A Mobility-Aware Two-Phase Commit Protocol

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

Abstract: *The exploding activity in the telecommunication domain and the increasing emergence of portable devices are making mobile ubiquitous computing a reality. However, many challenging issues have to be faced before enabling users to take part in distributed computing while moving in an efficient and quasi-transparent manner. Many researches focus on revisiting the conventional distributed computing paradigms for use in the new environment. In this paper we propose to revisit the conventional implementation of the Two Phase Commit (2PC) protocol which is a fundamental asset of transactional technology for ensuring consistent effects of distributed transactions. We propose a new execution framework that provides an extension aware of the mobility of the hosts. The proposed Mobility-aware 2PC (M-2PC) protocol preserves the 2PC principle and the freedom of the mobile clients and servers while it minimizes the impact of unreliable wireless communication links.*

Keywords: *Mobile computing, mobile transactions, mobile 2PC protocol, disconnection, mobility.*

Received March 15, 2004; accepted October 30, 2004

1. Introduction

In distributed systems, an Atomic Commitment Protocol (ACP) is needed to terminate distributed transactions. A transaction is a set of operations that form a *logical unit of work*. The essential idea of a transaction is its *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, 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 data 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 and standardized mechanism dealing with the commitment problem is the two-Phase Commit (2PC) protocol [3, 12, 27, 20] that allows the involved parties to agree on a common decision about to commit or abort the transaction even in the presence of failures. Much has been written about this protocol and its variants (Presumed Commit (PrC) [18], Presumed Abort (PrA) [18], Early Prepare (EP) [23], and Implicit-Yes-Vote [2]) until recently [26] and motivated us to study the possibility of adapting it to mobile systems.

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 not 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 adapt the implementation of the 2PC protocol for mobile distributed transaction systems. Papers in mobile computing literature such as [13], argue that weak consistency is sufficient in mobile environment and propose optimistic approaches. However, there are many applications such as banking services or health care applications that can not tolerate working without strict atomicity. For example, a doctor at the bedside of his patient needs strict atomicity to modify the file of the patient located at the hospital database. Banking services issued from mobile devices can not work correctly without strict atomicity. Our mobile 2PC (Mobility-aware 2PC) protocol aims at providing the A and D properties despite the challenges of mobile wireless environment.

The remainder of this paper is organized as follows. Section 2 outlines the mobile environment challenging issues affecting the commitment paradigm. Section 3 discusses the strategies chosen to design our protocol and its correctness. Section 4 provides an overview of related research. Section 5 concludes the paper.

2. Challenges Faced by the 2PC Protocol in Mobile Environment

We adopt the system model depicted in Figure 1. 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 equipped 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 *hand-off*. The hand-off 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.

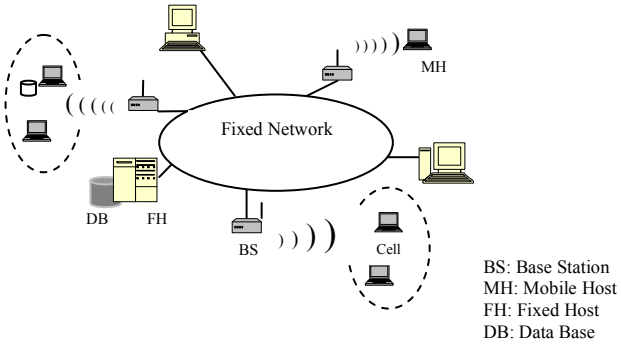

Figure 1. Mobile system architecture.

The first difference between a distributed system and a mobile system is that 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: Whereas in a distributed system FH are connected to the network through continuous high-bandwidth links, in a mobile system devices are connected via wireless links characterized by a very lower bandwidth and big latency. In the fixed system the hosts are rarely disconnected either explicitly or implicitly because of a failure. The MHs 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.

2.1. Traditional 2PC Protocol

This section outlines the execution of the well known 2PC protocol (see Figure 2). 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: A voting phase and a decision phase. In the first phase, the site where the transaction was originally initiated (generally called the *coordinator*) sends messages to all the sites involved *(participants)* in the transaction, asking them to prepare to commit the transaction *(prepare message)*. If, for any reason, including 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 that are used for 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.

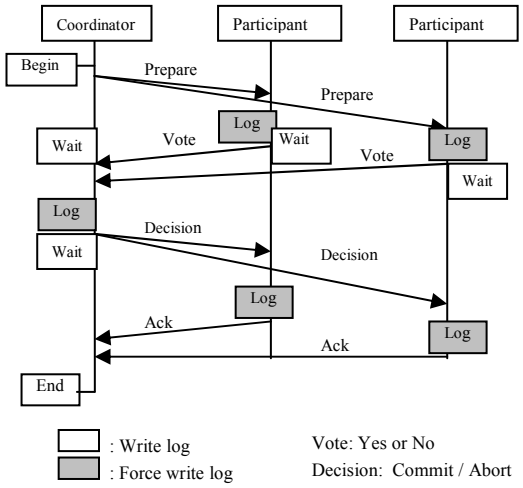

Figure 2. The 2PC protocol.

2.2. Executing 2PC Protocol in Mobile Environment

In this section, we summarize the principal issues intrinsic to mobile computing that affect transaction commitment:

1. The characteristics of *wireless communication* are an important factor to be taken into account. The 2PC protocol requires two message rounds and *4n* messages, where *n* is the number of participants. This is not sustainable with regard to the expensiveness, low and variable bandwidth and the latency of wireless communication. Thus, to minimize the use of wireless communication it is necessary to make the role of the MHs as minimal as possible. This principally means that the coordinator must not be executed on the mobile host. Thus the BS [19] is chosen as the coordinator host. Then, the number of messages exchanged over the wireless network, especially in up-stream, is significantly reduced. As the coordinator is on the fixed part of the network logs will be kept more safely. Communication between participants and the coordinator do not transit via a BS.
2. The *scarcity of processing and energy resources* of the MHs means that resources conservation is an important issue to consider. As we saw before, the role of coordinator necessitates some transaction processing and storage capabilities that may not be available in certain MHs. This can also be achieved by minimising the role of MH and its messages exchange and shifting the workload to the fixed part of the network [4, 7, 19].
3. The *mobility* of hosts makes them subject to unpredictable *hands-off*. MHs may connect from any BS, at any time and may also move while staying connected. In general, this issue is treated in papers dealing with the overall transaction management problem [7]. For accuracy purpose, we focus on the mobility during the commitment phase only. When a MH moves from one cell to another, it will not be able to communicate with the BS (the coordinator location) of the cell it has just left. This may impact more severely failure or crash recovery situations where the coordinator and the client need to get in touch to finally terminate the transaction. There are two possible strategies:
   - A static coordination in which the coordinator resides at the same BS during all the protocol execution. It is necessary to make the coordinator know about the MH new location each time it moves from a cell to a new one. The distance between the coordinator and the MH may increase the protocol latency and augment its vulnerability window[1].
   - A migrating coordination in which the coordinator moves as the MH does, i. e., the coordinator may always reside at the same BS as the MH does. The drawback of this strategy is the cost overhead that can be introduced by eventually frequent migration as there will be as many coordinator hands off as network hands off. The handoff of coordination consists of the transfer of state information from the old coordinator to the new coordinator. The latter has to inform all the participants about this change. This process introduces message overhead.
4. A MH may disconnect 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 may lead to 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 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. Contrary to the traditional 2PC, a protocol must not account on MHs to be continuously available to participate in the transaction commitment.

[1] Vulnerability window is the period of time between the voting phase and the decision phase.

3. M-2PC Protocol

The objective of our ACP protocol is to globally commit a mobile transaction T_m which is being executed over more than one host. In our design we try to answer each of the requirements listed above. The hardware architecture assumed is depicted in Figure 1. We also assume that a transaction T_m is issued at an MH that we call *Home-MH* and the BS to which it is attached is called *Home-BS*. While the MH moves from a cell to another it attaches to a new BS that we call *Current-BS*. At commit time a *Commit-request* is issued from the *Home-MH*, thus its *Current-BS* (it may be the Home-BS) becomes the *Commit-BS*. The M-2PC protocol may either terminate in the same cell or in a new cell covered by a new BS. The software architecture reflects the different roles that each entity participating in the M-2PC protocol must play in such a way to adapt in a flexible manner to the underlying network configuration (Figure 3). Similarly to the 2PC protocol, there are three important roles to represent:

The *transaction initiator* which is the MH launching T_m, the *participants* which are the processing entities of the transaction operations and the *coordinator* which coordinates the consistent termination of the transaction. As depicted in Figure 3, the transaction initiator is called a *client,* the servers are called *participants* and the *Commit-BS* is the *Coordinator*. Many execution scenarios could be considered according to the underlying network infrastructure. The scenario depicted in Figure 3 supposes that only the client is mobile whereas the servers are fixed.

3.1. The Case of Mobile Client and Fixed Servers

The strategy we choose is to split the duties of M-2PC protocol into two tasks: The first one maintains the same schema on the fixed part of the network as in traditional 2PC; the second one adjusts the schema to manage the mobile wireless part. In other words the coordination of FHs decision must be conducted as it is in the traditional 2PC protocol, thus a *coordinator* must reside in the fixed part of the network to be directly reachable by the fixed participants. The coordinator must also be reachable by the client residing on the MH, thus the best choice is to make it reside on the *Current-BS* (it may be the Home-BS if the MH never moves). The *coordinator* executes on the first BS that receives the Commit-request; the *Commit-BS*. This means that the coordinator is likely to be located as close as possible to the client. The *coordinator* executes on the same *Commit-BS* even if the MH changes cell during the commitment process. Indeed, as we treat mobility only during the limited period of ACP execution, migrating control as frequently as the MH moves appears to be useless for two reasons: Either, MH moves relatively slowly thus the probability of the commitment protocol terminating at the same cell is high. Or, it is fast moving then a frequent migration of the control may increase the protocol latency and thus its vulnerability.

The *client* sends the request for commit to the coordinator along with its logs. Afterwards, the client can disconnect. The *coordinator* sends vote messages to all *participants* and decides on whether to commit or abort according to the traditional 2PC principal. After receiving the acknowledgements the coordinator informs the client, which may be in another cell, about the result. The coordinator waits for the client acknowledgement before forgetting about the transaction (releasing resources).

During execution of M-2PC, the client can cross boundaries between cells and register in a new BS. In contrary to solutions suggesting to handoff the control, M-2PC does not. Recall that the *coordinator* is launched dynamically on the *Commit-BS* (the first receiving the commit order) and stays their during all the commit protocol execution. M-2PC protocol requires only the coordinator permanently knows about the client current location in order to forward the results to it. The solution adopted is to make the client contact the *coordinator* (on the *Commit-BS*) to tell it about its new address. Thus an uplink wireless message is required. This solution offers a mean to deal with mobility at the application layer and embeds the mobility mechanism in the protocol. This is adequate as actually the underlying networks do not still provide adequate mobility management.

It is clear that the client is responsible to record identity and location information of the coordinator for use when it registers at a new BS. However, if no precaution is taken this information could be lost as a consequence of a sudden disconnection or failure. So, to avoid these problems, the client must force-write the identity and location information of the coordinator (*Commit-BS*) just before sending the *Commit-request*. Thus, if the *Commit request* is correctly received by the BS (which then becomes the coordinator), one can be sure that the force-writing has taken place without any problem. We choose a static coordinator, so only one force-write is needed to record the *coordinator* information during the entire execution of ACP.

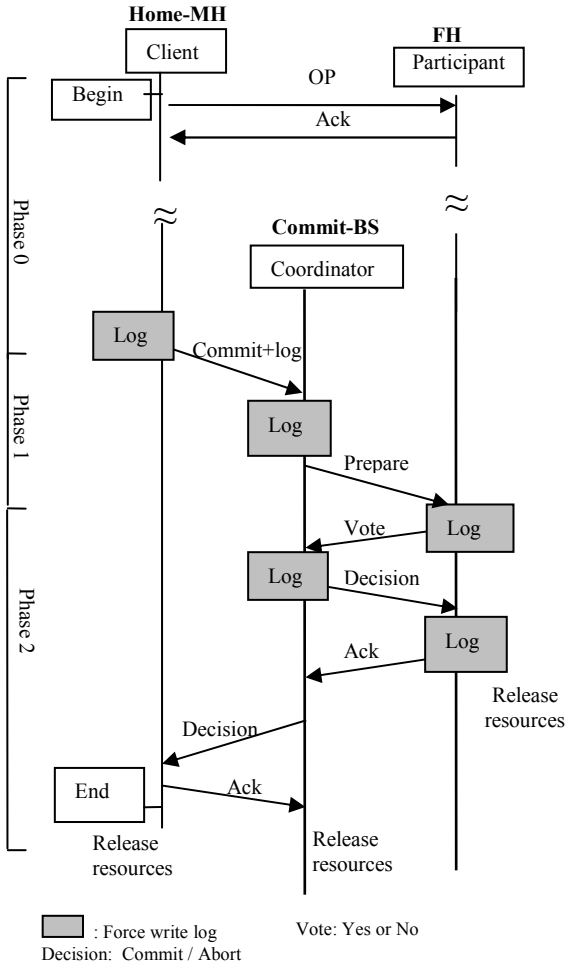

Figure 3. The M-2PC protocol.

3.2. The Case of Mobile Client and Mobile Servers

Figure 4 depicts a scenario where at least one server or *participant* (other than the transaction initiator) is mobile. That is, T_m accesses, for example, data located on a MH. Assume, for example, a scenario where a researcher meets other researchers at a conference and needs to agree on a rendezvous with them. In this application, the researchers' respective agendas have to be synchronized. Thus, in this application, all the participants are mobile. The M-2PC protocol behaves similarly to the case of fixed servers. The idea is to have in the *mobile participant* side a scheme similar to that of the client side.

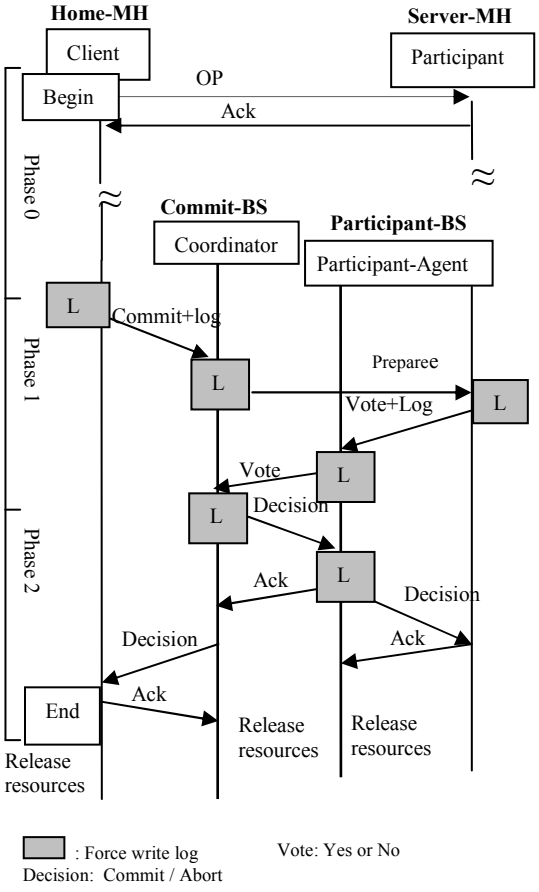

Figure 4. The M-2PC protocol with mobile servers.

A representation agent, we call it *participant-agent*, will work on behalf of the mobile server which is free to disconnect from the moment it delegates its commitment duties to its representation agent: The mobile participant receives the request to vote (prepare message), it sends its vote along with its log to the *participant-agent*, and may disconnect. The *participant-agent* forwards the vote to the coordinator and waits for the final decision. It is responsible of transmitting the result to the participant at reconnection time and also of keeping logs and eventually recovering in the case of failure. The *participant* is free to move to another cell. Similarly to the mobile client, the *participant* MH (or *mobile participant*) informs its *participant-agent* about its new location when it registers to a new BS. Again, the workload is shifted to the fixed part of the network thus preserving processing power and communication resources and minimizing traffic cost over the wireless links.

3.3. Correctness of M-2PC Protocol

The M-2PC uses the same schema as 2PC protocol which correctness has been largely proven [3, 10, 26] and does not make any assumption about the consistency and local concurrency control mechanisms. The correctness in the case of disconnection or mobility is straightforward. If neither disconnection nor handoff occur, the protocol execute normally in the same cell; its *Current-Bs* does not change. The *coordinator* only forwards its decision to the *mobile client* and waits for an ACK. In the case of a sudden disconnection or failure, we assume that the MH recovers in a finite delay. During this disconnection, the coordinator/participant-agent keeps the results on behalf of the MH until it reconnects to the network. Then, it forwards it the results and waits for the ACK. Thus, the MH is free to disconnect without affecting the protocol execution. In the case of a handoff without disconnection, as soon as the MH reconnects to the network and registers at its new BS, it contacts the *coordinator/participant-agent* to inform it about the new location. So the MH is free to move during the commit protocol execution. The most complicated scenario is when the MH hands off while disconnected, i. e., it disconnects from the *Current-BS* and reconnects to a new BS. At reconnection time, it registers at the new BS which becomes its *Current-BS* and then it contacts the *coordinator/participant-agent* at the *Commit*-BS. The protocol goes on normally. The mobile client/server does not directly participate in the M-2PC protocol execution. In fact, after sending the commit request/vote along with the needed logs, the MH is free to disconnect. The coordinator/participant-agent does the work and plays exactly the 2PC principle and is in charge of communicating the result to the mobile client/server in an asynchronous manner. Also, the movement of the MH does not affect the atomicity of the transaction. The only thing to take care of in the case of handoff is to make the *coordinator* knows of the *mobile client* (respectively, the *participant-agent* of the *mobile server*) location. This process introduces only one force-write at the MH side to log the *coordinator/participant-agent* information and one message from the MH. It is important to recall that all necessary logs are stored in stable storage to guaranty permanence and fault tolerance. If we assume that any failed component would recover and reconnect to the system, we claim that all or nothing property is guaranteed by the M-2PC protocol. The advantage of keeping the protocol as in the traditional

distributed systems makes it possible to conduct the ACP in hybrid infrastructure with mobile and fixed hosts without worrying about local concurrency or recovery mechanisms used by the different systems. The only requirement is that the rules of the commit protocol itself are followed by all parties.

4. Related Work

Much literature studied 2PC protocol [1, 9, 10, 14] and its variants in the context of distributed systems. The abundant literature on mobile computing generally focuses on transaction models [5, 6, 7, 8, 11, 15, 16, 17, 19, 22, 23, 25]. In this paper, we focus on the commitment issue.

We classify the mobile computing solution for ACP into two main approaches. One approach argues that the strict atomicity is not adequate and rejects the 2PC mechanism. The focus is on strictly designing new protocols to meet the mobile environment requirements [13, 21]. These protocols follow an optimistic approach and sometimes tolerate weak or semantic atomicity by admitting the concept of *compensation* as in [13]. The other approach tries to adapt the 2PC or its variants to mobile environment. For example, [4] adapts a variant of 1PC (one-phase) ACP. Perron and Baochun [21] propose a new timestamp based protocol called Optimistic Concurrency Control with Update Time Tamp (OCC-UTS). Kum and *et al.* [13] propose a new timeout based commitment protocol called Timeout-based mobile Transaction Commitment Protocol (TCOT). These papers compared their solutions to slightly modified 2PC protocol version. However, they do not further improve 2PC protocol for mobile computing but rather focus on proposing a new way of committing a mobile transaction.

The basic idea of OCC-UTS is to verify that a transaction is serializable before deciding if it should be committed or aborted [21]. The protocol assumes the existence of a local cache at the client 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 determine if a transaction is serializable by comparing 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. A request to commit message is sent to the server after positive validation. This protocol is suitable for PDAs or portable computers relatively well equipped in order to provide local application execution, generally offline execution. However, OOC-UTS may cause difficulties in data base servers with a heavy load as the MHs will be waiting for a long time before their messages are processed and may timeout.

The TCOT protocol is based on a timeout approach used to reach a final transaction termination decision (commit/abort) in a message oriented system [13]. The timeout mechanism is used to minimize the impact of slow and unreliable wireless link. The author shows that their protocol commits transactions in mobile database systems with minimum number of uplink (client to server direction) by allowing to every processing unit participating in the transaction to have independent decision making capability. Contrary to [21], 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.

[4] propose a new ACP called Unilateral Commit for Mobile (UCM). Their goals 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 such as smart cards) are accepted and UCM uses a single message round thereby saving wireless communications.

Table 1 summarizes the protocol characteristics. It is clear that UCM has the best message complexity in terms of wireless messages, but this is obtained at the price of strict assumptions about the local concurrency and recovery mechanisms which limits its usability in arbitrary heterogeneous systems. An important conclusion that comes from table 1 is that the handoff process does not increase the number of wireless messages as the transfer of MH state between the BS takes place over the wired links. However, M-2PC can be used in a network where no location management is put in place. In such situation, the MH initiates itself the handoff process by sending an uplink message to notify its new BS each time it changes cell. Thus an additional wireless message is required at each handoff. Table 1 also indicates the application type to which the protocol could be suitable. For example, if a continuous connectivity is required it is clear that this can not suit offline applications.

TCOT protocol considers the handoff effect from the point of view of migrating (dynamic) or not (static) the control of the transaction (coordinator). M-2PC protocol considers the mobility problem as a matter of maintaining the communication between the participating entities in the protocol in terms of message exchange. Thus, M-2PC protocol conserves the traditional 2PC principle and solves the wireless and mobile new problems. The handoff of control may

or may not occur according to the application or service (in this paper the commitment) semantics and requirements. The principle idea is to deal with the handoff issue at the application layer so as to adapt in a flexible manner to network infrastructure that do not provide mobility management.

Table 1. Comparison of the protocols.

Protocol	No. of Wireless Messages to Commit a Transaction (only the Client is Mobile)	Site of Transaction Execution	Mode of Connection	Type of Coordination
M-2PC	2 U + 1 D U: uplink D: downlink	MH FH	Continuous Intermittent	Static but dynamic allowed
OCC-UTS [21][23]	1 U + r D r : No of invalidation reports	MH	Intermittent	Not applicable
TCOT [13]	Low layer connectivity handoff 2 U + e U e : No. of timeout extensions	MH FH	Continuous	Static or dynamic
UCM [4]	1 U + 1 D U: uplink D: downlink	MH FH	Continuous Intermittent	Static or dynamic

5. Conclusion

The M-2PC protocol does not modify the 2PC basics. So, it keeps its advantages and its drawbacks too. Any improvements made on 2PC protocol could easily be brought into the M-2PC protocol. We claim that our architecture is generic in the sense that it can fit to all 2PC variants. No assumption is made about local concurrency and consistency mechanism, thus M-2PC can co-exist with traditional ACP such as the traditional 2PC nodes that do not wish to implement mobility.

References

[1] Abdallah M. and Pucheral P., "Validation Atomique: état de l'art et Perspectives," *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)*, 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, 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. and Gruenwald L., "A Pre-serialization Transaction Management Technique for Mobile-Multi-Databases," *Special Issue on Software Architecture for Mobile Applications*, vol. 5, no. 4, 2000.

[7] 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. 149-162, 1997.

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

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

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

[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," *ADBIS-DASEAA 2000, Advances in DB and Information Systems, in Cooperation with ACM SIGMOD*, Prague, Czech Republic, 2000.

[14] 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*, Santa Barbara, Department of Computer Science, University of California at Santa Barbara, April 1994.

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

[16] Madria S. K. and Bhargava B. K., "A Transaction Model for Mobile Computing," *International Database Engineering and Application Symposium (IDEAS 98)*, 1998.

[17] 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 Distribution System, Italy, pp.82-89, September 1999.

[18] Mohan C., Lindsay B., and Obermarck R., "Transaction Management in the R*Distributed Database Management System," *ACM Transactions on Database Systems*, vol. 11, no. 4, 1986.

[19] Narasayya V. R., "Distributed Transactions in Mobile Computing System," *Technical Report*, University of Washington, 1994.

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

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

[22] 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.

[23] 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.

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

[25] 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.

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

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

Nadia Nouali obtained her engineer degree in computer science from Houari Boumediene University of Algiers and her master degree from the Centre of Advanced Technologies of Algiers, Algeria. She has been a member of the scientific and research staff since 2001 and 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 has received the master degree in computer science from Case Western Reserve University, Cleveland, OHIO, USA in 1984 and the doctorate degree prepared at Paris6 University from Algiers USTHB University in 1993. She has directed the Computer Science Department of USTHB and then the Laboratory of Research in Artificial Intelligence 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 has been 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 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 databases. Her recent research focuses on the integration of heterogeneous and distributed data.