/time-access);

– energy consumption is affected by low bandwidth (more battery is consumed);

– failures may occur due to unexpected disconnections or battery breakdown.

In traditional environments, application designers do not care about host and network characteristics. Nevertheless, in mobile applications we claim the necessity of being *environment aware* to overcome the infrastructure variability and to react to variations satisfying application and user requirements.

Several works concerning mobile transactions have been introduced (e.g. [DUN 97, YEO 94, DIR 00]). In the analysis made in [SER 03b], we found out that the majority of these proposals are particular solutions oriented to specific application context. Moreover, most of these works do not take into account the importance of mobile environment variability and therefore environment awareness.

The contribution of this paper is twofold, we propose an Adaptable Mobile Transaction (AMT) model and the middleware TransMobi to support it. The general idea of the AMT model is to define mobile transactions (T_{AMT}) with several *execution alternatives* associated to a particular mobile environment state. When a T_{AMT} transaction is launched, the appropriate execution alternative is initiated depending on the current mobile environment. We argue that adapting transaction executions improves commit rate, execution costs, response times and application availability, in short, the application's quality of service. We made an analytical study that shows how the AMT model increases transaction commit probability according to expected costs. The middleware TransMobi coordinates the execution of T_{AMT}s and manages the variations of the mobile environment state (e.g. bandwidth rate, communication price, MH disconnections, MH available energy, etc.). It uses suitable protocols to guarantee AMT properties. Among these protocols we propose (CO2PC) a Combination of an Optimistic approach and 2PC [GRA 78] to ensure *semantic atomicity*.

This paper is organized as follows: Section 2 presents the adaptable mobile transaction model and Section 3 gives an analytical study of its performances. Section 4 introduces the TransMobi middleware. Section 5 discusses related work and Section 6 concludes this article.

2. Adaptability for Mobile Transactions

2.1. *Overview*

Traditionally, transactions are defined independently of execution infrastructure. This approach is well suited for centralized and distributed systems where the execution characteristics have acceptable, predictable and controlled state variations. As the mobile environment is highly variable, transaction execution can be unpredictable.

We consider that in order to optimize resources, both transaction design and management should be made taking into account environment awareness. Our proposal is done along these lines. For each mobile transaction, application programmers give execution alternatives suitable to a particular environment context. For instance, a transaction distributed over a fixed and a mobile host will be launched if a good connection is available, whereas, a local execution will be preferable if there is no connection or if only a very low bandwidth is available.

To allow context aware transaction executions we propose an Adaptable Mobile Transaction (AMT) model. This model offers concepts to design mobile transactions (T_{AMT}). Generally speaking, a T_{AMT} is composed of at least one *execution alternative* involving one or several mobile or fixed hosts. Execution alternatives may be semantically equivalent. The successful execution of one of them, represents a correct execution of the T_{AMT}. A T_{AMT} also contains *environment descriptors* which express the state of the mobile environment required to execute each alternative. When a T_{AMT} is launched, the mobile environment state is checked and the appropriate execution alternative is chosen – only one alternative must be active by T_{AMT}. If the environment state does not allow the execution of any alternative, the execution of the T_{AMT} may be deferred. As soon as an acceptable environment state will be detected, an execution alternative will be triggered.

2.2. *Mobile Environment Awareness*

Mobile environments include the wireless network (WN), MHs involved in mobile transactions and MH locations. Several dimensions – connection state, bandwidth-rate or communication price – characterize the state of a mobile environment. As said before, the variability of such environment may affect transaction execution and impact resources consumption. Environment descriptors introduced here reflect the execution context and its potential state variations. Thus, transaction designers who know the application characteristics and requirements for quality of service, specify different execution alternatives and the required execution context for each of them.[1]

The set of relevant dimensions is specific to the application environment. It depends on the available mobile network, used MHs or user habitudes. For instance,

1. This might put the burden on the application designer. Future work should be oriented to develop computer aided environments for application developers.

	Dimension	States	Unit
WN	connection-state	*connected, disconnected*	
	bandwidth-rate	*high, medium, low*	kbytes/s
	communication-price	*free, cheap, expensive*	Euros/time
MH	available-battery	*full, half, low*	hh:mm:ss
	available-cache	*full, half, low*	kbytes
	available-persistent-memory	*full, half, low*	kbytes
	processing-capacity	*high, medium, low*	mhz/s
	estimated-connection-time	*t*	hh:mm:ss
	location		

Table 1. *Mobile environment characteristics.*

network bandwith is important under packet-switched networks (e.g. UMTS) and not under circuit-switched ones (e.g. GSM) where bandwidth is guaranteed if the connection is available. Also, communication price is not relevant under WLAN networks but may be crucial in other environments. In addition, user-defined dimensions can be introduced, for instance, specific "quality" of data.

The basic set of dimensions we consider is introduced in Table 1. To simplify, *States* are divided in levels of quality – *high*, *medium*, *low* – nevertheless, they can be defined according to each specific dimension.

Environment descriptors (ED) indicate dimensions and states as follows.

Definition 1 *An Environment Descriptor ED = {dimension=state(s)} contains a set of dimensions with their respective states at a given instant.*

For instance, to execute an alternative involving large data transfer, the required environment state may be:

Example 1 *ED = {connection-state = connected, bandwidth-rate = high, communication-price= free, cheap}.*

Notice that several hosts may be involved in a transaction. In that case, ED may or may not specify the required state for each involved host.

2.3. *The AMT Model*

This model allows to describe mobile transactions having one or more *execution alternatives* (EA_k), each of them is associated to an ED_k. EA_ks may take the form of any of the following execution types: the mobile transaction (1) is initiated by an MH and entirely executed on FHs, (2) is initiated by an MH/FH and entirely executed on an MH, (3) execution is distributed among MHs and FHs, and (4) execution is distributed among several MHs. So, a wide variety of mobile transactions is addressed.

An EA_k contains a set of *component transactions* (t_{ki}) which must respect ACID properties. They can be traditional *flat*, *distributed* or *close nested* transactions. *Compensating transactions* may be associated to component transactions. They will be executed in case of failures. An EA_k may be aborted if a component transaction aborts or if the mobile environment changes and the new state does not satisfy its environment descriptor. The EAs and the T_{AMT} are coordination units, data access is made only by component transactions. Making the analogy with multidatabases, component transactions are *local transactions* participating into *global transactions*.

Next, we present a semi-formal definition of the AMT model. A formal specification in ACTA [CHR 90] is presented in [SER 03a].

Definition 2 *An adaptable mobile transaction is a* $T_{AMT} = \langle EA_k \rangle$ *where:*

– $\langle EA_k \rangle$*,* $k > 0$*, is a list of execution alternatives for* T_{AMT} *where* EA_k *has higher priority than* EA_{k+1}.

– $EA_k = (ED_k, EP_k)$*, an execution alternative has an execution plan* EP_k *to be executed if the actual mobile environment satisfies the environment descriptor* ED_k.

– ED_k *describes the environment state for the suitable execution of* EP_k.

– $EP_k = \{(t_{ki}, ct_{ki}, HostId)\}$*, is a set of triplets introducing a component transaction, its compensating one and the host where they have to be executed.*

Let $\mathcal{RD}$ *be a relationship dependence over* EP_k*, such that:*
$\forall (t_{ki}, ct_{ki}, HostId_x), (t_{kl}, ct_{kl}, HostId_y) \in EP_k$;
$(t_{ki}, ct_{ki}, HostId_x) \mathcal{RD} (t_{kl}, ct_{kl}, HostId_y)$.

HostId indicates a database and the MH/FH responsible for the execution. Such host must execute only one component transaction per execution alternative. In $\mathcal{RD}$, we consider *parallel* or sequential execution dependencies.

Compensating transactions (ct_{ki}) are semantically equivalent to physical rollbacks and are defined to undo already committed component transactions. They recover semantically the database and avoid cascading aborts. Defining a ct_{ki} to compensate a t_{ki} is not always possible, thus, ct_{ki} will not always appear in EP_k.

AMT example

To continue with the example introduced in Section 1, consider an MH client with storage capacity, and an e-mall with two servers on the wired network: CatalogS and PurchaseS. The first site allows to query the store catalog whereas the second one takes purchase orders and payments.

We define the $T_{AMTshopping}$ with the following component transactions:

– GetCatalog allows clients to get a catalog;

– SelectItems allows clients to select items from a local copy of the catalog;

– Order-Pay allows clients to send the purchase order and payment to the store;

– AutoPay allows clients to pay on the MH (with e-money) without contacting other host;

– Order allows clients to send a purchase order (no payment included);

– Select-AutoPay = SelectItems + AutoPay

EA_k	ED_k	EP_k
k=1	{catalog-state= *uptodate*}	{(SelectItems, MH), (Order-Pay, PurchaseS)}
k=2	{connection-state=*connected*, bandwidth-rate = *high, medium*, communication-price=*cheap*, catalog-state=*present, missing*}	{(GetCatalog, CatalogS) (SelectItems, MH), (Order-Pay, PurchaseS)}
k=3	{connection-state=*connected*, bandwidth-rate=*low*, catalog-state=*missing*}	{(GetCatalog, CatalogS), (Select-AutoPay, MH), (Order, PurchaseS)}

Table 2. $T_{AMTshopping}$ *example.*

In Table 2, $T_{AMTshopping}$ proposes three execution alternatives to be triggered according to the environment state. In this example, wireless network dimensions that determine the choice of an alternative are: connection availability, bandwidth and communication price. The presence of the catalog on the MH is an application defined dimension. It takes the states *missing* (the catalog is not available on the MH) *present* (a version, probably out of date or incomplete, is on the MH) or *uptodate* (an up to date version is on the MH). Dimensions not appearing in environment descriptors are not considered as relevant for the context application.

In this example, the priorities between EA_ks are determined by the communication cost. Executing EA_1 is cheaper than executing EA_2. In EA_1 the MH has an up to date catalog, this allows saving communication messages. EA_1 may be launched even in disconnected mode and Order-Pay can be deferred until reconnection. EA_2 will be launched provided that the communication quality is acceptable. EA_3 is executed even under bad communication rates. The advantage of this alternative is that Select-AutoPay can be made in disconnected mode because the payment is in the MH. Order will be launched at reconnection.

Defining compensating transactions for this example is easy. They would mainly include operations to cancel orders and refundings.

2.4. *AMT Properties*

A T_{AMT} can be considered at three different levels: the T_{AMT} itself, the execution alternatives and the component transactions. For the last ones we assume that ACID properties are guaranteed, nevertheless, as we will see latter, durability is conditioned by the success of the corresponding execution alternative.

An execution alternative (actually the associated EP_k) is a kind of *sagas* [GAR 87] containing a set of transactions which execution may be distributed among mobile and fixed hosts. The $\mathcal{RD}$ defined inside the alternative describes the possibility of a parallel or sequential execution of component transactions. Integrity constraints can be defined and verified at t_{ki} level, under the responsibility of the underlying DBMS. Global data integrity constraints are not considered but value dependencies between t_{ki}s (of the same EA_k) are allowed.

Property	t_{ki}	**EA**$_k$	T_{AMT}
Atomicity	$\surd$	Semantic atomicity	Semi-atomicity
Consistency	$\surd$	Semantic consistency	
Isolation	$\surd$	Relaxed (local commits)	
Durability	$\surd$ conditioned	Underlying DBMS	
Correctness	Serializability	Global serializability	

Table 3. *Summary of AMT properties.*

Atomicity and isolation for EA_k

Considering the restrictions of mobile environments, the AMT model relaxes atomicity by adopting semantic atomicity [GAR 83] (as in sagas).

Definition 3 *EA_k preserves semantic atomicity if either:*

– *All t_{ki}s defined in EA_k commit.*

– *All t_{ki}s defined in EA_k are compensated or rolled back.*

The goal is to avoid blocking participant hosts and to allow MH disconnections. This is obtained with *local commits* where *partial* results are shared before the EA_k commits. The durability of locally committed transactions is conditioned to the commit of the EA_k. In case of abortion, compensating transactions are used.

To address critical applications with non-compensatable transactions, resources are blocked. Thus, when transactions terminate, resources are retained until a global decision (EA_k commits/aborts) is made. Hence, compensating transactions are not needed. If no participant commit locally, compensating transactions will be unnecessary and traditional atomicity is obtained.

Since dependency values between t_{ki}s of the same EA_k are allowed, a correct global ordering must be ensured. That is because concurrent execution of several alternatives might introduce indirect interference between value dependent transactions. The criterion used to control the correctness of concurrent execution alternatives is *global serializability*.[2] Global serializability states that transactions of each EA_k must have the same relative serialization order in their corresponding underlying DBMS.

Even though a global serializable order is preserved, semantic consistency [GAR 83] is provided – due to semantic atomicity.

Atomicity and Isolation for T_{AMT}

We mentioned in Section 2.3 that EA_ks defined in a T_{AMT} may be semantically equivalent. A T_{AMT} is correctly executed if one of its EA_ks is successfully executed. This results in *Semi-atomicity* [ZHA 94] which is guaranteed for T_{AMT}s as follows.

Definition 4 *AMT preserves semi-atomicity if either:*

2. In this paper, serializability is actually *conflict-serializability*.

– All t_{ki} defined in EA_k commit and all attempted t_{li}s, of another alternative EA_l in the same T_{AMT}, are aborted or have their effects compensated.

– No partial effects of component transactions of the T_{AMT} remain permanent in the underlying DBMSs.

Serializability of T_{AMT}s is offered trough the serializability of execution alternatives. Regarding the durability property, once the corresponding EA_k commits (and consequently the T_{AMT}), durability of component transactions is provided by the underlying DBMS. Table 3 summarizes properties at all levels (t_{ki}, EA_k, T_{AMT}). Section 4 introduces appropriate protocols to ensure these properties.

3. Impact of Environment Awareness: Analytical Study

This section provides an analytical study of the capabilities allowed by the AMT model. In Section 3.1, we use a probabilistic model to analyze several execution alternatives one by one – separated of the AMT model. For each of them, we evaluate its initiation probability and its execution cost. Section 3.2 highlights the benefits expected from environment awareness and AMT adaptable facilities. It is shown that the T_{AMT} initiation probability is always greater than the initiation probability of one alternative. In addition, it is shown how the AMT model allows the designer to define the best T_{AMT} according to the required quality of service: low cost without complete guarantee of success or at the opposite success any time at any cost. The $T_{AMTshopping}$ example (introduced in Section 2.3) is used all along this section to illustrate the analytical study. A short concluding discussion is proposed in Section 3.3.

3.1. *Analytical Study of an Execution Alternative*

Mobile Environment Model

As mentioned in previous sections, a mobile environment state can be seen as a set of dimensions. An EA_k is initiated only if the environment state satisfies the acceptable states for each dimension.

Example 2 *The environment descriptors used by $T_{AMTshopping}$ include the following dimensions and states:*

		$j = 1$ Good	$j = 2$ Medium	$j = 3$ Bad
$i = 1$	connection-state	*connected*		*disconnected*
$i = 2$	bandwidth-rate	*high*	*medium*	*low*
$i = 3$	communication-price		*cheap*	*expensive*
$i = 4$	catalog-state	*uptodate*	*present*	*missing*

Definition 5 *Let p_{ij} be the probability of dimension i to be in state j.*

In the resulting matrix $P = (p_{ij})$, $\forall i$, $\sum_j p_{ij} = 1$. Next example shows that this matrix depends on the considered mobile environment.

Example 3 *An example of the matrix probability of an environment like the one introduced in the example 2 could be :*

$$P = (p_{ij}) = \begin{bmatrix} 0.8 & 0 & 0.2 \\ 0.7 & 0.2 & 0.1 \\ 0 & 0.4 & 0.6 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

Here, the probability of being connected is given by p_{11}, the probability of having an uptodate catalog is given by p_{41} and so on. p_{31} has 0 probability because in the considered environment, communication is never free.

Environment Description Matrix

Definition 6 *For each EA_k=(ED_k, EP_k) we denote by Δ^k the boolean matrix where:*

$$\delta_{ij}^k = \begin{cases} 1 & if\ the\ state\ j\ of\ dimension\ i\ is\ acceptable\ for\ EA_k \\ 0 & otherwise \end{cases}$$

Example 4 *The Δ^k matrix for the three proposed alternatives of $T_{AMTshopping}$ (see Table 2) are:*

$$\Delta^1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \Delta^2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \Delta^3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

If a dimension i does not appear in ED_k, any state is suitable: if $i \notin ED_k$ then $\forall j, \delta_{ij}^k = 1$. Δ^1 and Δ^2 illustrate this case.

Cost Matrix

Definition 7 *Let C^k be the matrix where c_{ij}^k is the cost of the EP_k execution in state j of dimension i.*

c_{ij}^k must be defined by the designer according to application needs in cost improvement. Example 5 shows some particular definitions of cost.

Example 5 *For the $T_{AMTshopping}$ one could identify the memory utilization as a cost associated to the dimension* catalog-state *or the CPU consumption as a cost associated to the dimension* connection-state *(since in disconnected mode more operations are done on the MH). Let us focus on communication price and battery utilization respectively associated to dimension* communication-price *and* bandwidth-rate *(bandwidth limitations increase battery consumption). The considered wireless network is UMTS which uses a packet-switched communication where bandwidth rate goes from 144 kbps (vehicular mobility), 384 kbps (pedestrian mobility) to 2 mbps for indoor traffic. These bandwidth rates correspond to* high, medium *and* low *states. Communication price depends on the size of transmitted data (packets). For instance, the execution of EP_1 requires three wireless logical messages. First, a transaction request (kind of login), then the purchase order (component transaction* Order-Pay*) and finally, a message is received by the MH to confirm the order (acknowledgment). The execution of EP_2 and EP_3 requires an extra message to ask for the catalog which is received through another message (*GetCatalog*).*

Three different sizes of messages may be identified: small messages (login, ack and ask for catalogue), medium ones (for Order-Pay *and* Order*) and large ones (for* GetCatalog*). Let us assume that small, medium and large messages are composed respectively of* 1, 10 *and* 20 *packets. So, if the execution plan of EA_k sends n_s small, n_m medium and n_l large messages then n_p is the number of packets sent or received*[3] *by the MH during the EP_k execution, $n_p = n_s + 10n_m + 20n_l$.*

To evaluate communication cost (communication price and battery consumption), we assume that:

– Sending a single packet in the state cheap *(resp.* expensive*) costs 1 unit of price (resp. 2 units).*

– If bandwidth is high *(resp.* medium*,* low*) sending/receiving a single packet uses* 0.1% *(resp.* 0.2%, 0.4%*) of the battery capacity. In this case, the cost associated to the dimension* bandwidth-rate *is the battery consumption.*

Under these assumptions we have:

$$C^k = \begin{bmatrix} 0 & 0 & 0 \\ 0.1n_p & 0.2n_p & 0.4n_p \\ 0 & n_p & 2n_p \\ 0 & 0 & 0 \end{bmatrix}$$

where c^k_{2j} is the battery consumption of the EP_k execution when the bandwidth-rate *is in state j. c^k_{3j} is the price of the EP_k execution when the* communication-cost *is in state j. In $T_{AMTshopping}$, $n_p = 12$ for EP_1 and $n_p = 33$ for EP_2 and EP_3, so:*

$$C^1 = \begin{bmatrix} 0 & 0 & 0 \\ 1.2 & 2.4 & 4.8 \\ 0 & 12 & 24 \\ 0 & 0 & 0 \end{bmatrix} \quad C^2 = C^3 = \begin{bmatrix} 0 & 0 & 0 \\ 3.3 & 6.6 & 13.2 \\ 0 & 33 & 66 \\ 0 & 0 & 0 \end{bmatrix}$$

Mean Cost of an EA_k

The execution plan of EA_k is launched by the system when the mobile environment is in the state j of the dimension i with the probability:

$$\frac{\delta^k_{ij} p_{ij}}{\sum_j \delta^k_{ij} p_{ij}}$$

So, the mean cost due to dimension i of the EP_k execution is given by:

$$c^k_i = \frac{\sum_j \delta^k_{ij} c^k_{ij} p_{ij}}{\sum_j \delta^k_{ij} p_{ij}}$$

Example 6 *With previous P, Δ^k and C^k we can compute:*

$$c^3_2 = \frac{13.2 * 0.1}{0.1} = 13.2\%$$

$$c^3_3 = \frac{33 * 0.4 + 66 * 0.6}{0.4 + 0.6} = 52.8$$

This means that the average of battery consumption of EA_3 is 13.2% with an average price of $c^3_3 = 52.8$ units, whereas $c^1_2 = 1.8\%$, $c^1_3 = 19.2$ units and $c^2_2 = 4.03\%$, $c^2_3 = 33$ units.

3. Eventually, it could be interesting to distinguish between sending or receiving messages.

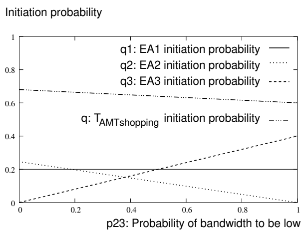

Figure 2. *Initiation probability vs probability of bandwidth to be* low.

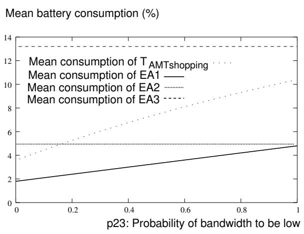

Figure 3. *Battery consumption vs probability of bandwidth to be* low.

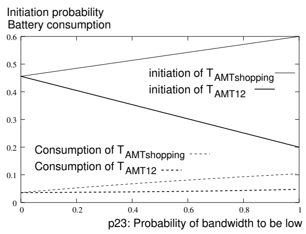

Figure 4. *Initiation probability and battery consumption vs bandwidth.*

EA_k Initiation Probability

Definition 8 *Let q^k be the probability for EA_k of being selected for execution. q^k will be called the EA_k initiation probability. Since the probability that dimension i has an acceptable state for EA_k is given by $\sum_j \delta^k_{ij} p_{ij}$, we have:*

$$q^k = \prod_i \left(\sum_j \delta^k_{ij} p_{ij}\right)$$

So the execution plan of an EA_k has a chance to be initiated ($q^k > 0$) iff $\forall i, \exists j$ such that $\delta^k_{ij} = 1$.

Comparing EAs

Example 7 *In order to study EAs in different types of environment, performances indices are studied in regard to the probability of the bandwidth to be* low *(p_{23}):*

$$P = \begin{bmatrix} 0.8 & 0 & 0.2 \\ (1-p_{23})/2 & (1-p_{23})/2 & p_{23} \\ 0 & 0.4 & 0.6 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

Fig. 2 shows the EA_k initiation probability and Fig. 3 the battery consumption associated to each EA_k in $T_{AMTshopping}$. It can be seen that if the bandwidth is often *low* (e.g. $p_{23} = 0.8$) then EA_3 is the alternative with the highest battery consumption whereas it has the best initiation probability. q^1 is constant because it does not depends on the bandwidth to be *low* but only on the probability for the catalog to be *uptodate*.

3.2. *Analytical Study of a* T_{AMT}

As mentioned in Section 2, mobile environment awareness allows the system to choose an EA_k if the environment state corresponds to the associated ED_k. EA_k has higher priority than EA_{k+1}; this allows us to assume without lost of generality that:

Property 1 *If a state of the environment is suitable for an EA_k it should not be suitable for an $EA_{k'}$ of the same T_{AMT}. That is: $\forall (k,k'), k \neq k', \exists i$ such that $\forall j, \delta^{k'}_{ij} \neq \delta^k_{ij}$*

T_{AMT} Initiation Probability

Definition 9 *Let q_{AMT} be the T_{AMT} initiation probability. Thanks to property 1, we have: $q_{AMT} = \sum_k q^k$*

Mean Cost of a T_{AMT}

Definition 10 *Let c_i be the mean cost (associated to dimension i) of the execution of T_{AMT}. Under the assumption that the environment is stable during the execution, EA_k is initiated with the probability q^k so the cost of the whole T_{AMT} is c_i^k with the probability q^k, hence:*

$$c_i = \frac{\sum_k c_i^k q^k}{\sum_k q^k}$$

As an example, we study the mean battery consumption and the initiation probability of $T_{AMTshopping}$ in regards to the probability of the bandwidth to be *low* (p_{23}).

Fig. 2 shows that the $T_{AMTshopping}$ initiation probability never reaches 1. This is due to the fact that for certain states no alternative is defined. For instance, if the MH is *disconnected* with a *missing* catalog $T_{AMTshopping}$ can not be initiated. This figure also shows that the initiation probability of a single EA_k is smaller than the initiation probability of the whole $T_{AMTshopping}$.

Fig. 3 shows that the mean battery consumption of the $T_{AMTshopping}$ increases with the probability of the bandwidth to be *low* is going up. This is due to the fact that EA_3 initiation probability is bigger than the one of EA_2 (see Fig. 2).

Fig. 4 shows performance parameters for $T_{AMTshopping}$ and a variant without EA_3. This new transaction is called T_{AMT12}. It can be seen that $T_{AMTshopping}$ has a better initiation probability whereas T_{AMT12} has a better mean battery consumption.

3.3. *Discussion*

This section showed that compared to non-adaptable approaches, adaptability in transaction execution improves performances and allows choosing the way the transaction will be executed according to execution costs. Without environment awareness, a transaction is defined in a standard way (the execution plan is fixed). The system will try to execute this transaction whatever the state of the environment is. If the current state does not allow the execution, the transaction will fail even if another execution alternative could have been successful. In the same way, the environment state may lead to a costly execution without considering cheaper alternatives. Allowing the system to choose the execution plan in regards to the current environment state is a way to ensure better performances.

With n different execution plans, environment awareness and AMT facilities allow to choose among $n!$ different T_{AMT} (each one including n alternatives) depending upon the user optimization criteria, e.g. minimal costs or execution time. The definition of the T_{AMT} that provides best initiation probability does not depends on the number of defined EA_ks but on the capacity to overcome environment variations.

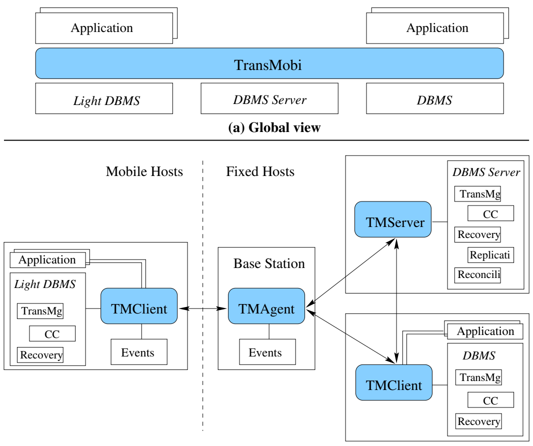

Figure 5. *TransMobi global architecture.*

To ensure a better quality of service with T_{AMT} execution, specific tools based on this analytical model could be used to define optimized T_{AMT}s.

Environment awareness allows to define the T_{AMT} that fits the best to the quality of service required by the application, for instance, a trade-off between reducing costs and relaxing quality of service can be done.

4. TransMobi Middleware

We designed the TransMobi[4] middleware which supports adaptable mobile transactions (T_{AMT}). This section presents the architecture of TransMobi, its functionalities, the environment awareness and the protocols to ensure the AMT properties.

4.1. *Overview and Architecture*

As a middleware between application code and existing DBMSs, TransMobi coordinates the execution of T_{AMT}s, see Fig. 5a. We assume the existence of DBMS functionalities on fixed and mobile hosts.[5] TransMobi relies on them for the execution – ensuring ACID properties – of component and compensating transactions. To preserve DBMS autonomy, TransMobi uses standard interfaces.

TransMobi has the client/agent/server architecture showed in Fig 5b. A TMClient acts as an intermediate between the mobile application and underlying DBMS. It may run on FHs, called *fixed TMClients* or on mobile hosts, called *mobile TMClients*. Among others a TMClient manages MH awareness.

4. Preliminary ideas on a Mobile Transaction Service (MTS) were introduced in [SER 02].
5. Products like FastObjects j2 [Fas] or Oracle9i Lite [Ora] provide ACID properties.

	Event type	Attached information	Notification mode
WN	*e-connection*	MH id	Immediate push
	e-disconnection	MH id	
	e-hand-off	New BS id	
	e-bandwidth-rate-changes	Bandwidth rate	
	e-communication-price-changes	Communication price	
MH	*e-available-battery-changes*	Available battery	Pull before choosing an EA_k
	e-available-cache-changes	Available cache	
	e-available-persistent-memory-changes	Available persistent memory	
	e-available-processing-capacity	Available processing capacity	
	e-location-changes	Location	

Table 4. *Event types for environment awareness.*

TransMobi Agents (TMAgent) are on BSs[6] or on some FHs able to communicate with MHs. TMAgents play a particularly important role, they (1) facilitate the interaction between mobile and fixed hosts acting as a representative (e.g. storing communication messages addressed to disconnected MHs, keeping track of particular MH information), (2) store transactional coordination information (e.g. transaction state, commit process), (3) follow MH moves (in case of hand-off, the *old* TMAgent sends the related MH information to the *new* TMAgent), and in particular, (4) manage network awareness.

A TMServer does some kind of global control, for instance, it maintains a global serializability graph. Furthermore, it interacts with database servers requesting for transaction execution on behalf of mobile applications. Besides transaction execution, servers may provide replication and reconciliation features, but it is not our objective to address these issues here.

Generally speaking, T_{AMT} application requests are intercepted by a TMClient. It chooses the best EA_k to be initialized according to the mobile environment state and distributes the t_{ki}s to the appropriate hosts. Distribution is made according to relationship dependences over EP_k (see Section 2.3). Mobile TMClients send the EA_k to their TMAgent which takes in charge the distribution. Value dependent component transactions are executed sequentially, independent ones can be executed concurrently.

4.2. *Adaptability and Environment Awareness*

TransMobi takes advantage of the adaptability potential of the AMT model. TMClients take in charge the comparison of the actual mobile environment state with environment descriptors. Comparison is made in the order defined by the EA_k list.

The environment state concerns the wireless network, MHs and location. A mobile TMClient knows its state (see Table 1) and perceives the WN one. For *fixed TMClients* the state of the mobile environment is provided by TMAgents which perceive the WN

6. Wireless network providers, generally, do not allow to install software on BSs, we believe that this will change in the future.

state. An MH may publish its state completely or partially. It may include physical characteristics but also particular processing functions and available data.

Environment awareness can be implemented with events. Table 4 defines the event types that reflect the variations of the considered mobile environment. Related information can be attached to events. Notifications are made in *pull* and *push* manners. This approach facilitates immediate and deferred triggering of T_{AMT}s.

TransMobi uses an *event service* (Events component in Fig. 5b). Such a service produces environment events from information supplied by *environment monitors* of different types and levels (e.g. operating system or communication layer). Some events are easy to generate, for instance, the *e-bandwidth-rate-changes* can be estimated by using knowledge of the active wireless interface (determined from active wireless card) or by obtaining the link latency and connection throughput. The *e-connection* event can be obtained by using the environment monitors *mhmicp* or *pumicp* proposed in modules of the Columbia Mobile IP [IOA 91]. There are events like *e-communication-price-changes* which may be obtained from a kind of *environment profile* where information related to contracts with wireless network providers are stocked.

4.3. *TransMobi Protocols*

This section presents the protocols to provide the properties of the AMT model (see Section 2.4), particularly, semantic atomicity and global serializability. Thanks to the assumptions made by the AMT model – each T_{AMT} has only one EA_k active – the commit/abort of EA_k ensures semi-atomicity.

4.3.1. *Semantic Atomicity*

CO2PC Protocol

Our objective is to provide semantic atomicity for execution alternatives by allowing participants to perform either optimistic local commit (locally committed results are shared) or non-optimistic commit. We attempt to increase the flexibility of participants (particularly MHs), thus compensable transactions can be committed locally in an optimistic manner, whereas non-compensable ones have to wait for the global EA_k decision.

We propose CO2PC, a Combination of an Optimistic approach and 2PC [GRA 78]. CO2PC works as follows: all participants send a *vote (commit/abort)* to the coordinator which takes the global decision and sends it to all participants. An unanimous vote for commit leads to a *commit* decision, otherwise the decision is *abort*.

Participants are TMClients or TMServers working with a local DBMS. The CO2PC coordinator must be a participant FH. If there is no FH participant then a TMAgent will play this role. MH should not be coordinator because they are subject to frequent disconnections and have restrained resources for logging and communication.

A participant making optimistic commit (called *opt-participant*) executes its t_{ki} and commits/aborts it unilaterally. Then, it sends its *vote* to the coordinator. If the

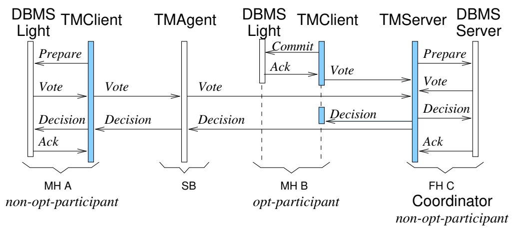

Figure 6. *CO2PC example.*

global decision is *commit*, *opt-participants* are done; if it is *abort* they have to launch compensating transactions. Fig. 6 shows CO2PC with one *opt-participant* (MH B).

A participant making non-optimistic commit (called *non-opt-participant*) executes the 2PC protocol locally.[7] The TMClient (or TMServer) acts as coordinator for this 2PC and the unique participant is the underlying DBMS. In Fig. 6, MH A and FH C are *non-opt-participants*. The TMClient/TMServer initiate the 2PC with their respective DBMSs (*prepare* message). Each DBMS sends its *vote* to the TMClient/TMServer which forwards it to the CO2PC coordinator. If the *vote* is *abort*, the participant aborts the t_{ki} unilaterally, otherwise t_{ki} enters into a *prepared* state. When the decision of the CO2PC coordinator arrives to participants, the corresponding *commit/abort* is executed on the underlying DBMS. Notice that the 2PC part of the protocol is made on the same host and does not require messages through the wireless network. Fig. 6 shows that only *vote* and *decision* messages are transmitted over the wireless network (2 messages per participant).

In order to tolerate MH disconnections, to limit undefined blocking and to break eventual deadlocks, CO2PC uses timers. A timeout is assigned to each t_{ki} and to EA_ks. Commits must be done before timeouts expire. Timeouts could be renegotiated for instance, when an MH disconnection period is known in advance. When a participant *votes commit*, it deactivates its timer and has to wait for the global decision. If the EA_k timeout expires before the coordinator has all *votes*, the EA_k is aborted or compensated (consequently the T_{AMT}).

2PC is used by CO2PC because – for non-compensable transactions – resources must be retained until a global commit/abort. 2PC provides this state, besides it ensures atomicity, produces *recoverable* systems, *avoids cascading aborts* (when it is combined with two-phase locking) and its interface is widely implemented.

Notice that 2PC is not used for the global coordination of T_{AMT} atomicity. 2PC is not the best choice for mobile environments because it requires a high rate of messages (four per participant) and it blocks the entire global transaction for an undefined period of time, until a global *decision* is made. This is particularly constraining in the mobile context. Moreover, light MHs may not support the 2PC interface or may not have enough resources to spend on it. Thus, trying to address several kind of transactions

7. A non-compensable transaction cannot be executed on a DBMS not supporting 2PC.

Hosts	Log information
Opt-participant	*Commit, Acknowledge*
Non-opt-participant	*Start2PC, Prepare*
All participants	*Vote, Decision*
Coordinator	*StartCO2PC, Votes, Decision*

Table 5. *Logging during CO2PC.*

(non-compensable as well as compensable ones) and different types of MHs (with limited and unlimited resources) we propose CO2PC.

Recovery

To provide recovery, CO2PC records its progressing steps in the coordinator and participant logs. Since failures and disconnections can occur at any moment, logging information should be forced to be written (i.e. flushed into a stable storage) before sending messages. For MHs, CO2PC information will be logged in the TMClient log as well as in the corresponding TMAgent. Physical storage of MH is not considered stable because MHs are subject to lost, damages and undefined disconnections.

Table 5 shows information to be logged by the participants and the coordinator. In order to preserve DBMS heterogeneity, we make no assumptions about the recovery policy used by underlying DBMS. To preserve autonomy, recovery information (e.g. logs) is not required.

Once compensating transactions are initiated they should complete successfully. This characteristic is called *persistence of compensation* [GAR 87] and is ensured by resubmitting compensating transactions until they commit. If persistence of compensation is guaranteed there is no need to use an atomic commit protocol to obtain the atomicity of the set of ct_{ki}s of an EA_k. ct_{ki}s of sequentially executed t_{ki}s are executed in the reverse commit order. Other ct_{ki}s can be executed concurrently.

4.3.2. *Global Serializability*

Concurrency control algorithms synchronize concurrent transactions by ordering their conflicting operations such that a serialization order can be maintained. At a global level (EA_k), determining conflicts is not easy, particularly when DBMSs do not share local management information. Two global transactions without direct conflicts may however conflict if there exists local transactions that do not belong to EA_ks - such conflicts may affect the consistency of value dependent component transactions. These *indirect conflicts* are known at global level only if DBMS autonomy is relaxed. In our context, DBMS autonomy is respected. To provide global serializability two conditions should hold. (1) local DBMS should produce serializable and recoverable schedules. (2) each local DBMS should maintain the subtransactions relative order determined at global level.

Thanks to the assumptions that we made (underlying DBMSs preserve ACID properties), the first condition holds. To guarantee the second condition, we propose to use the Optimistic Ticket Method (OTM) [GEO 93]. Several reasons motivate our choice:

– OTM maintains global serializability.

– Autonomy of underlying DBMS is preserved. No information about local ordering is required.

– No extra wireless messages are necessary.

– Concerning traditional approaches, using homogeneous concurrency control protocols does not guarantee global serializability. Exception is using *strong 2PL* [BER 87], however this approach does not allow early unlocking (i.e. optimistic commit) and is therefore not suitable for mobile environments.

– Sequential execution of global transactions preserves global serializability only when it is combined with *timestamp ordering* [BER 87]. This approach is not suitable because it reduces concurrency, and applied to mobile environments may generate undefined waiting periods of time for the entire system.

The main idea in OTM is that indirect conflicts are converted into direct ones. For this, each underlying DBMS has a special data item, called *ticket*. Each global subtransaction reads the ticket at the host where it executes (there is one ticket by host), increments it, and writes it into the database. At global level, a *global serialization graph* (GSG) is maintained.

TransMobi uses OTM at EA_k level and combines it with CO2PC. The GSG is maintained by a global *serializability manager* (SM), located on a TMServer. When the commit process of an EA_k begins, the SM creates an EA_k node in the GSG. Each t_{ki} increments the ticket and when the execution terminates the participant sends the *vote* and the ticket value to the CO2PC coordinator. The latter sends the ticket value to the SM which inserts edges between EA_k and recently committed EA_js. Edges must reflect the relative ticket order [GEO 93]. As soon as a cycle is generated, the SM advises the coordinator which aborts the EA_k. If no cycle is generated, after all edges have been inserted, the coordinator is able to commit EA_k. Communication cost to maintain the GSG is minimal: 1 message to create the EA_k node (from the CO2PC coordinator), 1 message by participant to transmit the ticket (from the CO2PC coordinator) and 1 message to validate/abort EA_k (to the CO2PC coordinator). Maintaining GSG does not require extra wireless messages. Information needed by the SM is included in messages used in the CO2PC process.

4.4. *Disconnection Management*

A key property of mobile computing is to support disconnections as a normal state of the system. For this, TransMobi gives support to disconnections at a transactional level. That is, during the transaction execution, participant MHs may disconnect. There are two kinds of disconnection, the programmed one and the unexpected one. The former is requested by the user and the latter is a consequence of some mobile environment variations e.g. MH is out of the area covered by the wireless network or MH battery is exhausted.

TransMobi protocols tolerate both kind of disconnections. In CO2PC, after the *vote* is sent, an MH may disconnect temporally. In that case, the coordinator *de-*

cision is logged in a TMAgent and sent to the MH when reconnection occurs. In the same way, the timeout assigned to each component transaction allows the MH to disconnect provided that the *vote* is sent before the timeout expires. Both, *opt* and *non-opt-participants* may disconnect. Nevertheless resources of *non-opt-participants* will remain blocked until reconnection (because their 2PC phase is not finished).

Concerning the OTM protocol, disconnections are not a problem because the serialization order is dictated by the ticket updating. This is completely independent of the MH connection state. The unique requirement is to send the ticket value to the coordinator. Nevertheless, the disconnection duration has an important consequence. The GSG is constructed in the commit order of the execution alternatives (actually the T_{AMT}s). Thus, if there is a conflict, the alternative trying to be serialized will be aborted. That is, the abort/success probability is related to the time that a transaction takes to be serialized.

5. Related Work

The AMT model was inspired from DOM [BUC 92] and Flex [ELM 90] where the general idea is to define *equivalent* transactions for being executed in case of failures. The DOM transaction model allows *close* and *open nested* transactions. Compensating transactions as well as contingency ones can be specified for being executed if a given transaction fails. In the Flex transaction model, contingency transactions are defined in terms of *functionally equivalent* transactions. A failure order is defined where the execution of a transaction depends on the failure of another one. Unlike AMT, DOM and Flex transactions are not defined to deal with mobile environments and the notion of context awareness is not considered.

The panorama of mobile transactions is vast. A detailed analysis of several works is given in [SER 03b]. In general, the adaptability vision is very limited, almost all studied systems adapt their behavior to support disconnections. Only Moflex [KU 00] addresses execution adaptability when hand-off occur. On the contrary, our proposition – AMT model – can take into account any defined environment characteristics. As in KT [DUN 97], Pre-serialization [DIR 00] and Moflex [KU 00], our work addresses mobile transactions as multidatabase ones. Unlike AMT, none of these propositions consider execution on MHs.

Concerning atomicity protocols for mobile environments we identify UCM [BOB 00] and TCOT [KUM 02]. UCM defines an Unilateral Commit Protocol that supports off-line executions (on MHs) and disconnection. Several assumptions are made, for instance, all participants are constrained to use strict 2PL and at commit time the effects of all local transactions are logged on stable storage. UCM guarantees atomicity and durability, nevertheless, transactions are blocked until global commit and log of updates must be flushed (on a FH) at each transaction commit. In TCOT, participants may commit locally before the global commit using timeouts. If global commit fails, compensating transactions are executed. To be able to commit independently, a mobile participant must send its log of updates to the coordinator. This protocol does not take disconnections into account. Our proposal – CO2PC – makes no assumptions about concurrency control and recovery techniques used by DBMS participants. Thus, their autonomy is preserved and heterogeneity is allowed. Further, CO2PC reduces the use of the wireless network (2 messages by participant are necessary) and allows cohabitation of compensable and non-compensable transactions.

6. Conclusion and Perspectives

This paper addressed mobile transactions and made several contributions: (1) We proposed the AMT model inspired by previous extended transaction models. However, we put a special emphasis on environment awareness by considering specific dimensions, e.g. bandwidth rate, connection state, mobile host resources, etc. The model concerns transactions involving several heterogeneous DBMS running on mobile or fixed hosts. (2) We provided an analytical study that can be viewed as a semi automatic tool to optimize mobile transaction design and executions. This is done by an a priori study of several execution alternatives and their respective probabilities of success. (3) We defined a TransMobi middleware which implements the AMT model. It uses a client/agent/server architecture and among others it manages environment awareness based on events. (4) Inside TransMobi we proposed the CO2PC commit protocol to guarantee semantic atomicity of T_{AMT}. It preserves the autonomy of participant DBMSs, reduces the use of the wireless network (2 messages by participant are necessary) and allows cohabitation of compensable and non-compensable transactions. (5) We developed a prototype of TransMobi. It uses Personal Java and the PointBase DBMS (Java package). Mobile hosts are of type Ipaq H3850 (Compaq) over a WLAN 802.11b. This is a first prototype to validate our ideas. Experiences in a more general environment for specific mobile applications are part of our future works.

Research perspectives include:

(1) The implementation of an application developer utility in order to facilitate the design of AMT transactions. Environment awareness provided by TransMobi can be used to profile the application environment and users habits. Once the environment profiled, an analytical calculus – like the one presented here – could be done to dynamically optimize T_{AMT} in regards to required quality of service. (2) The analysis of the suitability of adapting transactions not only at the beginning (as done in our currents proposal) but also during their execution; aspects related to reusing work already done by the adapted transaction have to be explored. (3) Applying the AMT model to peer-to-peer and network ad-hoc environments; as the mobile environment they are characterized by high variability.

Acknowledgments

We wish to thank the members of the NODS project for their feedbacks all along this research, particularly to S. Drapeau, E. Benitez-Guerrero and N. Jayaprakash for their valuable remarks on earlier versions of this article.

7. References

[BER 87] BERNSTEIN P. A., HADZILACOS V., GOODMAN N., *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[BOB 00] BOBINEAU C., PUCHERAL P., ABDALLAH M., "A Unilateral Commit Protocol for Mobile and Disconnected Computing", *PDCS Conf.*, USA, 2000.

[BUC 92] BUCHMANN A., OZSU M. T., HORNICK M., GEORGAKOPOULOS D., MANOLA F. A., "*Database Transaction Models for Advanced Applications*", chapter 5, Morgan Kaufmann, 1992.

[CHR 90] CHRYSANTHIS P. K., RAMAMRITHAM K., "ACTA: A framework for Specifying and Reasoning about Transaction Structure and Behavior", *ACM SIGMOD Conf.*, 1990.

[DIR 00] DIRCKZE R. A., GRUENWALD L., "A Pre-Serialization Transaction Management Technique for Mobile Multidatabases", *MONET*, vol. 5, num. 4, 2000.

[DUN 97] DUNHAM M. H., HELAL A., BALAKRISHNAN S., "A Mobile Transaction Model that Captures Both the Data and the Movement Behavior", *MONET*, vol. 2, num. 2, 1997.

[ELM 90] ELMAGARMID A. K., LEU Y., RUSINKIEWICS M., "A Multidatabase Transaction Model for INTERBASE", *VLDB Conf.*, Australia, 1990.

[Fas] FASTOBJECTS J2, "http://www.fastobjects.com/".

[GAR 83] GARCIA-MOLINA H., "Using Semantic Knowledge for Transaction Processing in a Distributed Database", *TODS*, vol. 8, num. 2, 1983.

[GAR 87] GARCIA-MOLINA H., SALEM K., "Sagas", *SIGMOD Conf.*, USA, 1987.

[GEO 93] GEORGAKOPOULOS D., RUSINKIEWICZ M., SHETH A., "Using Tickets to Enforce the Serializability of Multidatabase Transactions", *TKDE*, vol. 6, num. 1, 1993.

[GRA 78] GRAY J., "Notes on Database Operating Systems", *Advanced Course: Operating Systems*, num. 60 LNCS, 1978.

[IOA 91] IOANNIDIS J., DUCHAMP D., MAGUIRE G. Q., "IP-Based Protocols for Mobile Internetworking", *SIGCOMM Conf.*, Switzerland, 1991.

[KU 00] KU K., KIM Y., "Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems", *RIDE Conf*, USA, 2000.

[KUM 02] KUMAR V., PRABHU N., DUNHAM M. H., SEYDIM A. Y., "TCOT- A Timeout-Based Mobile Transaction Commitment Protocol", *IEEE TC*, vol. 51, num. 10, 2002.

[Ora] ORACLE9I LITE, "http://otn.oracle.com/products/lite/".

[SER 02] SERRANO-ALVARADO P., "Defining an Adaptable Mobile Transaction Service", *EDBT Young Researchers WS*, num. 2490 LNCS, March 2002.

[SER 03a] SERRANO-ALVARADO P., "Transactions Mobiles Adaptables", PhD thesis, Université Joseph Fourier, Grenoble, France, 2003, In preparation.

[SER 03b] SERRANO-ALVARADO P., RONCANCIO C., ADIBA M., "A Survey of Mobile Transactions", *Distributed and Parallel Databases (DAPD)*, , 2003, To appear.

[YEO 94] YEO L. H., ZASLAVSKY A., "Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment", *ICDCS Conf.*, Poland, 1994.

[ZHA 94] ZHANG A., NODINE M., BHARGAVA B., BUKHRES O., "Ensuring Relaxed Atomicity for Flexible Transactions in Multidatabase Systems", *SIGMOD Conf.*, USA, 1994.