A Two-Phase Commit Protocol for Mobile Wireless Environment

Nadia Nouali[1,2,3], Anne Doucet[2], Habiba Drias[3]

[1]Centre de Recherche sur l'Information Scientifique et Technique (CERIST)
Division de l'Informatique Mobile
Rue des 3 Frères Aissou, Ben Aknoun, 16030, Algiers, Algeria.
Tel: 213 21 9120 25, Fax: 213 21 91 21 26

nnouali@{cerist.dz, wissal.dz}

[2]Université Pierre et Marie Curie
Laboratoire LIP 6, France

Anne.Doucet@lip6.fr

[3]Université des Sciences et Technologies Houari Boumediene (USTHB)
Faculté du Génie Electrique, Algeria

drias@wissal.dz

Abstract

The challenges of wireless and mobile computing environments have attracted the attention of researchers to revisit the conventional implementation of distributed computing paradigms. 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 the consistent commitment of distributed transactions. We propose a new execution framework providing an efficient extension that is aware of the mobility. The proposed M-2PC (Mobile 2PC) protocol preserves the 2PC principle and the freedom of the mobile clients and servers while it minimizes the impact of unreliable wireless communications.

Keywords: Two-phase commit protocol, Mobile transaction processing, Handoff, Disconnection.

1 Introduction

The exploding activity in the telecommunication domain and the increasing emergence of portable devices are making mobile and 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.

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 *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 (Bernstein, Hadzilacos and Goodman 1997). The most commonly used and standardized mechanism dealing with the commitment problem is the *two-phase commit* (2PC) protocol (Bernstein, Hadzilacos and Goodman 1997, ISO 1992, X/Open 1996, Object Management Group 1994] 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) (Mohan, Lindsay and Obermarck 1986), presumed abort (PrA) (Mohan, Lindsay and Obermarck 1986), early prepare (EP) (Stamos and Christian 1990), Implicit-Yes-Vote (Al-Houmaily and Chrysanthis 1995) until recently (Weikum and Vossen 2002) 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, and exchanging messages over wired networks with a permanently available bandwidth. These assumptions are not valid in the new environment where a typical architecture for modern Information System (Bolchini, Schreiber and Tanca 2004) includes portable and small devices equipped with more or less limited 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 Kumar, Dash, Dunham and Seydim (2000), 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 or e-commerce transactions issued from mobile devices can not work correctly without strict atomicity. Our mobile 2PC **(Mobile 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 and evaluation. Section 4 provides an overview of related research. Section 5 concludes the paper.

2 The challenges faced by the 2PC in mobile environment

We adopt the system model, depicted in Fig. 1, consisting of two main parts: the fixed part of the network (e.g. Internet) and a wireless access network such as a IEEE 802.14. The global architecture consists of two distinct sets of entities: mobile hosts and fixed hosts. A *mobile host* (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 *fixed hosts* (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.

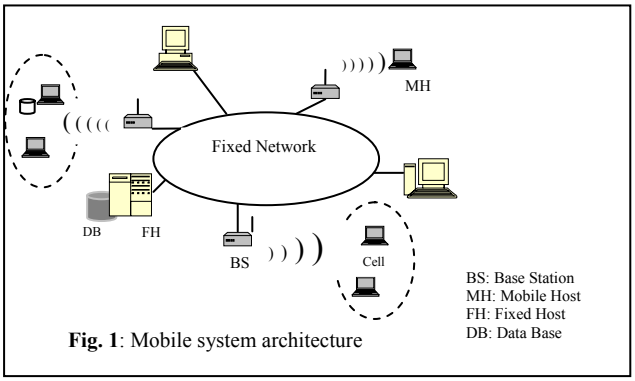

Fig. 1: Mobile system architecture

2.1 The traditional 2PC protocol

This section outlines the execution of the well known 2PC protocol (see fig. 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 roll back 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.

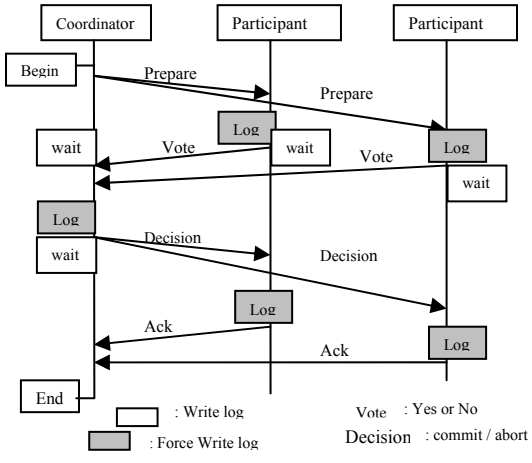

Fig. 2. The 2PC protocol

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.

2.2 Executing 2PC protocol in mobile environment

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

i- Wireless communications. 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. 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 (Narasayya 1994) 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.

ii- The scarcity of processing and energy resources of the MHs. The role of coordinator necessitates some transaction processing and storage capabilities that may not be available in certain MHs. Resources conservation 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 (Narasayya 1994), Dunham, Hellal and Balakrishan 1997, 4].

iii- The ***mobility*** of hosts make 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 (Dunham, Hellal and Balakrishan 1997). For accuracy purpose, we treat the mobility issue in the context of commitment only. When a MH moves from one cell to another, it will not be able to communicate with the BS (where the coordinator is homed). This may impact more severely failure or crash recovery situations where the coordinator and the client need to get in touch to 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 is attached to. 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 also to inform all the participants about this change. This process introduces message overhead and may increase the protocol latency.

iv-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. If the traditional 2PC is executed in mobile environment, disconnections will increase the number of, may be wrong, abortion decisions of transaction because if a FH tries to communicate with it a disconnected MH this will cause a failure. As frequent are disconnections, as transaction abortions are. 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.

3 The 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 fig. 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 cell 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 to adapt in a flexible manner to the underlying network configuration (Fig. 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 a *coordinator* which coordinates the consistent termination of the transaction. As depicted in fig. 3, the transaction initiator

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

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 fig. 3 assumes 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 address 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.

To handle disconnections we make the client delegate its commit duties to the coordinator which is always available during protocol execution. 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 the *coordinator* (on the *commit-BS*) to tell it about its new address. Thus an uplink wireless message is required. This solution offers a way 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 mitigate the unforeseeable breakdowns, 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.

3.2 The case of mobile client and mobile servers

Fig. 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 situation where a researcher meets other researchers at a conference and needs to agree on a rendezvous with other researchers. In this application, the researchers' respective agendas have to be synchronized. Thus, all the participants are mobile.

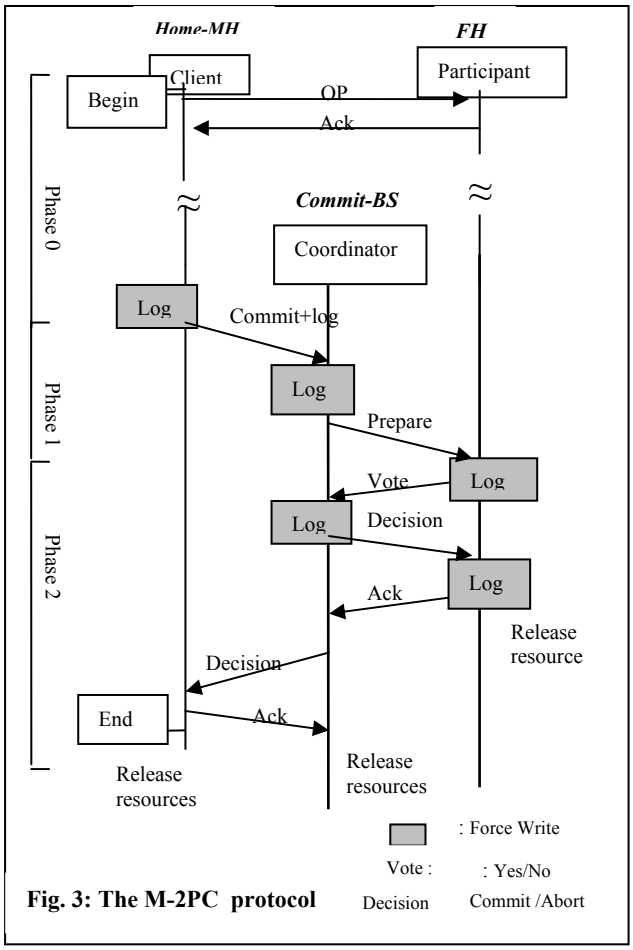

Fig. 3: The M-2PC protocol

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. 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 *participant-agent* 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 during the protocol execution. When it registers to a new BS, the *participant* MH (or *mobile participant*) informs its *participant-agent* about its new location. 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 (Bernstein, Hadzilacos and Goodman 1997, Weikum and Vossen 2002, Gray J 1993) 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* know of the *mobile client* (respectively, the *participant-agent* of the *mobile server*) location. This is done in the case of handoff. This process introduces only

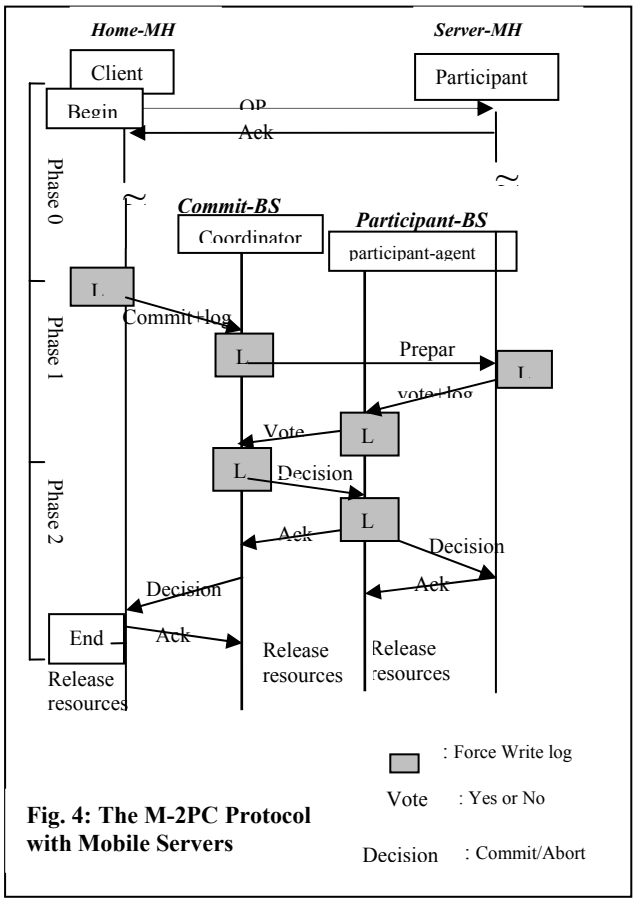

Fig. 4: The M-2PC Protocol with Mobile Servers

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 guarantee 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 (Bolchini, Schreiber and Tanca 2004) 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.

3.4. Evaluation of the mobility management technique

Our protocol maintains the principle of the traditional 2PC. Thus, the new techniques M-2PC introduces are the disconnection handling and the mobility management. In this section we analyse the influence of the mobility management technique used as it is embedded within the protocol and this is a new approach. In the literature, mobility management is rather considered at lower layers such as Mobile IP. The basic technique to tackle mobility with M-2PC is to notify by an I-AM-HERE the coordinator/participant-agent (C/P) directly about the MH new location. This location update entails the one-way delay after the MH recognizes its IP address change. The timing diagram of such hand-off for an MH in an IEEE

802.11 network is shown in Fig. 5 in which the identification of a foreign network and then acquisition of DHCP address constitute a bulk of the hand-off delay. Once this is done, the main component of the handoff delay becomes the one-way transmission delay of the I-AM-HERE message from the MH to the C/P.

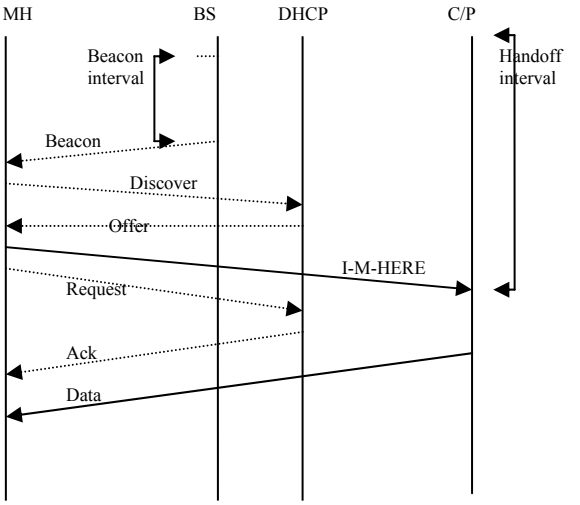

Fig. 5 Timing for MD-2PC terminal handoff

We analyze the ACP terminal hand-off as depicted in Fig.5. When an MH moves under a new base station (BS), the delay due to the foreign network identification by listening to the beacons transmitted from the new BS and the following DHCP address acquisition can be minimized using link layer mechanisms. Packets are sent to the new MH location as soon as the I-M-HERE message reaches the coordinator/participant-agent and is processed. So the effective hand-off delay is the one-way transmission time of the I-M-HERE message to the coordinator/participant-agent and its subsequent processing time at the latter. However, since our ACP is an application layer protocol, its messages may not be served with highest priority.

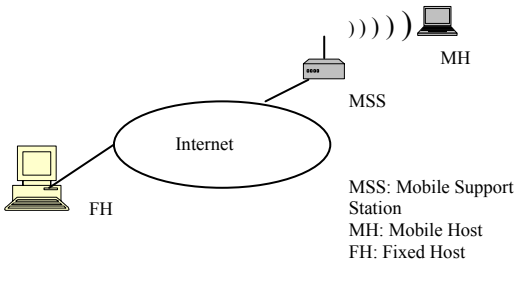

Fig. 6: Involved components in the handoff delay

Thus when an MH sends I-M-HERE message to the C/P, the C/P's operating system may be busy with its own time critical operating system functionality and defer the processing of the I-M-HERE request. Moreover, the unreliable wireless access network introduces its own delay due to the additional error recovery protocol layers. The main components introducing the delay in the mobility management procedure of M-2PC are shown in Fig.6. The MH generates M-2PC messages to update its current location and the messages are sent to the nearest base stations for transmission to the coordinator/participant-agent through the Internet. Once the C/P receives the I-M-HERE message from the MH, after it moves, the C/P gets the updated location information and henceforth sends data to the right location. So the hand-off delay is typically the time required for the I-M-HERE message to reach its destination. Major delays in this hand-off procedure occur at (i) the MH, (ii) the wireless radio link between the MH and the BS, (iii) the Internet, and (iv) the C/P.

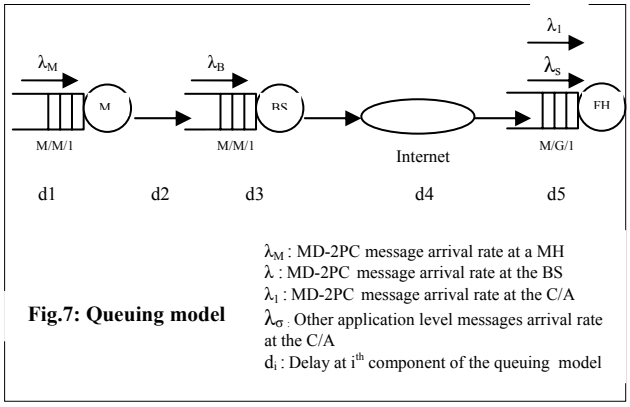

Fig.7: Queuing model

Except from Internet, each of these delay components is modelled as a queue. The MH and the BS are modelled by an M/M/1 queue and the C/P by a non-pre-emptive priority-based M/G/1queue. The queuing model for the delay incurred in hand-off is shown in Fig.7.

The hand-off delay *(D_h)= d1+d2+d3+d4+d5*, where *d1* is the delay at the MH, *d2* is the transmission delay over the wireless link, *d3* ' is the queuing delay in the BS, *d4* is the Internet transmission delay (constant), and *d5* is the queuing delay in the C/P. From the queuing theory (Kleinrock 1975) the estimation of the delay parameters is:

$$d_1 = \frac{1}{\mu - \lambda_M} \text{ (1); } d_3 = \frac{1}{\mu - \lambda_B} \text{ (2); where } \lambda_M$$

and λ_B are the M-2PC message arrival rates at the MH and the BS respectively, μ is the processing rate for each M-2PC message in the MH and the BS. We consider that there are many MHs under a single BS, so $\lambda_M \leq \lambda_B$ and λ_M is a fraction of λ_B.

$$d_5 = \frac{\frac{1}{\mu_C}(1-\rho_1-\rho_C)+R}{(1-\rho_1)(1-\rho_1-\rho_C)} \text{ (4) where } R = \frac{\lambda_1 \overline{\chi}_1^{\,2} + \lambda_C \overline{\chi}_C^{\,2}}{2};$$

of μ_C and μ_1 are the processing rates at C/P for M-2PC messages and other messages respectively. $\overline{\chi}_1^{\,2}$ and $\overline{\chi}_C^{\,2}$ are the second moments of μ_1 and μ_C. Only higher priority messages are considered in (4) as only those messages may impact the processing delay of M-2PC messages.

ρ_C is the load at the C/P for M-2PC messages; ρ_1 is the load at the C/P for other messages; λ_1 is the arrival rate at the C/P for messages other than M-2PC.

The mean and variance are needed to derive the second moment $\overline{\chi}^2$. To calculate d_5 the standard deviation of the processing delay at the C/P is assumed to be 5% of the mean.

$\overline{\chi}_1^2 = E[\overline{\chi}_1^2]$ and $\overline{\chi}_C^2 = E[\overline{\chi}_C^2]$. Given the respective variances σ_1^2 and σ_s^2 then $E[\overline{\chi}_1^2] = \sigma_1^2 + (E[\chi_1])^2$ and $E[\overline{\chi}_s^2] = \sigma_s^2 + (E[\chi_s])^2$. We substitute $E[\chi_1]$ and $E[\chi_s]$ by μ_1 and μ_S and the values for the variances, thus $R = 0.501[\rho_1^2 + \rho_s^2]$.

For the numerical results, we assumed $\mu_C = \mu = 4*10^{-4}$ sec and $\lambda_M = 0{,}1\ \lambda_B$; $\rho_C\ \lambda_B / \mu$ ($\lambda_B < \mu$); $\rho_1 = 0.7$, $N_m = 10$.

The Internet transmission delay is assumed to be a constant, equal to typical worst case as it is difficult to standardize the heterogeneous transmission paths of Internet and compute the transmission delay which depends on the number of routers and the type of links in the path of message transmission (Eyers and Schulzrinne 2000). The Internet transmission delay is a constant I= 200msec (3).

The component delay d_2 represents the delay incurred by the wireless links. The model we used for TCP packet transmission over wireless links is suggested in (Das, Lee, Basu, Kakani, and Sen 2002) where the average delay for successfully transmitting a TCP segment with no more than N_m retransmission trials is given by:

$$d_2 = (k-1)\tau + \frac{D}{(1-q^{N_m})(1-2q)} + \frac{1-q}{1-q^{N_m}} D\left[\frac{q^{N_m}}{1-q} - \frac{2^{N_m+1}}{1-2q}\right]$$

(4) where k represents the number of frames of the TCP segment transferred over the wireless link, N_m is the number of TCP retransmissions before success, τ is the inter-frame time, D is the end-to-end frame propagation delay over the wireless channel, $q=1-(1-p)^k$ is the rate of packet loss and p is the probability of a frame being in error in the air link.

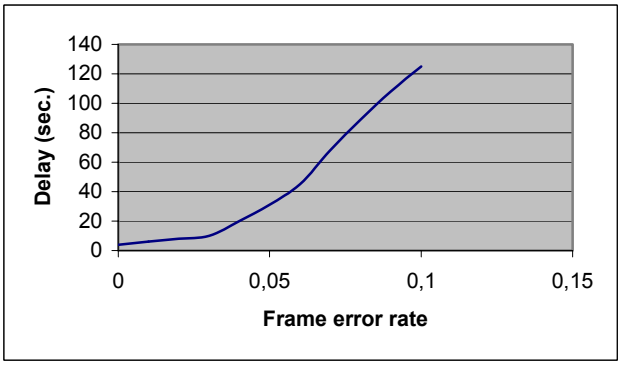

Fig. 8: Delay with arrival rate λ=500

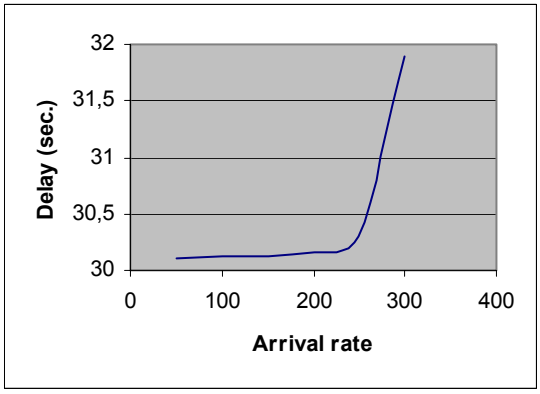

Fig. 9: Delay with frame error rate=0.05

Typical values reported in Das, Lee, Basu, Kakani and Sen (2002) are D=100 and τ=20 ms.

If we assume that one packet suffices to carry a TCP segment and that the frame duration is 20 msec, therefore, the frame of a 9.6 kps channel contains 24 bytes. If we also assume the M-2PC message size is 500 bytes then there are 21 (500/24) frames in the M-2PC message.

Fig. 8 shows the impact of the frame error rate on the handoff delay. The message arrival rate at the BS is fixed to 500. Fig. 9 shows the impact of the message arrival rate for a fixed frame error rate (FER =0.05). It is clear that the handoff delay increases with increasing frame error rate. This is due to the error recovery of TCP. Also, the handoff delay increases exponentially as the message arrival rate increases, i.e; as the processing rate at the different components approaches the message arrival rate. The hand-off delay component due to the processing of M-2PC messages at the C/P is negligible as compared to that incurred due to the wireless transmission of the messages. In both cases the wireless transmission delay is the major contribution to the total handoff delay (more than 90%). This indicates that the major factor in the handoff delay is induced by the unreliability of the wireless communications. Thus, even if application layer solution for supporting mobility may seem to be an attractive option, more investigation is needed to make them suitable for delay-sensitive applications.

4 Related work

Much literature studied 2PC protocol (Abdallah M. and Pucheral P. 1998, (Gray 1993, Gray, Reuter 1993, Liu, Agrawal and El Abbadi 1994) and its variants in context of distributed systems. However, similar studies in the mobile environment do not exist. The abundant literature on mobile computing generally focuses on transaction models (Chrysanthis 1993, Imielinski and Badrinath 1994, Pitoura and Bhargava 1994, Lu and Satyanaranyanan 1994, Narasayya 1994), Dunham, Hellal and Balakrishan 1995, Dunham, Hellal and Balakrishan 1997, Walborn and Chrysanthis 1997, Madria and Bhargava 1998, Pitoura and Bhargava 1999, Mazumbar and Chrysanthis 1999, Dirckze and Gruenwald 2000). In this paper, we focus on transaction commitment. 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. New protocols are designed to meet the mobile environment requirements (Perron and Bai 1999, Kumar, Dash, Dunham and Seydim 2000). These protocols follow an optimistic approach and sometimes tolerate weak or semantic atomicity by admitting the concept of *compensation* as in Kumar, Dash, Dunham and Seydim (2000). The other approach tries to adapt the 2PC or its variants to mobile environment. For example, (Bobineau, Pucheral and Abdallah 2000) adapts a variant of 1PC (one-phase) ACP. Perron and Baochun (Perron and Bai 1999) propose a new timestamp based protocol called OCC-UTS *(Optimistic Concurrency Control with Update Time Stamp)*. Kum and al. (Kumar, Dash, Dunham and Seydim 2000) propose a new timeout based commitment protocol called TCOT *(Timeout-based mobile Transaction Commitment Protocol)*.

The basic idea of OCC-UTS *(Optimistic Concurrency Control with Update Time Stamp)* is to verify that a transaction is serializable before deciding if it should be committed or aborted (Perron and Bai 1999). 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 *(Timeout-based mobile Transaction Commitment Protocol*) is based on a timeout approach used to reach a final transaction termination decision (commit/abort) in a message oriented system (Kumar, Dash, Dunham and Seydim 2000). The author shows that their protocol commits transactions in mobile database systems with minimum number of uplink (client to server direction) by allowing every processing unit participating in the transaction to have independent decision making capability based on the timeout mechanism. The protocol is designed for 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.

(Bobineau, Pucheral and Abdallah 2000) propose a new ACP called UCM (*Unilateral Commit for Mobile)* which aims 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. 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. 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.

Protocol	No. of wireless messages to commit a transaction (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 [24][27]	1 U + r D *r : No of invalidation reports*	*MH*	*Intermittent*	*Not applicable*
TCOT [9]	2 U + e U ***e : No. of timeout*** *extensions*	*MH* *FH*	*ontinuous*	Static or dynamic
UCM [12]	1 U + 1 D *U: uplink* *D: downlink*	*MH* *FH*	*Continuous* *Intermittent*	***Static but dynamic allowed***

Table 1: Comparison of the protocols

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. As no assumption is made about local concurrency and consistency mechanism, M-2PC can co-

exist with traditional ACP such as the traditional 2PC nodes that do not wish to implement mobility.

References

Abdallah, M. and Pucheral P. (1998) : Validation atomique: état de l'art et perspectives. *Revue Ingénierie des Systèmes d'Information (ISI)*,**5**(6).

Al-Houmaily, Y. and Chrysanthis P. (1995): Two-Phase Commit in gigabit-networked distributed database. *Proc. of the 8th int. conf. on parallel and distributed computing systems (PDCS)*.

Bernstein, P.A. Hadzilacos V. and Goodman N.(1997): Concurrency control and recovery in database systems. USA, Addison Wesley.

Bobineau, C. Pucheral, P. and Abdallah, M. (2000): A unilateral commit protocol for mobile and disconnected computing. *Proc. of the 12th int. conf. on parallel and distributed computing Systems (PDCS),* Las Vegas, USA.

Bolchini, C. Schreiber, F. A. and Tanca, L. (2004): A context-aware methodology for very small data base design. *SIGMOD Record,* **33(**1).

Chrysanthis, P. K. (1993): Transaction Processing in Mobile Computing Environment. *Proc. Of the IEEE Workshop on advances in parallel and distributed systems,* Princeton, New Jersy, USA, 7-83.

Das, S. K., Lee, E., Basu, K., Kakani, N. and Sen S. (2002): Performance Optimization of VoIP Calls over Wireless Links using H.323 Protocol, *Proc. of the 2002 INFOCOM*, 1386-1394.

Dirckze, R. and Gruenwald, L. (2000): A Pre-serialization Transaction Management Technique for Mobile-Multi-databases. *Special Issue on Software Architecture for Mobile Applications*, **5(**4).

Dunham, M., Hellal, A. and Balakrishan S. (1995): Mobile computing and databases: anything new? *Proc. of the ACM SIGMOD Record,* **24**(4).

Dunham, M., Hellal, A. and Balakrishan, S. (1997): A mobile transaction model that captures both the data and movement behaviour. *Mobile and Networks Applications*, **2**: 149-162.

Eyers, T. and Schulzrinne H. (2000): Predicting Internet Telephony Call Setup Delay. *Proc. of 1st IP-Telephony Wksp.*, Berlin, Germany.

Gray, J. (1993): Notes on Database Operating Systems. Operating Systems: An Advanced Course. *LNCS vol.60,* Springer Verlag.

Gray, J. and Reuter, A. (1993): Transaction processing: Concepts and Techniques. USA, Morgan Kaufman.

Imielinski, T. and Badrinath, B.R. (1994): Mobile wireless computing. *Communication of the ACM*, **37**(10): 19-28.

ISO (1992): Open System Interconnection- *Distributed Transaction Processing (OSI-TP) Model*. ISO IS 100261.

Kleinrock, L. 1975): Queing Systems, Volume I: Theory. USA, John Wiley & Sons.

Kumar, V., Dash, K., Dunham, M.H. and Seydim A. Y. (2000): A timeout-based mobile transaction commitment protocol. ADBIS-DASEAA 2000, in cooperation with ACM SIGMOD, Prague, Czech republic.

Liu, L., Agrawal, D. and El Abbadi A. (1994): The performance of Two-Phase Commit Protocols in the presence of site failures. Technical Report TRCS94-09. Department of Computer Science, University of California, Santa Barbara.

Lu, Q. and Satyanaranyanan, M. (1994): Isolation-Only Transactions for Mobile. *Operating System Review*, 81-87.

Madria, S. K. and Bhargava, B. K. (1998). A transaction model for mobile computing. IDEAS 1998.

Mazumbar, S. and Chrysanthis, P.C. (1999): Achieving consistency in mobile databases through localization in PRO-MOTION. *Proc. Of the DEXA Int. workshop on mobility in DB and dist. Sys.,* Italy, 82-89.

Mohan, C., Lindsay, B. and Obermarck, R. (1986): Transaction management in the R*distributed data base management system. *ACM transactions on database systems*, **11**(4).

Narasayya, V. R. (1994): Distributed transactions in mobile computing system. Technical Report. Depart. Of Computer Science, University of Washington.

Object Management Group (1994): Object Transaction Service. OMG Document 94.8.4. OMG editor.

Perron, M. and Bai, B. (1999): Low cost commit protocol for mobile computing environments. Http://www.cs.Ualberta.ca. Accessed Sept. 2003.

Pitoura, E., Bhargava, B. (1994): Revising Transaction Concepts for Mobile Computing. *Proc. of the IEEE Workshop on Mobile Systems and Applications*, Santa Cruz, Ca.

Pitoura, E., Bhargava, B. (1999): Data consistency in intermittently connected distributed systems. *IEEE Transactions on knowledge and data engineering.* **11**(6): 896-915.

Stamos, J., Christian, F. (1990): A low cost atomic commit protocol. *Proc. of the 9th symposium on reliable distributed systems*.

Walborn, G. D. and Chrysanthis, P. K. (1997): Pro-motion: Management of mobile transactions. *Proc. Of The 11th ACM Annual Symposium on Applied Computing,* Special Track on Database Technology, Van Jose, CA, 101-108.

Weikum, G. and Vossen G. (2002): *Transactional information systems, Theory, algorithms, and the practice of concurrency control and recovery.* USA, Morgan Kaufmann.

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