Context-aware Mobile Services Transactions

Muhammad Younas
Department of Computing and Electronics
Oxford Brookes University
Oxford, United Kingdom
m.younas@brookes.ac.uk

Soraya Kouadri Mostéfaoui
Communication and Systems Department
The Open University
Milton Keynes, United Kingdom
s.kouadri@open.ac.uk

Abstract — with the ubiquity of handheld devices people can now access information and conduct online transactions virtually anywhere and anytime. In such flexible but less reliable environment transaction management technology is believed to provide service reliability and data consistency. Existing models developed for mobile transactions mainly focus on variation of conventional commit procedures and give little or no attention to the context-awareness which has become a crucial part of modern mobile applications. Given the high availability of context-aware services, mobile transactions should not only acquire required services but should also comply with the service context. This paper exploits the service-oriented computing technology and accordingly proposes a novel approach that manages transactions using the contextual information of mobile services. The proposed approach has several advantages. It manages transactions such that they adapt to the required service context as well as user’s needs. It also ensures the reliability of transactions and also increases their resilience to failures.

Keywords: context-aware computing, transactions, mobile services.

I. INTRODUCTION

Recent developments in mobile technologies, such as wireless communication networks, handheld devices and service standards enable people to access information and acquire services in a ubiquitous manner. For instance, Google Mobile provides users with access to a variety of services from their mobile phones, ranging from simple web pages through to products’ prices to driving directions. The service-oriented computing and web services technologies further facilitate the provision and consumption of mobile services [1] — i.e., find, select and develop (or compose) new services from existing ones in order to: provide enhanced functionalities, enable universal accessibility, and reduce operational and development costs through service re-use [18]. In the following discussion we use the term mobile or M-services (as in [3, 4]) to refer to web services in the mobile environment.

This paper investigates into the issue of Transaction Management (TM) in M-services environments. A transaction represents an abstract view of a sequence of operations that are involved in the execution of an application. TM (originated from database systems) has been used in a number of applications such as e-commerce, m-commerce, and engineering applications [2, 15]. We believe that TM is important for M-services, since protecting and managing the integrity of M-services’ outcomes is essential. Specifically, from the users’ perspective, TM should guarantee that services obtained using mobile devices are consistent with what users request and what service providers deliver. From the M-service providers’ perspective, TM must ensure that transactions are correctly executed, the enterprise has correct information about the outcomes of those transactions, and information held in an enterprise’s databases is maintained to provide a truthful and consistent record of the state of the enterprises.

Various models and protocols have been developed for mobile transactions. However, they are limited to the classical commit procedures. Consider, for example, a simple transaction that books a table in a restaurant. Current approaches will simply commit a transaction if the required table is available in a restaurant. They do not take into account the ‘context’ information such as “a table should be booked in a restaurant which is located nearby and/or a less crowded”. Existing literature contains significant work on context-aware systems (see survey papers for details [17, 6]). But to our knowledge, context awareness is not addressed in the transaction management in general and mobile transaction management in particular. The philosophy of context-awareness is that systems must automatically adapt their operations to the environment by taking into account context information such as current location, time, users’ needs and other environmental parameters. Thus, in M-services it is desirable that a transaction reacts to the contextual information and adapts its behaviour according to the changes in context.

Furthermore, current approaches are generally based on the conventional ACID (Atomicity, Consistency, Isolation, Durability) criteria [16]. Such criteria require that a transaction must not expose its intermediate results (isolation) and must be atomic; all its actions must be carried out or none is. However, we see that problematic. Accordingly, our paper proposes a relaxed set of transaction correctness criteria, called SACReD (Semantic Atomicity, Consistency, Resiliency, Durability) (previously developed in [15, 16]) and a protocol for enforcing them, which it argues are more appropriate to the context-aware, more volatile, dynamic and open nature of M-services transactions. Unlike ACID criteria, SACReD do not impose isolation policy thus allowing transaction to be partially committed. We believe that such relaxation is more suitable for the autonomous and dynamic nature of M-services. In
addition, resiliency property allows for alternative services wherein a given service fails due to system/network failures or if it does not meet the required context then an alternative service can be selected and executed instead. Such resiliency property is missing from the conventional ACID criteria. In summary, SACReD criteria fits well to the nature of M-services as:

(a) in M-services environments, there exist multiple services that provide similar functionality but possibly with different levels of quality. For instance, weather forecast can be obtained using the Yahoo weather service or the BBC weather service. Similarly, a flight can be booked using the Expedia or the Opodo services.

(b) M-services suffer from frequent failures due to the less reliable nature of the wireless networks as well as of mobile devices. Thus, providing facilities for alternative services will greatly increase service resiliency to failures.

The work presented in this paper is motivated from the ideas and experience of our previous work [14, 15, 16, 17]. Our previous work was concerned either with transaction management [15, 16] or context-awareness [14, 17] and not the conjoint research of transaction and context management – which is the focus of this paper.

The remainder of the paper is organized as follows. Section II reviews related work and establishes its limitations with respect to the requirements of M-services transaction management. Section III analyses the characteristics of Mobile services and identifies related problems. Section IV presents the proposed approach for M-services transactions. Section V details how we are evaluating the proposed approach. Finally, Section VI summarizes the paper.

II. RELATED WORK

There is a scarcity of literature on context-aware transactions in M-services. This section therefore reviews only TM techniques that are developed for mobile database transactions but can be considered for M-services transactions.

Holanda et al [10] develop an intelligent transaction scheduler using a combination of conservative and aggressive concurrency control protocols. The proposed scheduler is claimed to be context-aware in the sense that it automatically identifies changes in the computational environment and adapts to the appropriate concurrency control protocol. The efficiency of the scheduler is tested using the Mobile Database Community which represents a set of databases connected through MANET. Rouvay et al [11] propose Context Aware Transaction sErvice (CATE) which provides facilities for selecting appropriate protocol (among the conventional 2PC, PC and PA protocols) that meets the execution context. CATE is claimed to improve performance as compared to using only one commit protocol. The context is defined in terms of the commit and abort rates of a transaction. That is, based on the number of commit and abort transactions, an appropriate commit protocol is chosen for the processing of a transaction. However, the above approaches [10, 11] are limited to the classical concurrency and commit protocols and do not take into account the context information such as location, time, QoS, and so on.

Kumar et. al define a protocol, called TCOT, for mobile transactions [8]. TCOT, a variant of the classical two-phase commit (2PC) protocol, is based on a timeout mechanism. TCOT is claimed to have improved performance and throughput over 2PC protocol in managing mobile transactions. 2PC protocol enforces classical ACID criteria [15]. As described above 2PC and ACID criteria are inappropriate for M-services transactions. Lee et. al [9] introduce a mobile transaction model called High Commit Mobile Transactions (HiCoMo) in order to improve the commitment rate of mobile transactions. Using HiCoMo transactions, aggregate data can be updated in a situation when mobile units are disconnected. The Kangaroo Transaction (KT) model [5] takes into account the movement behaviour of mobile transactions such that they can hop from one base station to another as their mobile device moves. But in certain situations it may not help in reducing communication overhead. In summary, the above approaches are limited to the classical mobile computing environment and they do not consider M-services or context-awareness.

Further, WS-Transactions [12] aim to address issues related to 2PC-based protocols. WS-Transactions define a framework for providing transactional coordination of web services offered by multiple autonomous businesses. WS-Transactions have greater potential for M-services than the aforementioned approaches. Our work complies with WS-Transactions. However, the latter is limited to the specification of frameworks and they do not provide analysis and evaluation in terms of failure resilience or context-awareness in M-services.

III. M-SERVICES

Basic Concepts

M-services are software applications that represent a higher level abstraction of a set of activities that manipulate different resources in order to fulfill users’ requests [18, 3, 4]. The proposed protocol requires that M-services are developed using the Service-Oriented Computing (SOC) architecture and technologies. The most common implementation technologies of SOC are the XML web services which are built using WSDL, SOAP, REST and UDDI among others. SOC enables dynamic composition of loosely coupled distributed services in order to facilitate software reuse, provide enhanced functionality, reduce operational and development costs and to enable inter-organisational collaboration. M-services are published using service interface definition languages such as WSDL or CWSDL [13, 14] such that they can be consumed (invoked)
by service consumer applications such as M-services transactions.

SOC enables the provision of multiple alternative services (or instances). That is, a particular M-service will have multiple alternatives that provide similar functionality but with different levels of quality. As mentioned earlier, a flight can be booked using the Expedia or the Opodo services. But these may have different levels of quality and constraints such as performance, reliability, etc.

M-Services Characteristics

A generalized architecture of mobile systems (in Figure 1) is presented in order to illustrate the characteristics of M-services [21]. In such architecture M-services may reside on a fixed host (FH) or on a mobile host such as a mobile device. Base stations (BSs), controlled by a Base Station Controllers (BSCs), are capable of communicating with mobile devices through wireless networks. BSCs are in turn controlled by the Mobile Switching Centre (MSC). Each BS covers a particular area, called a cell. Handoff happens when a mobile device moves from one cell to another.

One of the main problems with M-services is that they frequently suffer from failures such as service failures or context-related failures.

Service failures occur due to several reasons. In M-services, transactions are less prescribed, more prone to failure, and open-ended. Second, characteristics specific to mobile devices and networks add further complexity. Specifically, the mobility of the processing units (e.g., PDA) means transactions and data stores may physically re-locate during execution, as their originating and other participating devices move. Also, connection with the mobile devices may be intermittent. Thus, for example, a transaction may originate at one site and terminate at another, and require frequent re-connections in between. Also, the bandwidth of mobile networks is currently low in comparison to wired networks. Consequently, communication links may be overloaded by high volume of information exchange. In addition, mobile processing units are currently less reliable and with fewer resources, than the conventional "stationary" units. For example, the limited power supplies of mobile devices and the relatively limited resources affect failure resilience of M-services, since they increase the probability of transaction failure, for example, due to a flat battery or running out of memory. All the above issues significantly increase the volatility and unreliability of the M-services transactions.

Though existing research considers the above failures they do not give attention to context-related failures of transactions.

Context-related failures occur in a situation where required context criteria are not met. In a restaurant booking example, a transaction will be considered as failed (aborted) transaction if it books a table in a restaurant that is outside the specified location. In this case, the service is available but it does not meet the desired context. However, with current TM approaches, a transaction will be successfully processed (committed) if it can book the table in a restaurant irrespective of the location. Another example of context-related failure can be a performance failure, that is, a transaction will fail if it does not respond within a specified time interval.

IV. THE PROPOSED APPROACH

This section first describes the different types of context. It then presents our M-services transaction model. Finally, it illustrates the protocol developed for M-services transactions.

M-services context:

In the proposed approach we introduce an instantiation and a combination of Dey's [19] and Schilit's definitions [20] of context as our definition.

Context is then defined as "any information that can be used to describe the situation of people, resources and services in a service-oriented environment. It may include all other information that can be considered relevant to the interaction between a user and a service".

Context information of M-services can be obtained into two different ways and hence can be classified into two categories:

- **Internal context**: it represents context information related to the service and can be directly obtained from the service interface definition, for example, using the Context-Based Web Service Description Language (CWSDL) [14]. CWSDL enhances the standard W3C-WSDL in order to provide support for context-related information. CWSDL defines the service profiles and the related quality metrics for rating, comparing and selecting the desired services. Using CWSDL, service providers can attach different quality factors (e.g., response time) and

---

Figure 1: Generalized architecture of mobile systems

![Diagram of a generalized architecture of mobile systems](image)
constraints (e.g., service is not replaceable, maximum number of instances) to the M-services.

- **External context:** it represents any context information which is external to the service. This information is usually obtained using external (independent) services. For example, location service can provide location information about a hotel service and a weather service can provide weather forecast about a particular location.

*M-services transaction model:

M-services transactions can range over a wide range of services involving and spanning many enterprises distributed over the wired/wireless network. Accordingly, we define an M-services transaction as the execution of a (composite) service which can be divided into well defined units (or component service transactions) that provide correctness and consistency of M-services. M-services transaction is therefore decomposable into component transactions representing individual M-services. For instance, the individual services involved in the restaurant booking (such as table booking, or restaurant location), can be represented as component service transactions.

An M-service transaction, denoted MS_T, is defined as a tuple, MS_T = (cst_i, <); where cst_i, (1 ≤ i ≤ n) is a set of component services transactions, and < is a partial ordering of the cst_i, which determines their order of execution. A cst_i ∈ MST is compensatable if its effects can be semantically undone by executing a compensating service transaction. It is replaceable if there is an alternative service that can perform the same task.

Given the nature of M-services transactions, classical ACID criteria are proved to be inappropriate. We therefore propose the SACReD (Semantic Atomicity, Consistency, Resiliency, Durability) criteria [15, 16] for M-services transactions. That is, M-services transaction is required to meet the rules set by the SACReD criteria. Semantic atomicity allows partial commit of MST such that individual

M-Services transaction protocol:

The proposed protocol is implemented by the following components: M-services Transaction Coordinator (MTC), Component Transaction Coordinators (CTCs) and various M-services (MS) such as a restaurant service, or a weather service. Referring to Figure 1, MTC can be deployed at fixed hosts where CTCs and M-services can be deployed at fixed as well as mobile hosts. MTC and CTCs maintain log files in order to record the required information about the execution of M-services transactions. M-services transactions are coordinated by the MTC. For example, the parent transaction, MS_T, that makes restaurant booking will be coordinated by the MTC. The individual services are acquired using component service transactions, cst_i, which are managed by the CTC. Note that M-services represent both the required services (e.g., table in a restaurant) and the associated services used in collecting context information (e.g., location service).

The proposed protocol comprises the following phases.

**Phase 1: Service reservation**

This phase concerns the service reservation. The objective is to tentatively hold (reserve) individual services (e.g., a table in a restaurant) for a transaction, MS_T, prior to the collection of the context information about services. It adopts the W3C Tentative Hold Protocol (THP) [7] for service reservation. THP allows multiple transactions to tentatively hold the same service. Note that THP is adopted because it is not feasible in the M-services environment to employ classical locking protocols and to hold on services for a longer duration.

The service reservation phase, implementing the THP, works as follows:

1.1 As a part of M-services transaction, MS_T, a service reservation request is submitted to the M-services Transaction Coordinator (MTC). MTC records the start of MS_T in a log file in order to keep the necessary information about MS_T and its component services transactions, cst_i. This is represented as step 1.1 in Figure 2.

1.2 MTC in turn submits that request to the concerned Component Transaction Coordinators (CTCs), which are responsible for reserving the required services (step 1.2a). After receiving the request, each CTC performs local processing using a M-service policy module to determine whether the required service can be tentatively held for the given transaction. The policy
module implements the rules set by the service provider that relates to service reservation such as timeout for a tentative hold, number of tentative holds on a given service and so on. (Figure 2 represents these steps as 1.2b-c.)

1.3 If a service is available for reservation then the concerned CTC sends a message to the MTC (step 1.3) telling it that a service is tentatively held. Tentative holds have to comply with the rules defined in the M-service policy module.

1.4 If the service is unavailable and if there is an alternative service (e.g., booking a table in a different restaurant), the MTC then contacts the CTC for the reservation of the alternative service (as in step 1.1). In the proposed protocol, service replacement is possible according to the resiliency property of the SACReD criteria. CTC then follows the above procedure for reserving the alternative service (steps 1.2a,b, 1.3).

1.5 Once a service is reserved, MTC then starts collecting context information about the service (see Phase 2).

1.6 In case of the commit of MS_T (Phase 3), the CTC will notify the requests (from other transactions) about the situation. That is, it will inform them that their tentative holds on the service are no longer valid. As described above this situation arises as THP allows multiple requests to tentatively hold the same service.

I) Phase 2: Context gathering

After reservation, MTC then starts collecting context information about the required service. Generally, the decision on determining the overall context of a service is based on more than one type of context information. That is, MTC has to collect different types of context information such as location of a restaurant, menu and prices.

2.1 MTC contacts each CTC (involved in M-service transaction) to provide context information. CTC can either collect context information from a separate independent service (e.g., location service) or from a service interface definition, e.g., using CSWDL (see section on “M-services context”). Note that the former method is more common in M-services environment. Figure 2 represents this step as 2.2a-c. For each context information, a listener is attached that informs the CTC whenever there is a change in context?

2.2 Each CTC sends the required context information to the MTC. For example, CTC representing a location service will send the restaurant location information.

2.3 After receiving all the information, MTC decides whether the required service meets the desired context. Making such decision is not trivial as it has to take into account several factors. First, context is not static and it changes over time (such as distance to a restaurant). Second, the overall context can be aggregated from several individual contexts. For example, contexts such as restaurant location, menu and prices can be aggregated in order to make the overall decision. This is represented as step 2.3 in Figure 2.

2.4 If a service meets the required context information, MTC then starts the commit process (Phase 3). If not, MTC looks for alternative service (if available) and repeats the above steps for determining the context of the alternative service. Before determining the context, the required service should also be reserved (as in Phase 1).

Phase 3: Commit process

3.1 The coordinator checks if there are no messages related to the change in context. This is important as the context requirement must be fulfilled as a part of the commit process. If there is any change in the
context, the coordinator must take it into account in deciding the commit of MST.

3.2 If the MST still meets the required context, MTC then contacts each CTC about the reserved services (step 3.1). If not, then follow step 2.2.

3.3 If the tentative hold on the reserved service is still valid, each CTC commits the component service transaction (cst), and sends a commit vote to MST. It is also possible that a committed cst needs to be cancelled. In that case, each CTC must execute the compensating transaction for cst (canceling service). If the tentative hold on the reserved service has expired, the concerned CTC sends an abort vote to MST, and declares the cst as aborted. Figure 2 represents this step as 3.2a-c.

3.4 When MST receives commit decision from all concerned CTC, it commits the MST, by logging the commit decision. It then terminates the MST and starts a new MST (if any) (step 3.3).

V. PROOF OF CONCEPT

As a proof of concept we have implemented the approach presented in this paper within the “Smart Campus” prototype. The aim of the Smart Campus prototype is to assist students and university staff with a portable interface that provides them with information of interest (services) inside a University campus. Examples of such services might be: directions, exams’ results, library reminders, restaurants’ menus, restaurants’ bookings, meeting rooms’ occupancy schedules, and other suggestions upon their contexts.

The system is capable of detecting users' locations and communicating through a wireless infrastructure with the MTC so as to gather data and services requested by the users. For this purpose, mobile devices like palm pilots, mobile phones and laptops are both offering and requesting services (which can be represented through CTCs as component services transactions). Multimodal interactions are used to keep users as much comfortable as possible.

Two mechanisms are implemented: the automatic and the request based service provisioning. In the automatic service provisioning, once certain context conditions are met, services are automatically triggered by the MTC and the result is offered to the user without an explicit request. For example the library reminder service is triggered when the deadline is approaching, the restaurant's menu is sent to the user when it is lunch time and the user is not in class. Request based services correspond to the answer of the users' queries. For example the user explicitly requests the room occupancy schedule service to check if she can plan a meeting there.

As illustrated in Figure 3, for a number of user queries (corresponding to transactions) users have been asked to provide feedback about the quality of the services offered. The feedback consists in rating the different parameters related to quality of the services. Users are indeed asked to fill-in a satisfaction questionnaire at the end of each service offer. The questionnaire consists of a set of questions, related to the usefulness of the services offered, their degree of adaptability, the amount of interaction required for the users, etc.

We have intentionally made the questionnaire simple and concise to meet the requirements for the portable devices used in this prototype.

The results show greater satisfaction with our proposed approach. Indeed, feedbacks show that for the request-based services an average satisfaction degree of 73.9% has been achieved with the proposed approach comparing to an average of 57.82% without its adoption, and an average satisfaction rate of 61.9% has been achieved with the proposed approach comparing to an average of 53.30% without its adoption for the automatic service.

These results are indeed very satisfactory and encouraging and clearly show that the quality of the proposed services is improved when context information is added — 16.08% more satisfaction for the request based services and 8.06% for the automatic services.

However, as we can see from the results better satisfaction has been achieved with the request based
services comparing to the automatic services. This issue needs then further exploration.

We are now in the process of performing extensive evaluation of the proposed approach, to evaluate the global throughput of our approach when applied to other prototypes.

**CONCLUSION**

This paper proposes a new approach that manages transactions such that they comply with the required context of the desired services and adapt to the environmental conditions and user’s needs. Our ongoing work is decomposed into several thrusts. It includes the evaluation of context-based global throughput and performance efficiency and failure recovery. Security is also important to make sure that services do not misuse the computing resources. Another thrust consists of composite service adaptability by allowing MTC to carry out some run-time modifications, for instance, by adding new services, removing or replacing certain services. Therefore, the assessment of the effects of adaptation is deemed appropriate.

**REFERENCES**


[21] M. Younas, K-M Chao, R. Anane "M-Commerce Transaction Management with Multi-Agent Support" 17th International Conference on Advanced Information Networking and Applications (AINA'03), March 27-29, 2003, Xi'an, China