Protocols for Committing Mobile Transactions

Nadia Nouali[1&2], Habiba Drias[2&3], and Anne Doucet[4]
[1]Mobile Computing Department, CERIST, Algeria
[2]Faculté du Génie Electrique, Université Houari Boumediene, Algeria
[3]Institut National d'Informatique, Algeria
[4]LIP6 Laboratory, Université Pierre et Marie Curie, France

Abstract: *Mobile computing has attracted attention of intensive researches during the recent years. Many papers revisit the conventional implementation of distributed computing paradigms for use in this new environment. A key paradigm of the transaction processing is the transaction commitment. A commitment mechanism such as Two Phases Commit (2PC) protocol, a fundamental asset of transactional technology (and its variants), ensures consistent effects of a distributed transaction. This paper surveys the solutions proposed for mobile transaction commitment and outlines how the conventional commit protocols are revisited in order to fit the needs of a mobile environment. The different approaches try to deal with the slow and unreliable wireless links, the lightweight devices and their limited resources, the frequent disconnections and the movement of mobile devices.*

Keywords: *Mobile transaction commitment, disconnection, mobility, resources constraints, wireless communication.*

Received November 3, 2004; accepted February 21, 2006

1. Introduction

In distributed systems, an Atomic Commitment Protocol (ACP) is needed to terminate distributed transactions. A transaction is a paradigm designed as a data access consistency mechanism and its use is widespread in many different kinds of computing systems. A transaction is defined as a set of operations that form a *logical unit of work*. The essential idea of a transaction is *indivisibility*, i. e., either all the operations of the transaction are permanently performed or none of them is, and its partial results are not visible to other transactions. Traditionally, a transaction semantic is defined by the ACID properties: Atomicity, consistency, integrity and durability. In a distributed environment a transaction T may involve multiple parties, namely resources servers (generally data base servers) where its operations are executed. To preserve data consistency, the *all or nothing* effect of the transaction (namely A and D properties) is usually enforced at the commit time of the transaction [3]. The most commonly used mechanism to deal with the commitment problem is the *Two-Phase Commit* (2PC) protocol that allows the involved parties to agree on a common decision to commit or abort the transaction even in the presence of failures. Some well known standards such as ISO [12], X/OPEN [29] and CORBA [21] include the specification of services that provide the 2PC protocol semantics. Much has been written about this protocol and its variants. The most popular variants, i. e., Presumed Commit (PrC) [19], Presumed Abort (PrA) [19], Early Prepare (EP) [26], etc, attempt to minimize execution overhead, in terms of message traffic and log writes. The abundant literature about 2PC protocol and its continuing refinements until recently shows the importance of this topic for distributed systems [28] and encouraged us to take it as a basis for the study of the effects of mobile environment by the transaction commitment. In mobile environment earlier researches have generally focused on the design of new transaction models mostly relaxing ACID properties [5, 6, 7, 8, 16, 17, 18, 20, 23, 24, 27], in this paper we precisely focus on the ACP mechanisms.

Like many other protocols in distributed computing, the 2PC protocol assumes that all the communicating partners are stationary hosts, equipped with sufficient computing resources and power supply, exchanging messages over wired networks with a permanently available bandwidth. These assumptions are no longer valid in the new wireless environment where hosts may be portable devices equipped with more or less resources (CPU, memory, and power) and communicating over wireless links. Wireless communication induces much lower bandwidth, higher latency and error rates and more expensive cost. The objective of this paper is to prospect about the problems faced to insure transaction commitment in the new mobile wireless environment and to highlight the key issues to deal with in the design of ACP protocols for mobile distributed transactional systems.

The remainder of this paper is organized as follows. Sections 2 and 3 present the mobile environment characteristics and their impact on the 2PC protocol execution. Section 4 reviews papers related to the

commitment topic in mobile environment. In section 5, we highlight the principal ideas which emerge from this study. Section 6 concludes the paper.

2. System Architecture

We adopt the system model depicted in Figure 1 that is a largely accepted architecture in the literature of mobile computing research [4, 8, 11, 14]. A mobile system is a distributed one that supports mobility. The global architecture consists of two distinct sets of entities: Mobile Hosts (MH) and Fixed Hosts (FH). A MH is a computer that can move while maintaining its network connection through wireless links. MHs are connected to the fixed part of the network via a special type of FH called Base Stations (BS) or Mobile Support Stations (MSS). A BS is a computer augmented with a wireless interface to communicate with mobile hosts. BSs communicate with the other fixed hosts via wired links. Each BS covers a geographical area called a cell. A mobile host can directly communicate with one base station, the one covering the geographical area in which it moves. Due to its mobility, a MH may cross the border between two different cells while being active, this process is called *handoff*. The handoff process is under the BS responsibility. We assume that certain FHs are equipped with public databases and that certain MHs may also be equipped with personal databases. For simplicity purposes, we also assume that BSs have some processing capability such as interpreting MHs and FHs requests.

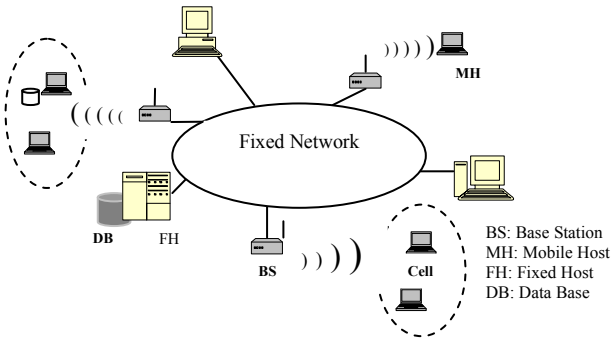

Figure 1. Mobile system architecture.

There are differences between a traditional distributed system and a mobile system. First, MHs have limited capabilities compared with those found in the fixed case like slow CPU speed, little memory, low battery power, small screen size. Type of connectivity is a major second difference. In a distributed system fixed hosts are connected to the network through continuous high-bandwidth links, while in a mobile system devices are connected via wireless links (i. e., GSM network, satellite, WaveLan, HiperLans, Bluetooth) characterized by a very lower bandwidth and big latency. In the fixed system the hosts are rarely disconnected. In the mobile scenario; the units are frequently disconnected involuntarily while roaming or because of a damage, or voluntarily, for example, to save battery power. The mobility of portable devices adds new difficulties to deal with such as handoff situations. In the next section we will show how these new characteristics impact the ACP paradigm.

3. Executing 2PC Protocol in Mobile Environment

The operations of a distributed transaction T_d can execute on different sites (hosts) widespread over the network. The subsets of operations executed on the different sites are called transaction branches. The 2PC protocol follows two steps or phases (see Figure 2): A voting phase and a decision phase. In the first phase, the site where the transaction was originally initiated *(generally called the coordinator)* asks all the sites involved *(participants)* in the transaction to prepare to commit the transaction *(prepare message)*. If, for any reason, including a concurrency control problem or a storage failure, any of the participants responds *No*, the coordinator decides to rollback the local branch of transaction and sends *Abort* messages to all participants. If all the received responses are *Yes*, the coordinator then decides to commit the local transaction branch and informs all the participating sites by *commit messages*. A *Yes* vote indicates that the local operations have been successfully executed and the updates could be made permanent or durable even if a failure occurs. A participant that votes *No* can unilaterally abort its transaction branch, whereas a participant that votes *Yes* must wait for the coordinator decision to abort or commit its branch. The participants acknowledge the coordinator decision. During the protocol execution, the coordinator and the participants keep, in stable storage, private logs which contain transaction control (prepare, commit, abort, end-transaction) and data manipulation records (updates, undo and redo information). The coordinator's log contains in addition to control and data manipulation records, the identities of all the participants. The logs are used during failure or crash recovery. Since failures may occur at any time, some records are *force-written* (written by blocking I/O) to a reliable stable storage.

In this paragraph, we analyze the main features of 2PC protocol and identify the problems they raise in mobile context. We assume that a mobile transaction T_m is issued by a mobile client executing on an MH. If we follow the 2PC scenario described above, the mobile client would play the role of coordinator. The responsibility of the coordinator means that some processing and storage capabilities are needed. But, MHs are generally lightweight and do not have sufficient resources. In addition, the number of messages to be exchanged between the coordinator and the participants over the wireless link will be too large.

This may lead to an unbearable cost in terms of communication and energy consumption. The messages will all transit by the BS the client is attached to, thus increasing the latency of the protocol. In addition, keeping logs on a mobile client is unreliable because MHs are more prone to loss, damage or theft. Even if the logs are stored on the fixed part, this still leads to unnecessarily and costly messages overhead in the case the coordinator is kept on the MH.

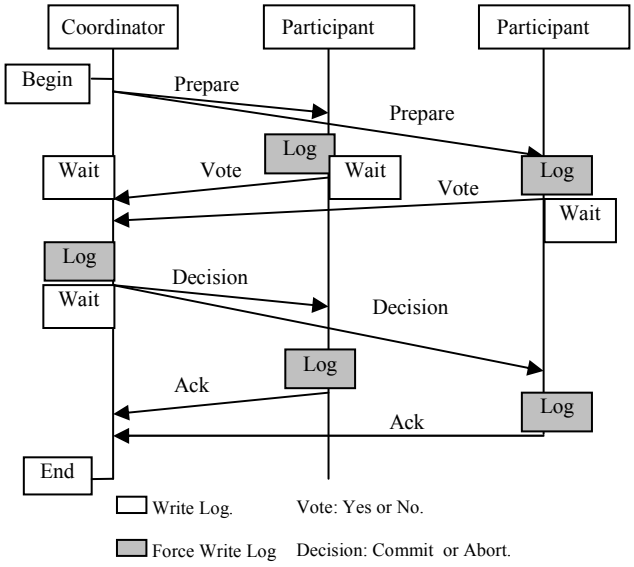

Figure 2 . The 2PC protocol.

The first intuitive modification that comes to mind is to change the coordinator location: Send the transaction in batch to be executed on the fixed part of the network and execute the coordinator on the BS [20]. The mobile client may act just as a participant. Thus the number of messages exchanged over the wireless network, especially in up-stream, is significantly reduced and the logs will be kept more safely. Communication between participants and the coordinator do not transit via a BS. The scarcity of the MH resources does not affect the commit process. However, this schema does not take into account the *mobility* of client MHs. When a MH moves from one cell to another, it will not be able to communicate with the BS (where the coordinator is homed) of the cell it has just left. This may impact more severely failure or crash recovery situations where the coordinator and the client may need to get in touch to terminate the transaction. Furthermore, batch execution of a transaction is not suitable for interactive applications when user intervention is needed to either introduce data or get intermediary results necessary for decision making. Say for example, a certain itinerant salesman at the house of a client needs to know the state of the stocks for a set of components of an item. He launches a transaction that in its first phase gets information from a central database server and outputs it on the portable device screen. Then the salesman decides, according to his client's desire, on whether to execute or not the second part of the transaction by submitting the command and eventually payment information. The same need of interactivity may exist in the case of a doctor present bedside of a patient (be it at the latter's home or in a room at the hospital) interacting with the hospital central DB to get more details about the treatment the patient received during its last stay at the hospital and eventually fix and record the next appointment of medical control at the hospital and the examinations to have according to his state.

Similarly to *mobility*, MH *disconnection* is an intrinsic characteristic of the mobile environment that impacts the 2PC protocol execution. Disconnections may occur voluntarily for a variety of reasons, for example, when the user deliberately cuts communication to not being disturbed while in a meeting, to reduce cost or power consumption or to save battery life. Disconnection may also occur involuntarily and unpredictably for example when the MH enters a non covered area (while in a train entering a tunnel), if battery runs out of power, because of a device damage or theft. It is important to note that disconnection of the MH can result in a failure if a FH tries to communicate with it without success. In other words, if the traditional 2PC is executed in mobile environment, disconnections will increase the number of, may be unnecessary, abortion decisions of transaction. As frequent are disconnections, as transaction abortions are. Things can be worse if these disconnections are *long*. This is not acceptable in mobile environments because frequent disconnections are not exceptions but rather are part of the normal mode of operation, so they should not be treated as failures.

In the more general case where T_m involves mobile servers the same analyse can be done. Preserving scarce resources, handling disconnections and mobility are to be considered for the same reasons as above. The papers reviewed in the next section have adopted different ways to deal with the problems described above we compare their approaches and try to show their advantages and limits.

4. A Survey of Commit Protocols for Mobile Environment

Much literature concentrate on studying ACP especially 2PC protocol [9, 10] and its variants PrA [19], PrC [19], EP [26], Coordinator Log (CL) [26], Implicit-Yes Vote (IVY) [2] in the distributed environment. A survey and analysis of ACP problem in the traditional environment can be found in [1] and interesting experiment results are also given in [15]. In the mobile computing literature, two main approaches are followed when it comes to the ACP problem. One approach focuses on the fact that strict atomicity is not adequate and rejects the 2PC mechanism, thus new

protocols are especially designed to meet the mobile environment requirements [13, 22]. These protocols follow an optimistic approach and sometime tolerate weak or semantic atomicity by admitting the concept of *compensation* as in [13, 25]. The other approach tries to adapt the 2PC or its variants to mobile environment. For example, [4, 14] adapt the 2PC and a variant of 1PC (one-phase) ACP respectively. Perron *et al.* propose M-2PC (mobile-2PC) in [22] and compare it to a new time stamp based protocol called Optimistic Concurrency Control with Update Time Stamp (OCC-UTS*)*. Kum *et al.* in [13] propose M2PC protocol and compare it to a completely new timeout based commitment protocol called Timeout-based mobile Transaction Commitment Protocol (TCOT*)*. But, these papers do not really concentrate on optimising the 2PC protocol to make it work with adequate performance in mobile. They show the drawbacks of executing it in mobile environment and focus on proposing a new way of committing a mobile transaction.

4.1. Mobile 2PC Protocol

In [22], authors' objective is to minimize the number of messages exchanged between mobile clients and a fixed server. They propose the Mobile 2PC (M-2PC) protocol an optimization of the traditional 2PC to send transactions in batch from the MH to the current BS which becomes the coordinator. The initiating MH acts as a participant. The control of transaction hands-off when the MH hands-off. That is, the uncompleted part of the transaction is sent along with the state information to the new BS which becomes the new coordinator. With each change in geographical location the MH will send a message to inform the BS that a handoff needs to be performed. By launching a transaction in batch the user interactivity with the transactional application is severely limited. Some applications may need this interactivity. Maintaining the MH as a participant means that the coordinator will continue to send messages to it (prepare downlink, vote uplink, decision downlink). In effect, all the BSs that participated in a transaction are also involved during commitment decision. 4*h* additional wired messages are needed if *h* is the number of hands-off. For *n* participants including the transaction MH initiator *(4n + 4h)* wired messages are generated. Another limit of this protocol resides in its assumption that if a MH fails in one cell it will recover in the same cell. This means that it does not manage the situation where disconnection and handoff occur simultaneously.

4.2. Optimistic Concurrency Control with Update Time Stamps

The second protocol proposed in [22] is called Optimistic Concurrency Control with Update Time Tamp (OCC-UTS) which basic idea is to verify that a transaction is serializable before deciding if it should be committed or aborted [22]. The protocol assumes the existence of a local cache at the client site and that the transaction executes locally offline before committing at the server. The protocol uses a backward validation of serializability by checking if a committing transaction is invalidated by any transaction that has already committed. Timestamps are associated with data items and used to compare the client data stamp with the last update stamp maintained at the server. To validate its data, a transaction checks invalidation reports that are broadcast periodically by the server. If the client locates a data item it has updated in this report, it can roll back the transaction without exchanging messages over the wireless link. Otherwise, a request to commit message is sent to the server.

This protocol is suitable for PDAs or portable computers relatively well equipped in order to provide local application execution (offline execution). It provides a way of minimizing the number of abortions by verifying serializability at the client side before attempting to commit the transaction at the server. However, in data base servers with a heavy load, the MHs may wait for a long time and timeout before having their messages processed. The clients are MHs, no mobile server is taken into account in this schema. OCC-UTS is analytically superior in terms of wireless messages. But, it is also concluded that M-2PC could be further optimized and could not be discarded. Its most advantage is its possible uses in hybrid mobile networks, i. e., networks where both mobile and static nodes interact with the same database. Because the implementation of the protocol on the database servers is no different from classical 2PC, it should not be too difficult to have the two working together.

4.3. Timeout-Based Mobile Transaction Commitment Protocol

Reference [13] proposes a new protocol for commitment, the protocol TCOT that was compared with M2PC protocol. M2PC is an adapted version of 2PC protocol to mobile environment. The modifications applied to 2PC protocol make it work in a similar way as TCOT, but M2PC is not studied in depth. The authors show that their protocol commits transactions in mobile database systems with minimum number of uplink (client to server direction) messages compared to M2PC. Contrary to [22], they envision a system offering a connectivity mode called *mobile connectivity* that allows clients to remain connected all the time to the network through the wireless channel irrespective of their states (mobile, static, dozing, etc.) and location in opposition to the *intermittent connectivity* mode where a client voluntarily decides when to connect/disconnect to/from the network.

TCOT is an optimistic approach that assumes that well defined timeouts may lead to fragments completing and committing within it at each participant site with a minimum risk for abortion [3]. The authors have proposed to calculate the timeout as a function of many system and communication variables (workload, I/O rate, cache hit rate, etc.). In TCOT a transaction T_m is split into fragments that will be executed on different sites (the first fragment is executed at the initiating MH). The BS coordinator sends the fragments to the relevant DBSs for update of the primary copy. The participant sites take a local commit/abort decision about their fragments and inform the coordinator. MH sends its log and updates to the coordinator which forwards them to the relevant DBS once it takes the final decision about Global commit/abort. Let E_t being an upper bound of the execution time, i. e., just long enough to allow a fragment to successfully finish its entire execution on a participant site and S_t the upper bound of data shipping time from the MH to the DBS. The Global commit is decided by the coordinator if it receives the updates log from the MH before S_t (shipment delay) expires and commit messages from all other participants. Otherwise, if the total time found by summing all E_t (execution time at the participant site) values and the S_t value expires, or an abort message is received then a Global abort is decided and all participants are informed. If a participant receives the abort message after local commit is performed then a *compensating* transaction is needed. In the case of handoff the authors propose either to choose the coordinator statically or dynamically. In the later case, when the MH moves, it informs the BS about its previous coordinator and participant Database Servers (DBS) during registration. The new BS informs all DBSs about this change with no additional traffic on wireless links.

Correctness and performance analysis of TCOT [13] shows that in normal execution only 2 messages are required whereas 3 messages are needed with M2PC. However, in the case of a fast moving and frequent disconnecting MH, TCOT increases the abort probability and consequently the number of wireless messages also increases exponentially. The E_t formula of TCOT was obtained from an analytical analysis, however empirical analysis is needed to provide more realistic verification. In addition little attention has been taken concerning the choice of S_t timeout and its effects on the protocol performance. This is an important factor because it is a function of networking parameters of wireless links that are extremely varying. The important assumption is that mobile hosts remain connected all time over wireless links which seems unrealistic in today mobile environments. TCOT may perform well in an environment where communication over wireless links is highly available and reliable; otherwise, the abortion rate may be unbearable for many application domains. Furthermore, the compensation may not satisfy certain applications where strict global ACID properties are required. Another issue to consider is how TCOT will behave if certain DBSs are mobile.

4.4. Unilateral Commit for Mobile

The goals of Unilateral Commit for Mobile (UCM*)* protocol [4] are to support off-line processing of transactions, lightweight and moving client and servers. A transaction executing offline can commit as soon as its log has been transferred on the BS without waiting for acknowledgement of the fixed servers because all the verifications take place before commit time. During the UCM execution some servers can disconnect, legacy systems (which do not export a standard 2PC interface) are accepted and UCM uses a single message round thereby saving wireless communications. UCM protocol is based on the idea of One-phase Commit that has been suggested in [9] and taken up in many variations such as the Early Prepare protocol, the Coordinator Log protocol [26] and the Implicit Yes-Vote protocol [2]. 1PC eliminates the Voting phase of 2PC, during which the coordinator verifies whether or not the participants can locally guarantee the ACID properties, by having these properties guaranteed at commit time at every participant site, i. e., during the execution phase of the transaction [4].

The paper is based on a network infrastructure similar to that depicted in Figure 1 with the possibility of including smart card based devices (for example a cellular telephone equipped with a SIM card) as a case of the lightweight servers. Five types of software components interact in the execution and termination phases of a transaction, namely the A*pplication* initiating the transaction operations, the *LogAgent* that logs each operation before execution, the *Participants* that execute these operations, the *Coordinator* that pilots the termination protocol and the *PAgents* that represent the participants during the termination protocol and are used during recovery. The *Coordinator* is always located on the fixed network while the other components can be hosted by a MH. During the execution phase, the *LogAgent* registers each operation before it is sent to the participant for execution by a non force-write. Each operation is acknowledged up to the application. At termination time, the application issues a commit request after receiving all the acknowledgements. At this point, the ACI properties are locally verified by the participants. The Durability property is insured by the *Coordinator* which gets the log from the *LogAgent* and force-writes it before broadcasting the decision to the participants.

Disconnections are treated by adopting the 1PC approach in which some sort of local pessimistic concurency control and immediate integrity control is

required to maintain the ACI properties before the commit time. This does not cause any problem in the case of lightweight servers such smart cards or cellular phones that are not likely to support parallelism (one user at a time). However, this is not always possible if the participants are not capable of providing such mechanisms. Recovery procedure accounts on the coordinator logs to re-execute a branch. This limits, to some extent, the autonomy of the participants, thus the availability of the coordinator logs is crucial to the success of the protocol. Moreover, the vulnerability window (or uncertain period), during witch a failure requires special recovery action, is longer than in 2PC as now it stretches from the beginning to the end of the transaction. The mobility/handoff problem was not explicitly treated by UCM.

4.5. Two-Phase Commit Protocol for Low-Powered Mobile Clients

Authors of [14] present a schema that we call in short L-2PC as an adaptation of the traditional 2PC protocol for mobile computing. It is designed to support location-dependent information access by dynamically allocating the most-fit service supplier for the on-the-move client. In summary, the protocol is designed for the commitment of a set of services executing on a set of servers which are spread over IP-like network. An m-commerce application on mobile Internet is an appropriate example. Four main modules are involved in the protocol: The *client* is the issuer of the commit request, the *current proxy* is the BS of the cell within which the mobile client is currently located, the *coordinator* is the BS who first receives the commit request and the *worker* is the server providing needed service in the commit request. Upon receiving the commit request, the coordinator tries to find and allocate qualified workers who physically supply the needed service. If it finds them and all promise to finish the job, the coordinator decides to really commit the transaction and inform all the participating workers. Otherwise, if the coordinator can not find a worker for a specific service, it informs all the workers to abort and rollback their jobs. After receiving the ACKs from all the workers, the coordinator finds the current proxy of the mobile client and forwards the result to it. Then the proxy delivers the result to the client which also sends an ACK to the proxy. The latter sends ACK to the coordinator who releases the resources.

In the L-2PC protocol, when the client moves to a new cell, a proxy handoff takes place. Disconnection handling is simplified as after submission of the request the coordinator takes over all succeeding job, so the client can disconnect during the service session. Power of mobile clients is saved by shifting the most workload to the fixed hosts and freeing them from staying connected during all the commit process. In our opinion the particularity of this protocol is the way it deals with the handoff problem as it embeds the mobile-IP concept at the application level. However it is clear that this is done at the price of introducing an additional messages cost to the original protocol.

4.6. Semantic Atomicity CO2PC Protocol

The CO2PC protocol [25] provides semantic atomicity by allowing participants to perform either optimistic local commit (locally committed results are shared) or non-optimistic commit. The authors 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 decision. CO2PC protocol is a Combination of an Optimistic approach and 2PC. The CO2PC coordinator must be a participant FH. If there is no FH participant then an Agent will play this role. A participant making optimistic commit (called *opt-participant*) executes its transaction branch and commits/aborts it unilaterally. Then, it sends its *vote* to the coordinator. If the global decision is *commit*, *opt-participants* are done; if it is *abort* they have to launch compensating transactions. A participant making non-optimistic commit (called *non-opt-participant*) executes the 2PC protocol locally with the underlying DBMS. Each DBMS sends its *vote* to the Client/Server which forwards it to the CO2PC coordinator. If the *vote* is *abort*, the participant aborts the transaction branch unilaterally; otherwise the branch 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. 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. Commits must be done before timeouts expire. When a participant *votes commit*, it deactivates its timer and has to wait for the global decision. If the timeout expires before the coordinator has all *votes*, abort or compensation is used. 2PC is used by CO2PC because – for non-compensable transactions – resources must be retained until a global commit/abort. Notice that 2PC is not used for the global coordination of atomicity. Thus, CO2PC is proposed to address several kinds of transactions (non-compensable as well as compensable ones) and different types of MHs (with limited and unlimited resources).

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 Client log as well as in the corresponding Agent. In order to preserve DBMS heterogeneity, the authors 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 compensating transactions.

In CO2PC, after the *vote* is sent, an MH may disconnect temporally. In that case, the coordinator *decision* is logged in an Agent 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).

5. Discussion

Table 1 and 2 summarize the principal properties of the protocols studied above. To commit a transaction, the best protocol in terms of wireless messages is UCM. However this is obtained at the price of making strong assumptions about the local concurrency and recovery mechanisms and this may limit its usability in arbitrary heterogeneous systems. TCOT and OCC-UTS adopt new approaches completely different from 1PC or 2PC protocols. The other protocols conserve 2PC principles and try to optimize it to fit mobile environment requirements. The number of wireless messages of M2PC and TCOT proposed in [13] do not indicate the superiority of TCOT. However, this superiority is important when the overall performance (including total number of messages on wired and wireless links) is considered; this is explained in more details in [13]. Note that Table 1 summarizes the message complexity in the case of mobile clients as besides UCM and CO2PC all the protocols studied do not consider the case of mobile servers.

UCM does not indicate how to address the handoff situations. With TCOT, if a client moves from a cell to a new one, it sends the identity of its coordinator in the registration process at the new BS which then becomes the new coordinator. The new coordinator gets the state information from the old one to take over the remaining job. This is mandatory if the coordinator changes. Otherwise, no transfer of information is needed; only a classical registration of the MH is assumed. L-2PC and CO2PC use also a similar schema as TCOT in the case the latter uses dynamic coordinator transfer (that is the coordinator migrates as the MH does). However L-2PC clearly separates the handoff that physically transfers MH connection between the old and the new BSs from the ongoing commitment service. Thus it defines the notion of *proxy handoff* as a solution to mobility management at higher layer and CO2PC uses Agent handoff. M-2PC protocol also takes into account the mobility management at the higher level in the absence of underlying mechanisms to take over this task. An important conclusion that comes from Table 1 is that the handoff process do not increase the number of wireless messages when it is assumed that the needed information are transferred from the MH to the BS as a part of the registration process. Otherwise, if the MH does the transfer then one additional wireless message is needed at each handoff (see M-2PC). Concerning the power and other resources constraints, the general idea is to shift the workload to the fixed part of the network except when off-line execution is desired, then the MH must be well equipped.

Table 1. The protocols performances.

Protocol	No. of Wireless Messages to Commit a Transaction (Only the Client is Mobile)	Site of Transaction Execution	Mobility Management	Impact of Frequent Disconnection
M-2PC [22]	*2 U + h U + 1 D* *h: No. of hands off* *U: No of upstream msg.* *D: No of downstream msg.*	*FH*	*At the protocol level*	*Increase the number of transaction abortions*
M2PC [13]	*2 U + 1 D*	*MH &FH*		
OCC-UTS [22]	*1 U + r D* *r: No of invalidation reports*	*MH*		*Reception of invalidation reports*
TCOT [13]	*2 U + e U* *e: No. of timeout extensions*	*MH &FH*	*Registration level & protocol level*	*Increase the number of transaction abortions*
UCM [4]	*1 U + 1 D*	*MH &FH*		*Delay local transactions as resources are not released*
L-2PC [14]	*2 U+ 1 D*	*FH*	*At the protocol level*	*Results must be kept on behalf of MH*
CO2PC [25]	*1 U + 1 D*	*MH&FH*	*Registration level & protocol level*	*Results must be kept on behalf of MH*

UCM, OCC-UTS and CO2PC support off-line execution and minimize the risk of abortion at reconnection time. Frequent disconnections augment the abortion risks of TCOT and drop its performance. Because a mobile environment is failure prone and disconnections are frequent and unforeseeable and may last for long periods of time we claim that even though the timeout mechanism is well accepted in traditional distributed systems as a means to deal with the network or site failures detection; in mobile environment it is more reliable to not rely on any assumption about message and/or process scheduling delays. Thus, if necessary, a timeout mechanism can be acceptable only on the fixed part of a protocol execution.

Table 2. The protocols usability.

Protocol	Consistency approach	Mode of connection	Type of network	Legacy system integration
M-2PC [22]	*Strict*	*Continuous*	*May be hybrid*	*Not possible*
M2PC [13]	Strict	*Continuous*	*Mobile clients, IP like*	
OCC-UTS [22]	*Optimistic Strict*	*Intermittent*	*Mobile clients, IP like*	*Not possible*
TCOT [13]	*Optimistic Semantic*	*Continuous*	*Mobile clients GPRS-like*	*Not possible*
UCM [4]	*Pessimistic Strict*	*Continuous & Intermittent*	*May be hybrid mobile clients and servers (even lightweight)*	*Possible*
L-2PC [14]	*Strict*	*Continuous & Intermittent*	*Mobile clients IP-like hybrid*	*Possible*
CO2PC [25]	*Optimistic Semantic*	*Continuous & Intermittent*	*May be hybrid*	*Possible*

6. Conclusion

Mobile database processing will be quite diverse with probably many different transaction models and processing modes. Some will follow the traditional ACID requirements while others may not. Semantic based commit protocols eliminate the uncertainty period of transaction termination and the blocking effects [13]. Such protocols are proposed as alternatives to 2PC. The principal of these alternative protocols is to allow a participant to unilaterally commit a transaction and release the resources it holds. If the final decision is global abort, compensation is used to semantically undo the aborted transaction effects. These protocols do not provide strict atomicity as it was defined in traditional ACID transactions but they provide a semantic atomicity. The problem that arises from this is that compensation has limited applicability and in many applications it is necessary to ensure the strict atomicity. Needs of future applications will certainly vary and both ACP types of protocols strict or weak must be developed.

This paper addresses the transaction commitment problem in mobile environment. It describes the challenges faced in the new environment and surveys papers dealing with this topic. The principal techniques or mechanisms adopted to solve the problems are highlighted. This study represents the first step of our ongoing research which consists of the design and experiment of a commitment protocol that satisfies as many requirements of the mobile environment as possible.

References

[1] Abdallah M. and Pucheral P., "Validation Atomique: état de L'art et Perspective," *Revue Ingénierie des Systèmes d'Information (ISI),* vol. 5, no. 6, 1998.

[2] Al-Houmaily Y. and Chrysanthis P., "Two-Phase Commit in Gigabit-Networked Distributed Database," *in Proceedings of the 8th International Conference on Parallel and Distributed Computing Systems (PDCS)*, pp. 554-560, 1995.

[3] Bernstein P. A., Hadzilacos V., and Goodman N., *Concurrency Control and Recovery in Database Systems*, Addison Wesley, USA, 1987.

[4] Bobineau C., Pucheral P., and Abdallah M., "A Unilateral Commit Protocol for Mobile and Disconnected Computing," *in Proceedings of the 12th International Conference on Parallel and Distributed Computing Systems (PDCS)*, Las Vegas, USA, August 2000.

[5] Chrysanthis P. K., "Transaction Processing in Mobile Computing Environment," *in Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, Princeton, New Jersy, USA, pp. 77-83, 1993.

[6] Dirckze R., Gruenwald L., "A Pre-serialization Transaction Management Technique for Mobile-Multi-Databases," *Special Issue on Software Architecture for Mobile Applications*, vol. 5, no. 4, pp. 311 - 321, 2000.

[7] Dunham M., Hellal A., and Balakrishan S., "Mobile Computing and Databases: Anything New?," *in Proceedings of the ACM SIGMOD record*, vol. 24, no. 4, December 1995.

[8] Dunham M., Hellal A., and Balakrishan S., "A Mobile Transaction Model that Captures both the Data and Movement Behaviour," *Mobile and Networks Applications*, vol. 2, pp. 49-162, 1997.

[9] Gray J., *Notes on Database Operating Systems. Operating Systems: An Advanced Course*, LNCS, vol. 60, Springer Verlag, 1978.

[10] Gray J. and Reuter A., *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993.

[11] Imielinski T. and Badrinath B. R., "Mobile Wireless Computing," *Communication of the ACM*, vol. 37, no. 10, pp. 19-28, 1994.

[12] ISO, *Open System Interconnection- Distributed Transaction Processing (OSI-TP) Model*, ISO IS 100261, 1992.

[13] Kumar V., Dash K., Dunham M. H. and Seydim A. Y., "A Timeout-Based Mobile Transaction Commitment Protocol," *in Proceedings of ADBIS-DASEAA 2000, Advances in DB and Information Systems, In cooperation with ACM SIGMOD*, Prague, Czech Republic, September 2000.

[14] Lin Y. W. and Wu H. U., "Commit Protocol for Low-Powered Mobile Clients," *IEICE Transactions on Information and System*, vol. E82-D, no. 8, pp. 1167-1179, August 1999.

[15] Liu L., Agrawal D., and El Abbadi A., The Performance of Two-Phase Commit Protocols in the Presence of Site Failures, *Technical Report TRCS94-09*, University of California at Santa Barbara, April 1994.

[16] Lu Q. and Satyanaranyanan M., "Isolation-Only Transactions for Mobile," *Operating System Review*, pp. 81-87, 1994.

[17] Madria S. K. and Bhargava B. K., "A Transaction Model for Mobile Computing," *in Proceedings of the International Database Engineering and Application Symposium (IDEAS)*, 1998.

[18] Mazumbar S. and Chrysanthis P. C., "Achieving Consistency in Mobile Databases through Localization in PRO-MOTION," *in Proceedings of the DEXA International Workshop on Mobility in DB and Distributed System*, Italy, pp. 82-89, 1999.

[19] Mohan C., Lindsay B., and Obermarck R., "Transaction Management in the R*Distributed Data Base Management System," *ACM Transactions on Database Systems*, vol. 11, no. 4, pp. 378-396, 1986.

[20] Narasayya V. R., *Distributed Transactions in Mobile Computing System*, draft, submitted as part of the requirement for CSE552, March 1994.

[21] OMG, Object Management Group, *Object Transaction Service*, OMG Document 94.8.4, in OMG (Ed), 1994.

[22] Perron M. and Bai B., "Low Cost Commit Protocol for Mobile Computing Environments," available at: http://www.cs.Ualberta.ca, December 1999.

[23] Pitoura E. and Bhargava B., "Data Consistency in Intermittently Connected Distributed Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 896-915, 1999.

[24] Pitoura E. and Bhargava B., "Revising Transaction Concepts for Mobile Computing," *in Proceedings of the IEEE Workshop on Mobile Systems and Applications*, Santa Cruz, CA, 1994.

[25] Serrano-Alvarado P., Roncancio C., Adiba M., and Labbé C., "Adaptable Mobile Transactions and Environment Awareness," *in Proceedings 19th Journées Bases de Données Avancées*, BDA 2003, Lyon, France, October 2003.

[26] Stamos J. and Christian F., "A Low Cost Atomic Commit Protocol," *in Proceedings of the 9th Symposium on Reliable Distributed Systems*, 1990.

[27] Walborn G. D. and Chrysanthis P. K., "Promotion: Management of Mobile Transactions," *in Proceedings of The 11th ACM Annual Symposium on Applied Computing, Special Track on Database Technology*, Van Jose, CA, pp. 101-108, 1997.

[28] Weikum G. and Vossen G., *Transactional Information Systems Theory, Algorithms, and the Practice of Concurrency Control and Recovery,* Morgan Kaufmann, USA, 2002.

[29] X/Open, *CAE Specification, Distributed Transaction Processing: Reference Model*, X/Open Guide, version 3, G307, X/Open Company Limited, 1996.

Nadia Nouali obtained her engineering degree in computer science from Houari Boumediene University of Algiers and her magister degree from the Centre of Advanced Technologies of Algiers, Algeria. She has been a member of the scientific and research staff. Since 2001 she is the head of the Mobile Computing Department of the Research Centre in Scientific and Technical Information of Algeria (CERIST). Her research interests include internet architecture and protocols, wireless networks, distributed computing, mobile computing, ad hoc networks, mobile databases and transactions, and security.

Habiba Drias received her master degree in computer science from Case Western Reserve University, Cleveland OHIO, USA, in 1984, and the doctorate degree prepared at ParisVI University from Algiers USTHB University in 1993. She has directed the Computer Science Department of USTHB and then the laboratory of research in AI for many years. She has over fifty published papers in the domain of artificial intelligence, e-commerce, computational complexity, and the satisifiability problem. Currently, she is the principal of the Algerian National Institute of Computer Science.

Anne Doucet started as an assistant, then lecturer at the Paris XI-Orsay University. She is a professor at Paris VI-Pierre & Marie Curie University since 1994. She is the head of the database research team of the Computer Science Laboratory of Paris VI. Her research field is mainly in databases. Her first work concerned the object databases; she took part in particular in the design of the O2 Object DBMS. Then she worked on the coherency of object and distributed data bases. Her research interests include integration of heterogeneous and distributed data.