

Revisiting Distributed Protocols for Mobility at the Application Layer

N. Nouali, H. Drias, A. Doucet

Abstract—During more than a decade, many proposals and standards have been designed to deal with the mobility issues; however, there are still some serious limitations in basing solutions on them. In this paper we discuss the possibility of handling mobility at the application layer. We do this while revisiting 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. The solution is based on an execution framework providing an efficient extension that is aware of the mobility and preserves the 2PC principle.

Keywords—Application Layer, Distributed Mobile Protocols, Mobility Management, Mobile Transaction Processing.

I. INTRODUCTION

IN the last decade, studies that address mobility management issues have arisen, coming up with a wide variety of mobility management schemes to support different applications for so many wireless access technologies. These schemas are designed at different layers of the protocol stack. The authors of [1] present and compare a detailed set of mobility support paradigms, each representing some specific changes to the existing protocol layer. They conclude that all existing solutions have different implications to their application scenarios; there is no single perfect solution so far and current mobility solutions do not solve all general problems related to Internet mobility.

The proposed schemes are not suitable for all types of applications. But, designing a unique scheme for mobility management is very difficult and requires identifying many application paradigms. These paradigms concern, for example, the range of mobility to be supported; a host may be mobile within a building, a campus, a metropolitan area, or may cross international boundaries, the required speed of mobility directly related to the hand-off frequency; a laptop may be

used in a car, on a train, or may remain in a conference room, the mobile host may be mobile or simply portable; portable mobile hosts may not need any sophisticated mobility management scheme and Dynamic Host Configuration Protocol (DHCP) may be sufficient in most cases. Finally, the nature of applications to be supported plays an important role in deciding the desired properties of the mobility solution. The deployment of elaborate mobility management schemes may be wasteful for many applications such as World Wide Web, since the cost of intermittent disconnection due to mobility is much less than the cost associated with implementing the mobility solution. As these applications work with short-lived connections, they do not require correspondent nodes to locate and transmit to a mobile host.

In [2] we have investigated the adaptation of the Two-phase Commit protocol to the mobile wireless environment. We showed how the characteristics of the mobile and wireless environment impact the commitment of mobile transactions; we focused on the Two-phase Commit Protocol. The proposal concerned the limited resources of portables mobile clients or Mobile Hosts (MH), the disconnection handling and the mobility management topics. Our adaptation was designed on the basis of client/agent/server architecture. At the moment we started working, the hypothesis was that actually the underlying network layers do not still provide adequate mobility management, so we must manage to design a solution that offers a way to deal with mobility at the application layer and provides a correct execution of the commitment protocol without counting on the support of lower layers.

The remainder of this paper is organized as follows. Section 2 outlines the mobile environment challenging issues affecting the commitment paradigm. Section 3 presents the traditional Two-Phase Commit protocol and Section 4 the Mobile version of the protocol. Section 5 focuses discussion on the mobility management and concludes the paper by proposing enhancement strategies for the proposal and ongoing work.

II. THE CHALLENGES FACED BY THE 2PC PROTOCOL IN MOBILE ENVIRONMENT

In distributed systems, an *atomic commitment protocol* (ACP) is needed to terminate distributed transactions. The most commonly used and standardized mechanism dealing with the commitment problem is the *two-phase commit* (2PC) protocol [3]-[4] that allows the involved parties to agree on a common decision about committing or aborting the transaction even in the presence of failures.

Manuscript received April 7, 2005.

N. Nouali is with CERIST, Rue des 3 Frères Aissou, Ben Aknoun, Algiers, Algeria (Tel: 213 21 20 25, Fax: 213 21 91 21 26, e-mail: nanouali@yahoo.com, cc: nnouali@cerist.dz).

H. Drias, is with USTHB University, Faculté du Génie Electrique, Algiers, and with the INI Institut National d'Informatique, BP 68 M Oued Smar El Harrach (e-mail: drias@wissal.dz).

A. Doucet is with Université Pierre et Marie Curie, LIP6, 8, rue du Capitaine Scott -75015, Paris, France (e-mail: Anne.Doucet@lip6.fr).

Like many other protocols in distributed computing do, 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 mobile wireless environment. The global architecture considered here for such environment consists of two distinct sets of entities: mobile hosts and fixed hosts (Fig. 1). 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. The Mobile Hosts are portable devices equipped with more or less resources (CPU, memory, and power). Wireless communication induces much lower bandwidth, higher latency and error rates and more expensive cost.

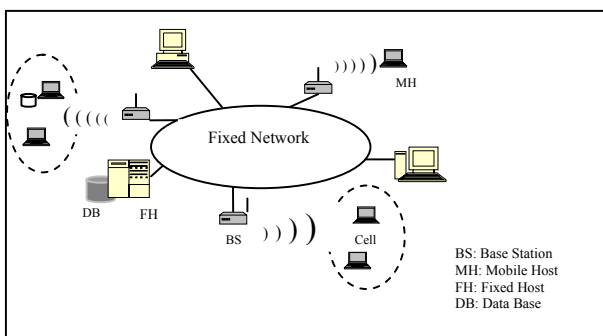


Fig. 1 Mobile system architecture

III. THE TRADITIONAL 2PC PROTOCOL

The 2PC protocol follows two steps or phases: a voting phase and a decision phase (see fig. 2). The first phase is the voting phase where the *coordinator* (generally the origin of the transaction) asks all the sites involved (*participants*) in the transaction to prepare to commit the transaction. If, any of the participants responds *No*, the coordinator decides to abort the local branch of transaction and informs all participants. If all the votes are *Yes*, the coordinator then decides to commit and informs all the participating. 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. The participants acknowledge the coordinator decision.

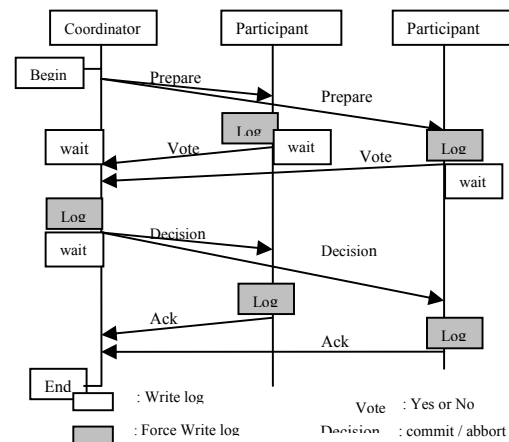


Fig. 2 The 2PC Protocol

IV. THE M-2PC PROTOCOL

The objective of our M-2CP (Mobile-2PC) 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 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.

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.

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 there 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 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.

In the case of a mobile server, 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. For more details on the M-2PC protocol and its correctness see [2].

V. DISCUSSING THE MOBILITY MANAGEMENT IN M-2PC

The 2PC protocol is a distributed protocol, i.e.; several entities participate in its execution and cooperate to terminate a transaction consistently by exchanging messages. This message exchange is directly and principally impacted by the wireless communication characteristics and the mobility of the

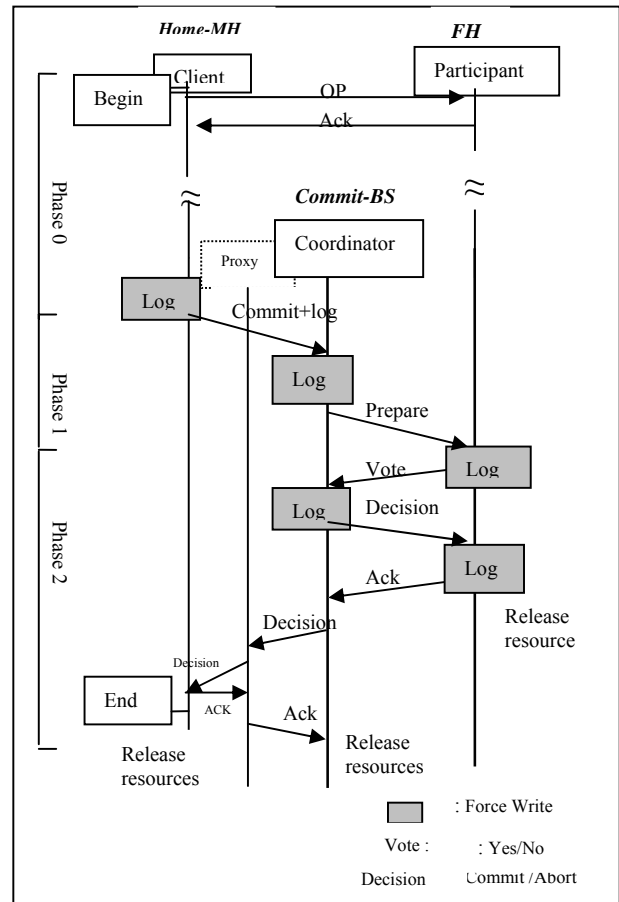


Fig. 3 The M-2PC protocol

MHs. In [2] we proposed an implementation framework for a commit protocol which supports 2PC functionality for mobile distributed transactional systems¹. Thus the goal was to make the protocol execute and terminate correctly, i.e.; preserving the semantic of the 2PC paradigm as it was originally defined. This is obtained if all the parties executing cooperatively the protocol have always the possibility of exchanging the information needed to terminate correctly the protocol (take unanimously the decision of abort/commit) and if all of them especially the MHs get the results. In other words no message concerning the protocol execution must be lost during a disconnection or a handoff. During disconnections the continuity of service is guaranteed thanks to the three-tier architecture where the agents (Coordinator for the mobile client and participant-agent for the mobile server) execute on behalf of the MHs. During handoff, the MHs are in charge of telling their correspondents about the new locations by sending them a message after registration in a new cell. This solves the problem of the address change; the MH must record identity and location information of the correspondent as it needs it when it registers at a new BS. Also, no loss of messages appears during the handoff process.

¹ In this paper we focus on the mobility management topic we do not further discuss the other constraints.

Concretely speaking, the agents² play the role of proxies to handle mobility. Each time the MH moves to a new cell, it registers itself to the new agent/proxy. The new agent/proxy contacts the old one that is located on the old BS to inform it about the new address of its attached MH. The old agent/proxy may then send messages to the MH. The information concerning the old agent/proxy is included in the registration message the MH sends to the new/agent proxy.

Actually the agent may have a double role and may be divided into two entities³. One entity is responsible for insuring the ACP duties; this part acts as a coordinator agent representing the client on the fixed part of the network and acts as a participant agent representing the server executing a branch of the transaction. The other entity is responsible for handling the mobility duties; this part acts as a proxy. The proxy part is designed to track the MH movement, i.e.; handles the roaming issues. This mechanism is in charge of maintaining the contact between the MHs and the other partners; in one hand this resolves location update problem, in the other hand it resolves the handoff problem as the new proxy requests the old proxy one and the coordinator to transfer the state information.

Our approach provides many advantages. First, it does not require modification of lower layer protocols, thus, it may work over the actual TCP/IP architecture. Second, as it is embedded in the protocol, it is more likely to capture the semantic of the application. This is interesting as each application has its own needs with regard to mobility. Third, fixed hosts that do not wish to implement mobility may co-exist with the mobile ones. For example, M-2PC can co-exist with the traditional 2PC at nodes that do not care of mobility.

The SIP signalling protocol is to our knowledge the first protocol to be proposed for mobility management at the application layer. SIP was initially developed by IETF as an application layer multimedia signalling protocol [5] and is recently adapted to provide Internet mobility support [6] without any modifications of lower layer protocols. However, the handover of SIP incurs considerable handover latency that is not suitable to real-time communications. Two principal variants of SIP tried to reduce the latency of mobile SIP [7]-[8].

As for SIP, the approach presented in this paper presents disadvantages too. In addition to the fact that our ACP is an application layer protocol thus its messages may not be served with highest priority, its handoff procedure also introduces a latency that may be not sustainable for real-time applications. In [2] we showed 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 correspondent nodes 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.

REFERENCES

- [1] D. Le, X. Fu and D. Hogrefe, "A Review of Mobility Support Paradigms for the Internet," *Tech. Rep. N°IFI-TB-2005-01*, C.S. Institute, Georg-August University, January 2005.
- [2] N. Nouali, A. Doucet, and H. Drias, "A Two-Phase Commit Protocol for Mobile Wireless Environment," in *Proc. Sixteenth Australasian Database Conference (ADC2005)*, Newcastle, Australia. CRPIT, **39**. Williams, H. E. and Dobbie, G., Eds., ACS. 2005, pp. 135-144.
- [3] P.A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency control and recovery in database systems (Addison Wesley, 1987)".
- [4] G. Weikum, G. Vossen, "Transactional information systems, Theory, algorithms, and the practice of concurrency control and recovery", USA: Morgan Kaufmann, 2002.
- [5] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol", *RFC 2543, Internet Engineering Task Force*, March 1999.
- [6] E. Wedlund, and H. Schulzrinne, "Mobility support using SIP' WOWMOM'99," in *Proc. of Second ACM International Workshop on Wireless Mobile Multimedia*, Seattle, Washington, USA, 1999.
- [7] D. Vali, S. Paskalis, A. Kaloxylos and L. Merakos, "An Efficient Micro-mobility Solution for SIP Networks," *IEEE globecom 2003*, San Francisco, CA, USA, 2003.
- [8] W. Kim, M. Kim, K. Lee, C. Yu and B. L. Link. "Layer Assisted Mobility Support Using SIP for Real-time Multimedia Communications," *ACM MobiWac*, 2004.
- [9] F. Chahbour, N. Nouali and K. Zeraoulia, "Fast Handoff for Hierarchical Mobile SIP Networks" accepted for presentation at the *3rd World Enformatika Conference, WEC'05* to be held on April 27-29, 2005 in Istanbul, Turkey.
- [10] W. Kim, M. Kim, K. Lee, C. Yu and B. Link, "Link Layer Assisted Mobility Support Using SIP for Real-time Multimedia Communications," *ACM MobiWac*, 2004.

² The agents are supposed to execute on the BSs on the fixed part of the network.

³ The architecture becomes four-tier architecture.