An Adaptable Mobile Transaction Model
Patricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, Cyril Labbé

To cite this version:

HAL Id: hal-00415333
https://hal.archives-ouvertes.fr/hal-00415333
Submitted on 10 Sep 2009
An Adaptable Mobile Transaction Model

Patricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, Cyril Labbé

e-mail: Firstname.Lastname@imag.fr

LSR-IMAG Laboratory BP 72, 38402 St. Martin d’Hères, France

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 variable execution costs. To raise the success rate of transactions and to have a minimal control on resources consumption we claim that both application design and transaction management should be environment aware. This paper proposes an Adaptable Mobile Transaction model (AMT) that allows defining transactions with several execution alternatives associated to a particular context. When an AMT is launched, the appropriate execution alternative is initiated depending on the current environment state. The goal is to adapt transaction execution to context variations. Our model relaxes atomicity and isolation properties but preserves conflict-serializability. A specification of the AMT model in ACTA (formalism based on the first order logic) is presented. 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.

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.

Figure 1: Mobile environment global architecture.

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/variation 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 mo-
bile 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-mail, 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. Nevertheless, in mobile applications we claim the necessity of being environment aware.

Moreover, most of these works do not take into account infrastructure variability and to react to variations satisfying application and user requirements.

Several works concerning mobile transactions have been introduced (e.g. [10, 27, 9, 19, 27, 20, 26, 15, 18, 16, 6]). In the analysis made in [24], we found out that the majority of these proposals are particular solutions oriented to specific application contexts. 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 an Adaptable Mobile Transaction (AMT) model. The general idea 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 propose a formal specification of the AMT model using ACTA [7, 8]. The goal is to specify the properties and behaviour of the AMT model with a well defined and accepted formalism. Finally, we make an analytical study that shows how the AMT model increases transaction commit probability according to expected costs.

This paper is organized as follows: Section 2 presents the adaptable mobile transaction model and Section 3 its formal specification. Section 4 gives an analytical study of its performances. 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 [23, 25]. 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 mobile network, on the nature of MHs or on user behavior. For instance, network bandwidth 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.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>States</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandwidth-rate</td>
<td>full, half, low</td>
<td>kbytes/s</td>
</tr>
<tr>
<td>communication-price</td>
<td>full, half, low</td>
<td>Euros/time</td>
</tr>
<tr>
<td>processing-capacity</td>
<td>high, medium, low</td>
<td>mhz/s</td>
</tr>
<tr>
<td>available-persistent-memory</td>
<td>full, half, low</td>
<td>hh:mm:ss</td>
</tr>
<tr>
<td>connection-state</td>
<td>connected, disconnected</td>
<td></td>
</tr>
<tr>
<td>estimated-connection-time</td>
<td></td>
<td>hh:mm:ss</td>
</tr>
</tbody>
</table>

Table 1: Mobile environment characteristics.

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\(k\)s 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 (ct\(k_i\)) 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 EA\(s\) and the TAMT 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 [7, 8] is presented in Section 3.

Definition 2 An adaptable mobile transaction is a TAMT = \{EA\(k\)\} where:

- \(EA\(_k\)\), \(k > 0\), is a list of execution alternatives for TAMT 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\(_k\) = \{(ct\(_{k_i}\), HostId)\}, is a set of triplets introducing a component transaction, its compensating one and the host where they have to be executed. Let RD be a relationship dependence over EP\(_k\), such that: \(\forall (ct\(_{k_i}\), HostId)\), \((ct\(_{k_i}\), HostId) \in EP\(_k\); (ct\(_{k_i}\), HostId)RD(tki,ctki,HostId).\)

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

Compensating transactions (ct\(_{k_i}\)) 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\(_{k_i}\) to compensate a t\(_{k_i}\) is not always possible, thus, ct\(_{k_i}\) will not always appear in EP\(_k\).

\(^1\)This might put the burden on the application designer. Future work should be oriented to develop computer aided environments for application developers.
AMT example

To continue with the example introduced in Section 1, consider an MH client with storage capacity, and an e-mail 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**

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 $EA_k$s 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 ($\text{bandwidth-rate} = \text{high, medium and communication-price} = \text{cheap}$). $EA_3$ is executed even under bad communication rates ($\text{bandwidth-rate} = \text{low}$). 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 refund and cancel orders.

<table>
<thead>
<tr>
<th>$t_{ki}$</th>
<th>$EA_k$</th>
<th>$t_{ki}$</th>
<th>$EA_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{k1}$</td>
<td>GetCatalog, CatalogS, MH, (Select-Order, Order-Pay, PurchaseS)</td>
<td>$t_{k1}$</td>
<td>GetCatalog, CatalogS, (Select-Order, MH)</td>
</tr>
<tr>
<td>$t_{k2}$</td>
<td>GetCatalog, CatalogS, MH, (Select-Order, Order-Pay, PurchaseS)</td>
<td>$t_{k2}$</td>
<td>GetCatalog, CatalogS, (Select-AutoPay, MH)</td>
</tr>
<tr>
<td>$t_{k3}$</td>
<td>GetCatalog, CatalogS, MH, (Select-Order, Order-Pay, PurchaseS)</td>
<td>$t_{k3}$</td>
<td>(SelectCatalog, CatalogS)</td>
</tr>
</tbody>
</table>

Table 2: $T_{AMTshopping}$ example.

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* [13] containing a set of transactions which execution may be distributed among mobile and fixed hosts. The $RD$ 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 $t_{kis}$ (of the same $EA_k$) are allowed.

Atomicity and isolation for $EA_k$

Considering the restrictions of mobile environments, the AMT model relaxes atomicity by adopting semantic atomicity [12] (as in sagas).

**Definition 3** Semantic atomicity of an $EA_k$

Each $EA_k$ ensures semantic atomicity if either:

1. all $t_{ki}$s defined in $EA_k$ commit if $EA_k$ commit
2. all $t_{ki}$s defined in $EA_k$ are compensated or rolled back if $EA_k$ aborts.

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 $t_{kis}$ of the same $EA_k$ are allowed, a correct global ordering must be...
Table 3: Summary of AMT properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Ttk</th>
<th>EAk</th>
<th>TAMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomicity</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Correctness</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Consistency</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Durability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Serializability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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. Global serializability states that transactions of each EA must have the same relative serialization order in their corresponding underlying DBMS.

Even though a global serializable order is preserved, semantic consistency is provided due to atomicity.

Atomicity and Isolation for TAMT

We mentioned in Section 2.3 that EA defined in a TAMT may be semantically equivalent. A TAMT is correctly executed if one of its EA is successfully executed. This results in semi-atomicity which is guaranteed for TAMT as follows.

Definition 4 Semi-atomicity of a TAMT

Each TAMT guarantees semi-atomicity if either:

1. the commit of a TAMT implies the commit of only one EA and the abortion or compensation of all component transactions of other EA,

2. the abortion of a TAMT implies the abortion or compensation of all other component transactions of the EA in progress.

Serializability of TAMT is offered through the serializability of execution alternatives.

Definition 5 Global serializability of EA

A set of EA ensures global serializability:

1. if the execution order of component transactions ensures a serializable order in each site and

2. if EA ensures also a serializable order on all sites.

Regarding the durability property, once the corresponding EA commits (and consequently the TAMT), durability of component transactions is provided by the underlying DBMS. Table 3 summarizes properties at all levels (tki, EA, TAMT).

In [25, 23], we define a middleware (named TransMobi) that implements the AMT model with appropriate protocols. Due to space constraints it is not presented here. TransMobi uses a client/agent/server architecture. It manages environment awareness based on events that are generated thanks to sensors that supervise MH and wireless communication capacities. Roughly speaking, applications request AMT executions to TransMobi which verifies the mobile environment state and decides the way transactions will be executed (it chooses the appropriate execution alternative).

As a middleware between application code and existing DBMSs, TransMobi coordinates the execution of TAMT. We assume the existence of DBMS functionalities on fixed and mobile hosts. TransMobi relies on them for the execution – ensuring ACID properties – of component and compensating transactions. EA and TAMT properties are ensured as follows. Concerning EA, semantic atomicity is guaranteed by the CO2PC protocol that we propose. Semantic consistency is a consequence of the execution of compensating transactions. As the AMT model is an open nested transaction, isolation at EA and AMT levels is relaxed. To guarantee global serializability, we propose to use the Optimistic Ticket Method that is used in multidatabase systems. Finally, as each TAMT has only one EA active, the commit/abort of EA ensures the semi-atomicity of TAMT.

3 AMT Formal Specification

This section proposes a specification of the AMT model in ACTA. ACTA is a formalism based on first order logic that allows defining and comparing principal characteristics of extended transaction models. With ACTA, it is possible to specify the effects of extended transactions on each other and on objects. We use ACTA because it is a well-accepted formalism and its capabilities of expression and extension are enough to specify the AMT properties.

This section is organized as follows. Firstly, an axiomatic definition of the AMT model (Section 3.1) is proposed. Secondly, in order to obtain the properties of the TAMT transactions, we analyze existing axioms (Section 3.2).

3.1 AMT axiomatic definition

Dependencies introduced in ACTA are not sufficient to specify the AMT model. Nevertheless, due to the extensibility capability of the formalism, we propose the following dependency:

Unique Begin Dependency (tj USD ti): ti can begin, if any other tj has not begun:

\( (\text{begin}_{ti} \in H) \Rightarrow \neg(\text{begin}_{tj} \in H) \)

3[3, 2] introduce a comparison of CO2PC with other validation protocols for mobile environments.
Notation
Next, we show the elements and sets used in the AMT axiomatic definition.

- \( T_{AMT} \) defines an *Adaptable Mobile Transaction* that contains a list of Execution Alternatives \( EA_k \).
- \( T_{AMT} = \langle EA_1, \ldots, EA_n \rangle, n > 0 \)
- An execution alternative \( EA_k \) is composed of a set of component transactions \( t_{ki} \).
- \( EA_k = \{ t_{k1}, \ldots, t_{kn} \}, n > 0 \)
- \( EA_k \neq EA_l \)

Here, we make abstraction of environment descriptor \( ED_k \) and execution site \( SiteId \) described in Definition 2.

- \( EA_k \) organize its \( t_{ki} \) in two sets, \( CT_k \) et \( NT_k \), where \( CT_k \) contains compensables \( t_{ki} \) and \( NT_k \) contains non-compensables \( t_{ki} \).
- \( CT_k \cap NT_k = \phi \)
- Compensables and non-compensables transaction sets must be disjoints.
- \( ST_k \) denotes the \( t_{ki} \) list that must be executed sequentially:
  \( (ST_k \subseteq CT_k) \lor ((t_{ki}, \ldots, t_{kn-1} \subseteq CT_k) \land (t_{kn} \in NT_k)) \)
  where \( 1 \leq i \leq n, t_{ki} \in ST_k \)
- Only compensable transactions can be executed sequentially, except for the last one.
- \( ct_{ki} \) denotes a compensating transaction for \( t_{ki} \): \( \forall t_{ki} \in CT_k \exists ct_{ki} \).
- All compensable transactions must have a compensating transaction.
- \( t \) denotes \( t_{ki} \) or \( ct_{ki} \).
- \( S_u \) denotes the transaction set that is executed on site \( u \). \( EA_k \) can execute only one component transaction by site.

Annex A.1 presents the types of dependencies used in the next axiomatic definition of the AMT model.

**Definition 6 Axiomatic definition of the AMT model**

1. \( SE_{T_{AMT}} = SE_{EA} = SE_t = \{ \text{begin}, \text{commit}, \text{abort} \} \)
2. \( IE_{T_{AMT}}, IE_{EA}, IE_t = \{ \text{begin} \} \)
3. \( TE_{T_{AMT}}, TE_{EA}, TE_t = \{ \text{commit}, \text{abort} \} \)
4. \( t \) satisfies the fundamental axioms I to IV (c.f. Annex A.2)
5. \( View_{T_{AMT}} = \phi \)
6. \( View_{EA} = \phi \)
7. \( View_t = H(S_u) \)
8. \( ConflictSet_{T_{AMT}} = \phi \)
9. \( ConflictSet_{EA} = \phi \)
10. \( ConflictSet_t = \{ p'_t[ob] | t' \neq t, t', t \in S_u, \text{Inprogress}(p'_t[ob]) \} \)
11. \( \forall ob \exists p (p_t[ob] \in H) \Rightarrow (ob \text{ is correct and serializable}) \)
12. \( \text{(commit}_t \in H) \Rightarrow \neg(t \text{ commits locally}) \)
13. \( \exists ob \exists p \left( \text{commit}_t[p_t[ob]] \in H \right) \Rightarrow (\text{commit}_t \in H) \)
14. \( \left( \text{commit}_t \in H \right) \Rightarrow \forall ob \forall p \left( (p_t[ob] \in H) \Rightarrow \left( \text{commit}_t[p_t[ob]] \in H \right) \right) \)
15. \( \exists ob \exists p \left( \text{commit}_t[p_t[ob]] \in H \right) \Rightarrow (\text{abort}_t \in H) \)
16. \( \left( \text{abort}_t \in H \right) \Rightarrow \forall ob \forall p \left( (p_t[ob] \in H) \Rightarrow \left( \text{abort}_t[p_t[ob]] \in H \right) \right) \)
17. \( (\text{commit}_t \in H) \Rightarrow \neg(\text{ED}_t^* EA) \)
18. \( \text{post(begin}_{T_{AMT}}) \Rightarrow \left( (\text{begin}_{EA_k} \in H) \Rightarrow \text{ConditionEnvironment} \right) \land \left( (EA_k, UB_D EA_k) \in \text{DepSet}_{ct} \right) \land \left( t_{ki} \subseteq BD EA_k \right) \)
19. \( \text{post(begin}_{EA_k} \Rightarrow \left( (T_{T_{AMT}} AD EA_k) \in \text{DepSet}_{ct} \right) \land \left( (EA_k, AD T_{AMT}) \in \text{DepSet}_{ca} \right) \land \left( t_{ki} \in TS_k \right) \Rightarrow \left( (t_{ki}, BC_D t_{ki-1}) \in \text{DepSet}_{ct} \right) \)
20. \( \text{post(begin}_{t_k} \Rightarrow \left( (EA_k, AD t_{ki}) \in \text{DepSet}_{ct} \right) \land \left( t_{ki} \in CT_k \right) \Rightarrow \left( (t_{ki}, WD EA_k) \in \text{DepSet}_{ct} \right) \land \left( (ct_{ki}, BC_D t_{ki}) \in \text{DepSet}_{ct} \right) \land \left( (t_{ki} \subseteq NT_k) \Rightarrow \left( (t_{ki}, AD EA_k) \in \text{DepSet}_{ct} \right) \right) \)

where \( 1 \leq k \leq m \),
If \( EA_k \) aborts \( T_{AMT} \) must abort and if \( T_{AMT} \) aborts \( EA_k \) must abort. Component transactions of \( ST_k \) are executed sequentially, other \( t_{ki} \) can be executed concurrently.
If \( t_{ki} \) aborts \( EA_k \) aborts. For compensable transactions, if \( EA_k \) aborts \( t_{ki} \) must abort (if \( t_{ki} \) has not committed) and \( ct_{ki} \) can begin only if \( t_{ki} \) has committed. For non-compensable transactions (\( NT_{ki} \)), if \( EA_k \) aborts \( t_{ki} \) must abort. \( t_{ki} \) commit is delayed until \( EA_k \) commit (\( t_{ki} \) wait for an \( EA_k \) abort until \( EA_k \) commits).

21. \( \text{post}(\text{commit}_{t_{ki}}) \Rightarrow (((ct_{ki}, \text{BADEA}_k) \in \text{DepSet}_{ct}) \land ((ct_{ki}, \text{CMD} \text{EA}_k) \in \text{DepSet}_{ct})) \)

where 1 \( \leq i \leq n \), 1 \( \leq k \leq m \)
\( ct_{ki} \) can begin if \( EA_k \) aborts and if \( EA_k \) aborts after \( t_{ki} \) commits, \( ct_{ki} \) must commit.

3.2 Deductions and analysis of the axioms

This section deduces the properties of \( T_{AMT} \) transactions from axioms of Definition 6.

Axioms 1-3 specify significant events of the AMT at three levels (\( T_{AMT}, EA \) and \( t \)). Axiom 4 states that component and compensating transactions (t) must respect fundamental axioms (c.f. Annex A.2). Axioms 5-6 show that \( T_{AMT} \) and \( EA_k \) are control points which do not access data. Thus, Axioms 8-9 specify that there are not possible conflicts for \( T_{AMT} \) and \( EA_k \). Axiom 7 states that the view of \( t \) is limited to the projection of history \( H \) on \( S_u \) (set of transactions executing on site \( u \)). Therefore, in Axiom 10 the set of conflicts of \( t \) is composed of all operations executed by other transactions in \( S_u \).

In Axiom 18, dependency ((begin\(_{EA_k} \in H \Rightarrow \text{ConditionEnvironment}) \), introduces a beginning condition for \( EA_k \). The goal is to launch the alternative when \( \text{ConditionEnvironment} \) is satisfied. \( \text{ConditionEnvironment} \) becomes true when the environment descriptor (ED) of \( EA_k \) (c.f. Section 2.3) matches with the current mobile environment state.

Next section (3.2.1), shows properties of component and compensating transactions. Sections 3.2.2 and 3.2.3 deduce semantic atomicity and global serializability of \( EA_k \) respectively. Finally, Section 3.2.4 shows the semi-atomicity property of \( T_{AMT} \).

3.2.1 \( t_{ki} \) and \( ct_{ki} \) properties

This section shows that component and compensating transactions are atomic (Lemma 2) so with AID properties.

**Lemma 1** If \( t_i \in T_{AMT}, t_i \) is failure atomic

Proof of Lemma 1:

\( t_i \) is failure atomic if it satisfies both conditions of Definition 2 (c.f. Annex A.3).

1. Condition 1 (all clause) is derived from Axioms 13 and 14.

2. Condition 2 (nothing clause) is derived from Axioms 15 and 16.

**Lemma 2** If \( t_i \in T_{AMT}, t_i \) is an atomic transaction

Proof of Lemma 2:

1. Condition 1 (failure atomic) is derived from Lemma 1.

2. Condition 2 (serializable) is derived from Axioms 11 and 12.

Since atomic transactions satisfy AID properties (c.f. Theorem 1), component and compensating transactions are AID transactions.

3.2.2 Semantic atomicity of \( EA_k \)

This section shows that execution alternatives have the semantic atomicity property.\(^4\) We begin by introducing the commitment (Lemma 3) and abortion (Lemma 4) of \( EA_k \). Next, we obtain the semantic atomicity of \( EA_k \) (Lemma 5).

**Lemma 3** Commitment of an \( EA_k \)

Let \( H \) be the history of an execution alternative \( EA_k \) with \( n \) component transactions.

\((\text{commit}_{EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n (\text{commit}_{t_{ki}} \in H))\)

The commitment of an execution alternative \( EA_k \) implies the commitment of all associated component transactions \( t_{ki} \).

Proof of Lemma 3:

If \( EA_k \) commits, its set of component transactions must commit due to the abort dependency of \( EA_k \) on \( t_{ki} \) of the Axiom 20 (the first one) and the fundamental Axiom III:

\( \forall i, 1 \leq i \leq n ((\text{abort}_{t_{ki}} \in H) \Rightarrow (\text{abort}_{EA_k} \in H)) \Rightarrow ((\text{commit}_{t_{ki}} \in H) \Rightarrow (\text{commit}_{EA_k} \in H)) \)

**Lemma 4** Abortion of an \( EA_k \)

Let \( H \) be a history of an \( EA_k \) with \( n \) component transactions.

\((\text{abort}_{EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq n, i \neq j ((\text{abort}_{t_{ki}} \in H) \land (\text{commit}_{t_{kj}} \rightarrow \text{commit}_{t_{kji}}))\)

If \( EA_k \) aborts, associated component transactions must be aborted or compensated.

\(^4\)Semantic atomicity expression is similar to the one introduced in [7].
Proof of Lemma 4:

Case 1. If $EA_k$ aborts when $t_{ki}$ is in progress, $t_{ki}$ aborts due to the $WD$ or $AD$ dependencies of $t_{ki}$ on $EA_k$ (Axiom 20). Due to fundamental Axiom II, it is not necessary to specify that only transactions that have begun are aborted. Similarly, due to fundamental Axiom III, it is not necessary to specify that only non-committed transactions are aborted. Due to the $BCD$ dependency of $ct_{ki}$ on $t_{ki}$ (Axiom 20) $ct_{ki}$ does not begin in this case:

$(abort_{EA_k}) \Rightarrow \forall i, 1 \leq i \leq m (commit_{ct_{ki}})$

Only $t_{ki} \in CT_k$ can commit due to the $WD$ dependency. By its side, $t_{ki} \in NT_k$ commit until $EA_k$ commit due to the abortion dependence of $t_{ki}$ on $EA_k$. Thus, all $t_{ki} \in NT_k$ in progress are aborted if $EA_k$ aborts.

Case 2.

1. If $EA_k$ aborts after $t_{ki}$ commits and before a $t_{kj}$ begins, $ct_{ki}$ must commit due to the $CMD$ dependency of Axiom 21:

$(abort_{EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq m (commit_{ct_{ki}})$

2. Due to the existence of $BCD$ dependency of $ct_{ki}$ on $t_{ki}$ on axiom 20, if $ct_{ki}$ begins, $t_{ki}$ has been committed:

$(begin_{ct_{ki}} \in H) \Rightarrow (commit_{ct_{ki}} \rightarrow begin_{ct_{ki}})$

By fundamental Axiom II:

$(commit_{ct_{ki}} \in H) \Rightarrow (begin_{ct_{ki}} \rightarrow commit_{ct_{ki}})$

Thus, by the semantics of the dependency relation the commit of $ct_{ki}$ is done after the commit of $t_{ki}$:

$(commit_{ct_{ki}} \in H) \Rightarrow (commit_{ct_{ki}} \rightarrow commit_{ct_{ki}})$

3. From 1 and 2:

$(abort_{EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq m (commit_{ct_{ki}})$

Simplifying:

$(commit_{ct_{ki}} \in H) \Rightarrow (commit_{ct_{ki}} \rightarrow commit_{ct_{ki}})$

Case 3. If an $EA_k$ aborts when a $t_{ki}$ is in progress (Case 1) and after a $t_{kj}$ has committed (Case 2):

$(abort_{EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq n, i \neq j$

$(abort_{t_{ki}} \in H) \land (commit_{t_{kj}} \rightarrow commit_{ct_{ki}})$

Lemma 5 Each $EA_k$ ensures semantic atomicity

Each $EA_k$ ensures semantic atomicity if conditions of definition 3 are ensured. 5

Proof of Lemma 5:

1. Condition 1 of Definition 3 (if $EA_k$ commits, all $t_{ki}$ transactions must commit) is ensured by Lemma 3.

2. Condition 2 of Definition 3 (if $EA_k$ aborts, all $t_{ki}$ transactions must abort or compensate) is ensured by Lemma 4.

3.2.3 Global serializability of $EA_k$

This section specifies that execution alternatives are globally serializable. From some axioms and Definition 6, it is possible to deduce the global serializability of $EA_k$ (Lemma 6).

Lemma 6 $EA_k$ ensures global serializability

Proof of Lemma 6:

To provide global serializability both conditions of Definition 5 must be ensured:

1. Condition 1 is derived from Axiom 12.

2. Condition 2 is derived from Axiom 17.

3.2.4 Semi-atomicity of $T_{AMT}$

This section states that $T_{AMT}$ have the semi-atomicity property. We begin by introducing the commitment of $T_{AMT}$ (Lemmas 7 and 8) as well as its abortion (Lemmas 9 and 10). Next, we obtain the semi-atomicity of $T_{AMT}$ (Lemma 11).

To obtain those properties, we define firstly the commitment (complete) of a $T_{AMT}$ (Lemma 8). For this, we specify the commitment (simple) of a $T_{AMT}$ (Lemma 7) and the commitment of an $EA_k$ (Lemma 3). Similarly, we define the abortion of a $T_{AMT}$ (Lemmas 10, 9 and 4).

Lemma 7 Commitment of a $T_{AMT}$

Let $H$ be the history of a $T_{AMT}$ and $EA_k$ an execution alternative associated to a $T_{AMT}$.

$((commit_{T_{AMT}} \in H) \Rightarrow \exists k, 1 \leq k \leq m (commit_{EA_k} \in H))$

This Lemma expresses that if the history contains the commit of a $T_{AMT}$ it contains also the commit of an associated execution alternative $k$.

Proof of Lemma 7:

1. If $T_{AMT}$ commits, $EA_k$ must also commit due to the abort dependency of $T_{AMT}$ on $EA_k$ specified in Axiom 19 (the first one) and the fundamental Axiom III, which says that a transaction must commit or abort:

$\forall k, 1 \leq k \leq m ((abort_{EA_k} \in H) \Rightarrow (abort_{T_{AMT}} \in H)) \Leftrightarrow ((commit_{T_{AMT}} \in H) \Rightarrow (commit_{EA_k} \in H))$
2. Only one $EA_k$ commits due to the $UBD$ dependency of the Axiom 18 where only one $EA$ must begin:

$$\forall k, 1 \leq k \leq m, \forall l, 1 \leq l \leq m, l \neq k
(b_{begin, EA_k} \in H) \Rightarrow \neg(b_{begin, EA_l} \in H)$$

3. From 1 and 2

$$((comm_{commit, EA_k} \in H) \Rightarrow \exists k, 1 \leq k \leq m \ (comm_{commit, EA_k} \in H))$$

**Lemma 8** Complete commitment of a $T_{AMT}$

Let $H$ be the history of a $T_{AMT}$ with $n$ component transactions and $EA_k$ an execution alternative associated to $T_{AMT}$.

$$((comm_{commit, EA_k} \in H) \Rightarrow \exists k, 1 \leq k \leq m \ (comm_{commit, EA_k} \in H)) \land
((comm_{commit, EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n \ (comm_{commit, t_i} \in H))$$

Simplifying:

$$(comm_{commit, EA_k} \in H) \Rightarrow \exists k, 1 \leq k \leq m \forall i, 1 \leq i \leq n, \ (comm_{commit, t_i} \in H)$$

Proof of Lemma 8:

This Lemma follows from Lemmas 7 et 3.

**Lemma 9** Abortion of a $T_{AMT}$

Let $H$ be the history of a $T_{AMT}$ and $EA_k$ an execution alternative associated to $T_{AMT}$.

$$((com_{abort, EA_k} \in H) \Rightarrow \exists k, 1 \leq k \leq m \ (com_{abort, EA_k} \in H))$$

This Lemma expresses the history in which the abortion of a $T_{AMT}$ implies the abortion of the $EA_k$ in progress.

Proof of Lemma 9:

1. If $T_{AMT}$ aborts, $EA_k$ must also abort due to the abort dependency of $EA_k$ on $T_{AMT}$ of Axiom 19:

$$((com_{abort, EA_k} \in H) \Rightarrow (com_{abort, EA_k} \in H))$$

2. Only one $EA_k$ aborts due to the $UBD$ dependency of the Axiom 18 where only one $EA_k$ must begin, see 2 in Lemma 7

**Lemma 10** Complete abortion of a $T_{AMT}$

Let $H$ be the history of a $T_{AMT}$ with $n$ component transactions and $EA_k$ an execution alternative associated to $T_{AMT}$.

$$((com_{abort, EA_k} \in H) \Rightarrow \exists k, 1 \leq k \leq m \ (com_{abort, EA_k} \in H)) \land
(com_{abort, EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq n, i \neq j
((com_{commit, t_i} \in H) \land (com_{commit, t_j} \rightarrow com_{commit, t_k}))$$

Simplifying:

$$((com_{abort, EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq n, i \neq j
((com_{commit, t_i} \in H) \land (com_{commit, t_j} \rightarrow com_{commit, t_k}))$$

This Lemma expresses the history in which the abortion of a $T_{AMT}$ implies the abortion of the $EA_k$ in progress and the compensation or abortion of component transactions associated to $EA_k$.

Proof of Lemma 10:

This Lemma follows from Lemmas 9 and 4.

Thanks to the analysis made, we can state that:

**Theorem 1** The execution of a $T_{AMT}$ produces one of the following histories:

1. $((comm_{commit, EA_k} \in H) \Rightarrow \exists k, 1 \leq k \leq m \ (comm_{commit, EA_k} \in H)) \land
((comm_{commit, EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n \ (comm_{commit, t_i} \in H))$

2. $((com_{abort, EA_k} \in H) \Rightarrow \exists k, 1 \leq k \leq m \ (com_{abort, EA_k} \in H)) \land
com_{abort, EA_k} \in H) \Rightarrow \forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq n, i \neq j
((com_{commit, t_i} \in H) \land (com_{commit, t_j} \rightarrow com_{commit, t_k}))$

Proof of Theorem 1:

This Theorem follows from Lemmas 8 and 10.

**Lemma 11** Each $T_{AMT}$ guarantees the semi-atomicity

Each $T_{AMT}$ guarantees semi-atomicity if both conditions of Definition 4 are satisfied.

Proof of Lemma 11:

1. Condition 1 of Definition 4 (all transactions in $EA_k$ commit) is satisfied by Lemma 8.

Abortion or/and compensation of all component transactions of other $EA_i$ is not necessary due to $UBD$ dependence of Axiom 18.

2. Condition 2 of Definition 4 (all transactions in $T_{AMT}$ are aborted or compensated) is satisfied by Lemma 10.

**Theorem 2** Each $T_{AMT}$ has the following properties:

1. $t_i$ has the AID properties,
2. $EA_k$ ensures semantic atomicity,
3. $EA_k$ ensures global serializability,
4. $T_{AMT}$ ensures semi-atomicity.

Proof of Theorem 2

(1) follows from Lemma 2, (2) follows from Lemma 5, (3) follows from Lemma 6 and (4) follows from Lemma 11.

**Definition 7** The management schema of a $T_{AMT}$ is correct if it follows the Definition 6.
4 Impact of Environment Awareness: Analytical Study

This section provides an analytical study of the capabilities allowed by the AMT model. In Section 4.1, 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 4.2 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_AMT_shopping 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 4.3.

4.1 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_AMT_shopping include the following dimensions and states:

<table>
<thead>
<tr>
<th>Dimension</th>
<th>State</th>
<th>Acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>Low</td>
<td>Acceptable</td>
</tr>
<tr>
<td>Connection</td>
<td>Uptodate</td>
<td>Acceptable</td>
</tr>
<tr>
<td>Battery</td>
<td>Medium</td>
<td>Acceptable</td>
</tr>
<tr>
<td>Communication</td>
<td>Cheap</td>
<td>Acceptable</td>
</tr>
</tbody>
</table>

Definition 8 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:

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

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 9 For each EA_k = (ED_k, EP_k) we denote by $\Delta_k^i$ the boolean matrix where:

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

Example 4 The $\Delta^k$ matrix for the three proposed alternatives of T_AMT_shopping (see Table 2) are:

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

If a dimension i does not appear in ED_k, any state is suitable: if $i \notin ED_k$ then $\forall j, \delta_{ij}^k = 1$. $\Delta^1$ and $\Delta^3$ illustrate this case.

Cost Matrix

Definition 10 Let $c^k_{ij}$ be the cost of the EP_k execution in state j of dimension i.

$c^k_{ij}$ 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_AMT_shopping 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\(^6\) by the MH during the \(E_P^k\) 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:

\[
c^k = \begin{bmatrix}
0 & 0.1p & 0.2p & 0.4p \\
0 & 0.2p & 0.4p & 0
\end{bmatrix}
\]

where \(c^k_{ij}\) is the battery consumption of the \(E_P^k\) execution when the bandwidth-rate is in state \(j\). \(c^k_{ij}\) is the price of the \(E_P^k\) execution when the communication-cost is in state \(j\). In \(T_{AMT\text{-shopping}}\), \(n_p = 12\) for \(E_P^1\) and \(n_p = 33\) for \(E_P^2\) and \(E_P^3\), so:

\[
c^1 = \begin{bmatrix}
0 & 0 & 0 \\
1.2 & 2.4 & 4.8 \\
0 & 0 & 0
\end{bmatrix},
\quad
c^2 = c^3 = \begin{bmatrix}
0 & 0 & 0 \\
3.3 & 6.6 & 13.2 \\
0 & 0 & 0
\end{bmatrix}
\]

Mean Cost of an \(E_A^k\)

The execution plan of \(E_A^k\) is launched by the system when the mobile environment is in the state \(j\) of the dimension \(i\) with the probability:

\[q^k_p = \frac{\delta^k_{ij}p_{ij}}{\sum_{j} \delta^k_{ij}p_{ij}} \tag{1}\]

So, the mean cost due to dimension \(i\) of the \(E_P^k\) execution is given by:

\[c^k_i = \frac{\sum_{j} \delta^k_{ij}c^k_{ij}p_{ij}}{\sum_{j} \delta^k_{ij}p_{ij}} \tag{2}\]

Example 6 With previous \(P\), \(\Delta^k\) and \(C^k\) we can compute:

\[
c^2 = \frac{13.2 \times 0.1}{0.1} = 13.2\%
\]

\[
c^3 = \frac{33 \times 0.4 + 66 \times 0.6}{0.4 + 0.6} = 52.8
\]

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

\(E_A^k\) Initiation Probability

Definition 11 Let \(q^k\) be the probability for \(E_A^k\) of being selected for execution. \(q^k\) will be called the \(E_A^k\) initiation probability. Since the probability that dimension \(i\) has an acceptable state for \(E_A^k\) is given by \(\sum_{j} \delta^k_{ij}p_{ij}\), we have:

\[q^k = \prod_{i} (\sum_{j} \delta^k_{ij}p_{ij}) \tag{3}\]

So the execution plan of an \(E_A^k\) 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 \((p_{23})\):

\[P = \begin{bmatrix}
0.8 & 0.2 \\
1 - p_{23} & 0.2
\end{bmatrix}
\]

Fig. 2 shows the \(E_A^k\) initiation probability and Fig. 3 the battery consumption associated to each \(E_A^k\) in \(T_{AMT\text{-shopping}}\). It can be seen that if the bandwidth is often low (e.g. \(p_{23} = 0.8\)) then \(E_A^3\) is the alternative with the highest battery consumption (see Fig. 3) whereas it has the best initiation probability (see Fig. 2). \(q^1\) is constant because it does not depend on the bandwidth to be low but only on the probability for the catalog to be up-to-date.

4.2 Analytical Study of a \(T_{AMT}\)

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

Property 1 If a state of the environment is suitable for an \(E_A^k\) it should not be suitable for an \(E_A^{k'}\) 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 12 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 13 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, \(E_A^k\) is initiated with the probability \(q^k\) so the cost of the whole \(T_{AMT}\) is \(c^k\) with the probability \(q^k\), hence:

\[c_i = \frac{\sum_{k} c^k q^k}{\sum_{k} q^k} \tag{4}\]
As an example, we study the mean battery consumption and the initiation probability of $T_{AMT_{shopping}}$ in regards to the probability of the bandwidth to be low ($p_{23}$).

Fig. 2 shows that the $T_{AMT_{shopping}}$ 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 $T_{AMT_{shopping}}$ cannot be initiated. This figure also shows that the initiation probability of a single $E_{A_k}$ is smaller than the initiation probability of the whole $T_{AMT_{shopping}}$.

Fig. 3 shows that the mean battery consumption of the $T_{AMT_{shopping}}$ increases when the probability of the bandwidth to be low is going up. This is due to the fact that $E_{A_3}$ initiation probability is bigger than the one of $E_{A_2}$ (see Fig. 2).

Fig. 4 shows performance parameters for $T_{AMT_{shopping}}$ and a variant without $E_{A_3}$. This new transaction is called $T_{AMT_{12}}$. It can be seen that $T_{AMT_{12}}$ has a better initiation probability whereas $T_{AMT_{12}}$ has a better mean battery consumption.

4.3 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 depend on the number of defined $E_{A_k}$s but on the capacity to overcome environment variations.

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.

5 Related Work

The AMT model was inspired from DOM [4] and Flex [11] 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 [24]. In general, the adaptability vision is very limited, almost all studied systems adapt their behavior to support disconnections. That is the case in Clustering [21], Two-tier replication [15], HiCoMo [18], IOT [22], Pro-motion [26], Prewrite [20], MDSTPM [27] and Pre-serialization [9]. Only Moflex [16] addresses execution adaptability when hand-off occur. On the contrary, our proposition – AMT model – allows adaptation to any defined environment characteristic.

Concerning execution types, several works consider transaction initiated by an MH and completely executed on FHs. For instance, KT [10], MDSTPM, Moflex and Pre-serialization. Other proposals focus
on transaction execution on an MH: HiCoMo, IOT and Pro-motion. Only clustering and Two-tier replication consider two execution types, (1) on an MH (during disconnections) and (2) distributed among an MH and FHs (during connections). None of analyzed proposals face distributed executions among several MHs or among mobile and fixed hosts. TCOT [17] and UCM [1] consider those kind of distributed executions even though these works address principally the transaction validation process. With the AMT model it is possible to define transactions following the five execution types mentioned here. Indeed, supporting different execution types facilitates AMT adaptation to mobile environments.

Although data replication is not necessarily a transactional issue, it is at the heart of several works on mobile transactions. Propositions like Clustering, Two-tier replication and IOT consider that MHs contain replicated data. During disconnections, copies are modified through a kind of second class transactions. At reconnection, different reconciliation process are proposed (re-executions with first class transactions, synchronization of copies, etc.). Our work does not address replication aspects, we separate transaction and replication issues. We consider a multidatabases system where each site (mobile or fixe) is independent of each other. A similar approach is taken in KT [10], Pre-serialization, MDSTPM and Mollex. Nevertheless, in these works the multidatabases environment is composed only of DBMSs installed on FHs. MHs are considered only to request transactions.

Finally, the principal difference of the AMT model and existent works is that the description of the environment execution is attached to the transaction definition.

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 introduced a formal specification of the proposed model, this will allow to compare our proposal with other advanced transaction models. (3) 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.

Research perspectives include: (1) The implementation of an application developer utility in order to facilitate the design of AMT transactions. Environment awareness 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 current 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.

References

Annex A

A.1 Types of dependencies used in the AMT specification

Here there is a review of the types of dependencies used in axioms of the AMT model.

**Begin Dependency** ($t_j \ BD \ t_i$): $t_j$ cannot begin executing until $t_i$ has begun:

($\begin{equation}
\text{begin}_{t_i} \in H \Rightarrow (\text{begin}_{t_i} \rightarrow \text{begin}_{t_j}).
\end{equation}$)

**Abort Dependency** ($t_j \ AD \ t_i$): if $t_i$ aborts then $t_j$ aborts:

($\begin{equation}
\text{abort}_{t_i} \in H \Rightarrow (\text{abort}_{t_i} \in H).
\end{equation}$)

**Begin-on-Commit Dependency** ($t_j \ BCD \ t_i$): $t_j$ cannot begin executing until $t_i$ commits:

($\begin{equation}
\text{commit}_{t_i} \in H \Rightarrow (\text{commit}_{t_i} \rightarrow \text{begin}_{t_j}).
\end{equation}$)

**Weak-Abort Dependency** ($t_j \ WD \ t_i$): if $t_i$ aborts and $t_j$ has not committed then $t_j$ must abort:

($\begin{equation}
\text{abort}_{t_i} \in H \Rightarrow (\neg \text{commit}_{t_i} \rightarrow \text{abort}_{t_j} \Rightarrow (\text{abort}_{t_j} \in H).
\end{equation}$)

**Begin-on-Abort Dependency** ($t_j \ BAD \ t_i$): $t_j$ cannot begin executing until $t_i$ aborts:

($\begin{equation}
\text{begin}_{t_i} \in H \Rightarrow (\text{abort}_{t_i} \rightarrow \text{begin}_{t_j}).
\end{equation}$)

**Force-Commit-on-Abort Dependency** ($t_j \ CMD \ t_i$): if $t_i$ aborts, $t_j$ must commit:

($\begin{equation}
\text{abort}_{t_i} \in H \Rightarrow (\text{commit}_{t_i} \in H).
\end{equation}$)

A.2 Fundamental axioms

**Definition 1 Fundamental axioms of transactions**

$\forall \alpha \in E_{I_1} \ (\alpha \in H^i) \Rightarrow \beta \in E_{I_1} \ (\alpha \rightarrow \beta)$

A transaction cannot be initiated by two different events.

$\forall \delta \in E_{T_1} \ \exists \alpha \in E_{I_1} \ (\delta \in H^i) \Rightarrow (\alpha \rightarrow \delta)$

If a transaction has terminated, it must have been previously initiated.

$\forall \gamma \in E_{T_1} \ (\gamma \in H^i) \Rightarrow \delta \in E_{T_1} \ (\gamma \rightarrow \delta)$

A transaction cannot be terminated by two different events.

$\forall \alpha \in E_{I_1} \ (\alpha \rightarrow \text{commit}_{t_i}) \land (\exists \gamma \in E_{T_1} \ (\text{commit}_{t_i} \rightarrow \gamma))$

Only in progress transactions can invoke operations on objects.

A.3 Failure atomicity

**Definition 2 Failure atomicity**

A transaction $t$ is failure atomic if:

1. $\exists \alpha \in E_{I_1} \ (\alpha \rightarrow \text{commit}_{t_i} \ [p_i \ [ob] \in H]) \Rightarrow$

$\forall \alpha \ \forall \gamma \ (\gamma \rightarrow \text{commit}_{t_i} \ [p_i \ [ob] \in H])$

2. $\exists \alpha \in E_{I_1} \ (\alpha \rightarrow \text{commit}_{t_i} \ [p_i \ [ob] \in H]) \Rightarrow$

$\forall \gamma \ (\gamma \rightarrow \text{commit}_{t_i} \ [p_i \ [ob] \in H])$

**Theorem 1 Properties of atomic transactions**

1. If $t$ is an atomic transaction, $t$ is failure atomic.
2. A set $T$ of committed atomic transactions is serializable.

Proof of Theorem 1

The proof is done in [5].