GMTC: A Generalized Commit Approach for Hybrid Mobile Environments

Brahim Ayari, Abdelmajid Khelil, and Neeraj Suri

Abstract—Mobile environments increasingly require distributed atomic transactions to support the growing diversity of financial, gaming, social networking and many other applications. The underlying mobile infrastructure is correspondingly evolving with increasingly diverse wired and wireless elements and also with increasing exposure to a variety of operational perturbations at the mobile elements and communication levels. Consequently, the challenge is not only in providing efficient nonblocking mobile commit (as a fundamental basis behind consistent mobile transactions) but to also provide efficient perturbation-resilient atomic commit in the heterogeneous mobile space. The contribution of this paper is in developing a perturbation-resilient mobile commit protocol that efficiently provides for and preserves strict atomicity for transactional applications. The protocol does not necessarily require access to the powerful communication/computation elements of the wired infrastructure during transaction execution. However, in case access to a wired network becomes possible, it then adapts to utilize this to 1) increase the resilience to network perturbations achieving higher commit rates, and 2) reduce the wireless message overhead and the blocking of transaction participants leading to higher transactions throughput. In contrast, existing solutions are often tailored either for 1) infrastructure-based mobile environments, or 2) infrastructure-less ad hoc networks. To our knowledge, there is no existing commit protocol that can adapt across diverse infrastructure communication modes. The proposed perturbation-resilient generalized mobile transaction commit (GMTC) protocol represents the first atomic commit protocol for hybrid mobile environments which 1) takes advantage of accessing infrastructures, by choosing reliable infrastructure nodes for coordination of transactions and for replication of commit data of mobile participants to tolerate network disconnections, and 2) tolerates network partitioning and delivers best-effort results—in terms of transaction commit rate, message complexity, and commit/abort decision time (latency)—if the access to wired infrastructure is unavailable. The protocol performance simulations (covering transaction commit rate, message complexity, and commit/abort decision time) demonstrate the effectiveness of the developed protocol in generalized mobile environments.

Index Terms—Mobile computing, mobile databases, distributed mobile transactions, atomic commit, fault tolerance

1 Introduction

The increasingly pervasive mobile computing ecosystem spans a diverse variety of e-commerce, medical, infotainment, social networking, monitoring, tracking, notification, etc., applications. While the nature of application specific data is naturally diverse spanning requirements on granularity, timeliness and reliable delivery, a core need across the broad applications space is to guarantee the consistency of data.

The linkage of mobile devices to hybrid—composite of wired and wireless—infrastructures raises basic database issues. Typically, data is stored and managed in hybrid mobile environments by databases installed on both mobile and stationary devices, and accessed through interlinked wired and wireless systems. Accordingly, hybrid mobile environments are increasingly supporting applications that require data consistency which is then guaranteed by the atomicity property of database transactions [16]. However, as applications, such as online purchases, involve both a wired infrastructure (servers) and wireless infrastructure-less (mobile devices) components with potential for ad hoc networking across mobile devices, data consistency is required on all participating wired and wireless resident databases. Similar coordination, computation and communication couplings across infrastructure and infrastructure-less elements exist for examples such as platoon driving, intervehicle communication or coordinated wireless control elements in a smart factory.

1.1 Communication Modes and Perturbations

Characteristic for the underlying mobile environments is an evolving heterogeneous communication mode spanning ad hoc infrastructure-less modes, infrastructure-based modes and their combination. Also the connectivity to a wired network is often sporadic either due to lack of infrastructure or due to high costs/expenses. Mixing communication modes also results in a wider range of failure modes covering both standard disruptions (device, host or base station failures, energy depletion, etc.) and new ones at mode interfaces. For instance, hybrid mobile environments usually suffer from frequent network disconnections and partitions due to the natural constraints of wireless networks and the mobile devices forming them. Mobile atomic commit protocols need to consider the new composed perturbation model to maintain the required service availability and data consistency [13].

1.2 From a Dedicated Single Communication Mode to an Adaptive Multimode Commit Solution

Commit protocols are typically tailored, for performance or efficiency, for a dedicated communication mode within the

- *B. Ayari is with the Information Systems Department, ABB AG, Reuteralle 22, Darmstadt 64297, Germany. E-mail: br.ayari@googlemail.com.*
- *A. Khelil and N. Suri are with the Department of Computer Science, Technische Universität Darmstadt, Hochschulstr. 10, Darmstadt 64289, Germany. E-mail: {khelil, suri}@cs.tu-darmstadt.de.*

Manuscript received 18 Mar. 2011; revised 9 Feb. 2012; accepted 22 Sept. 2012; published online 1 Oct. 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-03-0148. Digital Object Identifier no. 10.1109/TMC.2012.203.

wide range of generalized infrastructure/infrastructure-less mobile scenarios. Naturally, they are not expected to perform efficiently in the other modes they are not designed for. For instance, commit protocols that are designed for wired networks, such as the traditional two-phase commit (2PC) protocol [15] and its optimizations such as [25] rely on reliable communication between the transaction participants, rendering their applicability in a generic (typically hybrid) mobile environment limited as network disconnections and partitions prevent such reliable communication.

Consequently, and to gradually cope with the design complexity, most research efforts target efficient perturbation-resilient commit for a specific grouping of related communication modes. Some commit protocols are developed for transactions involving fixed and mobile participants with infrastructure access [5] such as FT-PPTC [1]. These approaches explicitly require some nodes in the wired network to coordinate mobile transactions. Therefore, these protocols are customized for the infrastructure-based communication mode with its specific perturbation model. Accordingly, this class of commit techniques is not directly adoptable for the mixed mode environments because it does not cover its larger set of commit perturbations as the connectivity to the infrastructure is only occasional. Other efforts address transactions for participants that are interconnected through ad hoc communication mode only. These commit protocols developed for infrastructure-less mobile environments such as ParTAC [4] tend not to be efficient in hybrid mobile environments as ignoring the available infrastructure unnecessarily increases the transaction abort rate, wireless messages overhead and latency. The main achievements of current research efforts are discrete solutions that are customized for specific communication modes and in covering the corresponding subset of commit perturbations.

The diverse and evolving nature of a generalized mobile environment obviously makes it challenging to predict the future class of communication mode(s) connecting the transaction participants. A typical approach is to select (in a static manner) a suitable commit strategy for each mobile transaction at its initialization and then conduct (an often complex) mode switch as the communication environment changes. Hence, the approach taken in this paper is that of an adaptive and composite protocol that 1) concatenates known efficient solutions from each dedicated communication mode (fixed, mobile infrastructure-based, or mobile infrastructure-less modes) and 2) provides an intermode adaptation strategy to efficiently switch to the current applicable communication mode or their combination as needed. To meaningfully conduct a comparison with existing specific mode solutions, we provide this discussion of related work at the paper end in Section 7.

1.3 Summary of Paper Contributions

We propose generalized mobile transaction commit (GMTC), the first perturbation-resilient atomic commit protocol for hybrid mobile environments that combines the advantages of fixed, infrastructure-based and infrastructure-less solutions.

- GMTC takes the advantage of infrastructure-based protocols, if an infrastructure is available, by choosing the more reliable and available static fixed nodes to coordinate mobile transactions and to replicate commit data needed to tolerate network and mobile node perturbations.
- GMTC delivers best effort transactional service availability in case no infrastructure is accessible. In Section 3.2, we summarize the main advantages and contributions of GMTC compared to the two most recent representatives of existing related work: FT-PPTC [1] as a representative of perturbation-resilient solutions for infrastructure-based environments and ParTAC [4] as a representative of perturbation-resilient solutions for mobile ad hoc networks.

Our approach is to interoperate three protocols each representing efficient solutions for their dedicated communication modes, namely 2PC, FT-PPTC, and ParTAC approaches, and then an adaptation strategy to switch between them depending on the dynamic mixture of communication modes and their resulting perturbations. The integration requires efforts to efficiently detect available communication modes and their perturbations and to accordingly tune the parameters of the superimposed commit building blocks. The key challenges we address to design GMTC are as follows:

1. An intelligent strategy to select the transaction coordinator(s) among heterogeneous participants with different resource allocation requirements.
2. The efficient collection of commit votes from these mixed participants.

Whereas FT-PPTC uses a single coordinator schema and ParTAC uses a multiple coordinator schema which converges at the end of the transaction to a single mobile coordinator in case of successful transaction execution, we present in this paper an integrated approach. We follow a shrinking-based scheme (similar to ParTAC) starting from a certain number of coordinators. However, besides reducing the number of coordinators the new shrinkage strategy pushes the coordinator role into the infrastructure if available. We design new mechanisms that realize the vote collection while following two simultaneous goals: Push the votes to any reachable coordinator and/or to an available infrastructure. The challenge that we address in this paper is to provide an efficient and seamless integration of commit "modes" to perturbation modes without the global knowledge of the evolving mixed communication modes. We will show, that our integrated approach allows covering the generalized mobile environment while converging to 2PC for transactions in wired environments, to FT-PPTC commit in infrastructure-based mobile environments and converging to ParTAC transaction commit if the hybrid environment degenerates to a pure ad hoc environment.

1.4 Paper Organization

Section 2 details the system model along with a comprehensive classification of perturbations. The overall protocol design requirements and objectives are discussed in Section 3. In Section 4, we detail the GMTC protocol. The protocol is evaluated in generalized mobile environment settings in Section 6. Related work is described in Section 7.

2 System Model and Perturbations

We first detail the system model of the hybrid mobile environment, where strict atomicity is essentially required

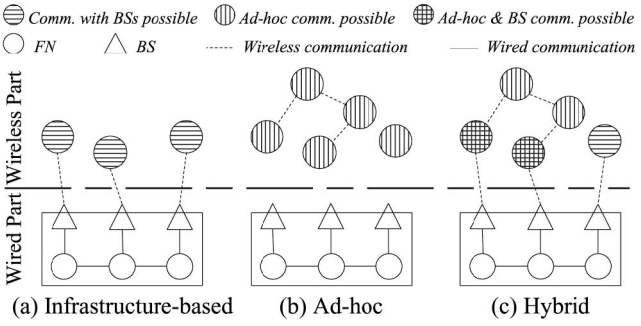


Fig. 1. Hybrid versus infrastructure-based and ad hoc mobile environments.

by applications. Next, we identify the relevant perturbations, i.e., constraints and failure modes that can occur in the considered environment to affect atomic commit functionality.

2.1 System Model

To consider and support a broad class of mobile applications, we develop a hybrid mobile distributed environment consisting of a set of mobile nodes (MN) and a set of static fixed nodes (FN). We assume that every node in this environment has a unique ID. The architecture of the environment considered is illustrated in Fig. 1c. Some MNs are equipped with appropriate wireless interfaces and can intermittently connect to the wired network through base stations (BS) via unreliable wireless channels (see Fig. 1). A subset of the MNs can communicate directly or multihop with each other in an ad hoc manner for instance using Bluetooth, WLAN ad hoc mode and so on. Every MN has at least one wireless interface and is able either to communicate only with BS or only ad hoc with other MNs or with both. When the communication between MNs is only infrastructure-based (i.e., through BSs' communication services), the environment is called *infrastructure-based mobile environment* (see Fig. 1a). In *ad hoc mobile environments*, MNs can communicate with each other only in ad hoc mode (see Fig. 1b). If both communication modes are supported by MNs, we are dealing with a *hybrid mobile environment* (see Fig. 1c).

We consider applications, which run on either MNs or FNs and access data located on *mobile* and/or *fixed* nodes. We focus on distributed transactions issued by either MNs or FNs and involving other MNs and/or FNs as participants. We refer to a distributed transaction where at least one MN participates in its execution as a *mobile transaction (MT)*. Commonly, an MT T_i is defined as a set $F = \{e_{i1}, \ldots, e_{in}\}$ of n "execution fragments" distributed among a set of locations (also sites) either mobile or fixed [20]. The node, where T_i is initiated, is termed as MT *initiator*. The *commit set* consists of all FNs and MNs participating in execution and commit of T_i including the initiator. FNs and MNs in the commit set are called *participant FNs (P-FNs)* and *participant MNs (P-MNs)*, respectively.

The transaction *coordinator (CO)* is responsible for storing information concerning the state of the transaction execution. Based on the information collected from and about the transaction participants, the CO takes the decision to either commit or abort the transaction and shares this decision with all participants. In this paper, we consider that the application/user is able to specify an appropriate (tolerable) *lifetime* for each initiated MT. The lifetime of an MT is defined as the maximal timeout the CO should wait (as long as there is no final decision) before deciding about the outcome of the MT.

We do not assume bounds on message transmission times between communicating nodes and also on data processing times on a node. We consider nodes to have accurate (not necessarily synchronized) clocks.

2.2 Perturbations

For mobile systems supporting transactional applications, we consider two main classes of perturbations: *operational constraints* (battery power, computing, connectivity, etc.) and *failures*.

2.2.1 Operational Constraints

The mobile environment is constrained by the characteristics of both *MNs* and *wireless links*. MNs inherently possess restricted computational capabilities such as computational and storage capacity. These resource constraints may lead to execution failures. MNs may also run in different energy modes or might be turned-off to save energy. Additionally, wireless links are characterized by high latency and restricted bandwidth. These characteristics lead to unreliable wireless links and thus considerably varied reliability/availability and connectivity of MNs.

2.2.2 Failures

We now outline the common failure modes and classify them into classes of communication and node failures.

Communication failures. These constitute the majority of failures in mobile environments. We consider three types of communication failures: *Message loss*, *network disconnection* and *network partitioning*. Messages exchanged across MNs are highly vulnerable to loss due to the high bit error rate of wireless links and possible network congestion and collisions. Also high node mobility often disrupts routes and causes message loss. Given its mobile nature, an MN can enter a geographical area out of coverage of any BS so that it loses its connection to the network. Network disconnections are specific to infrastructure-based environments and consequently a common communication failure in hybrid environments. The MN is said to be *disconnected* from the corresponding BS and hence from rest of the network. While disconnected from the network, the MN is not able to send or receive messages. Network partitioning occurs only in mobile ad hoc environments. Therefore, network partitioning represents a common occurrence in hybrid mobile environments. Due to the inherent node mobility and autonomicity, the hybrid mobile environment can easily get partitioned and reconnected. As network partitioning is the norm rather than the exception in a hybrid mobile environment, it needs to be explicitly considered in the design of atomic commit protocols.

MN failures. We consider only *transient* MN failures as permanent MN failures can be interpreted as a permanent network disconnection or partitioning. Transient MN failures occur from either software or hardware faults and usually disappear if the MN reboots. A common cause of transient failures is the lack of battery power to sustain

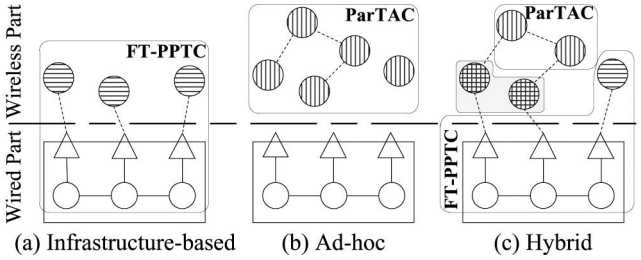


Fig. 2. Objectives of the proposed approach.

operation of the mobile device. Transient failures are the most probable failures of MNs. Transient MN failures can manifest for the transaction commit problem as a transient network partitioning, i.e., the MN disconnects from the network if a transient failure occurs and reconnects once this failure disappears and the MN recovers.

FN failures. We assume that if an FN crashes it stops receiving, sending, and processing messages until it recovers after a finite amount of time (crash-recovery model).

3 Requirements and Objectives

We briefly summarize the design requirements for a generalized integrated commit approach for hybrid mobile environments. A primitive requirement analysis on such generalized commit without providing a concrete solution has been conducted in [2] and [3]. Next, and starting from the limitations of the infrastructure-based FT-PPTC and the infrastructure-less ParTAC approaches, we present the design objectives for our integrated approach.

3.1 Design Requirements on Generalized Commit

Overall, a generalized commit should efficiently *integrate* the benefits of existing commit island solutions. In particular, a generalized commit solution should show comparable performance to established commit protocols such as 2PC, FT-PPTC, and ParTAC in wired networks, infrastructure-based mobile environments, and infrastructure-less ad hoc environments, respectively. We identify the following main requirements and design issues on our integrated generalized commit approach.

Resilience to perturbations. To build resilient mobile transaction protocols, the first requirement is to define a comprehensive set of perturbations (constraints and failures) and a set of techniques to cope with constraints and recover from failures. The overall objective for perturbation-tolerance is to maximize the number of committed MTs.

Delay-tolerance and -awareness. Masking latent faults such as long disconnections imposes that the MT execution time can be delayed till local commit/abort decisions can be collected. This implies that MT can last for minutes or even hours. We are dealing then with transactions that we refer to as *delay-tolerant transactions*. We believe that users can sacrifice latency for atomicity. The delay-tolerance design requirement is orthogonal to the efficiency requirement and implies a real challenge for the design of MTs.

Efficiency. The efficiency of MT protocols is measured in terms of message complexity and resource blocking time. The classical approach to improve the efficiency of such protocols is to reduce the communication overhead (message number and size) and to minimize the resource

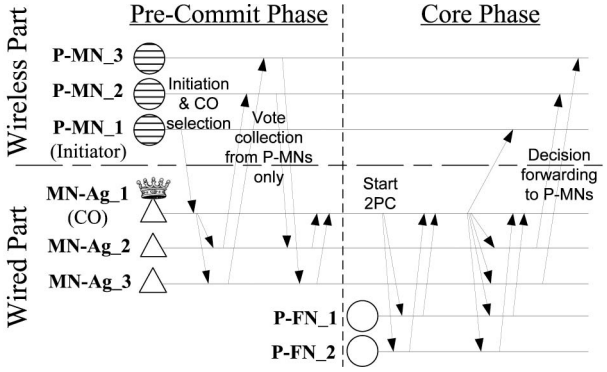


Fig. 3. FT-PPTC flow diagram.

blocking time. The reason behind minimizing resource blocking time is that transactions, especially executing on FNs, often lock expensive resources. The more transactions per second an application can process, the better its scalability and throughput. If resources are blocked, transactions using them are delayed waiting for the resources to be unlocked. The throughput of the system then suffers. For this reason, resource blocking time, especially of FN resources (because they are frequently much more loaded than MNs), should be minimized.

3.2 Objectives: Integrated Generalized Commit

Our primarily goal is to offer efficient perturbation-resilience to atomic commit protocols in hybrid mobile environments. We start with analyzing the building blocks of our prior island commit solutions (FT-PPTC and ParTAC) to identify their limitation in providing a perturbation-resilient commit solution in hybrid mobile environments. The FT-PPTC transaction atomic protocol [1] developed for infrastructure-based mobile environments

1. fulfills our efficiency requirements with respect to reduction of blocking time of FN resources,
2. copes with a wide range of perturbations encountered in mobile environments,
3. is delay-aware, and
4. transforms into 2PC in pure wired environments.

ParTAC is an atomic commit solution for mobile ad hoc environments which is designed to tolerate network partitioning in an efficient manner compared to other existing solutions [4].

Accordingly, it is natural to integrate these two approaches to build a perturbation-resilient, delay-aware and efficient solution for hybrid mobile environments. Fig. 2 (with the same legend as Fig. 1) visualizes our objective. The shaded area in Fig. 2c which is not covered by any protocol represents the gap/challenge for integrating FT-PPTC and ParTAC. The figure shows that the integration of these two protocols is not trivial lack of global view concerning the evolving mixture of communication modes in the network environment. We briefly describe the strength of FT-PPTC and ParTAC in infrastructure-based and infrastructure-less environments respectively, along with their limitations in generalized network settings.

As illustrated in Fig. 3, FT-PPTC decouples the commit of P-MNs from that of P-FNs. The execution of the transaction is therefore split in *two phases*. In the *precommit phase*,

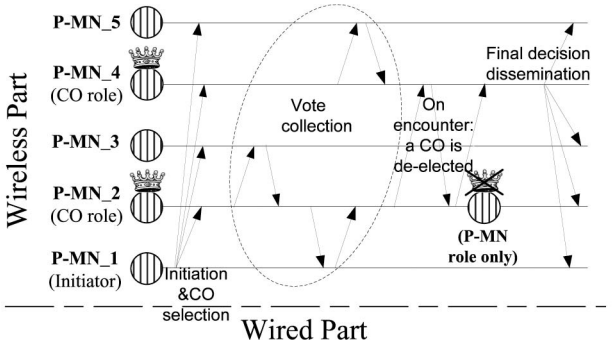


Fig. 4. ParTAC flow diagram.

"sufficient" information from mobile participants is collected to reduce the set of nodes participating in the MT execution to a set of FNs. In the second phase, called *core phase*, the commit involves only FNs and thus can be simply completed by any atomic commit protocol from wired networks, such as the established 2PC protocol. To allow for this decoupling, each P-MN should rely on a representant in the infrastructure that we refer to as mobile node agent (MN-Ag). An MN-Ag is a logical entity (proxy, private cloud, etc.) representing the P-MN in the wired network. The MN-Ag is responsible for storing all the information related to the state of MTs involving the MN and is also responsible for executing the 2PC protocol on behalf of its corresponding P-MN. As shown in [1], decoupling reduces the blocking time of the resources at the FNs. It also simplifies the handling of the different kinds of failures that raise from the mobility of P-MNs. For readability purposes, Fig. 3 illustrates only the failure-free execution of FT-PPTC. The critical commit perturbation is the disconnection of P-MNs, which can be easily addressed by a coordination scheme between the P-MN and its MN-Ag. To this end, we assume that a P-MN can inform its agent about the estimated time to execute its transaction fragments (E_t) and the time to ship the results to the agent (S_t).

Fig. 4 shows the main building blocks of the ParTAC protocol. In ParTAC, a set of coordinators is preselected among the P-MNs (the preselection of COs out of the P-MNs can be either random or based on node properties such as IDs, mobility, connectivity, storage capabilities, etc.). Every preselected CO can safely abort the MT upon expiration of the MT lifetime if no decision is reached by then. The preselected COs collect votes from P-MNs. When two COs encounter each other, they exchange their collected votes and elect a single active CO among themselves. The other CO immediately stops playing an active CO role and continues behaving like a P-MN. As a result, if all COs transitively encounter each other before the expiration of the MT lifetime, only one active CO remains which will take the final decision for the MT.

Table 1 summarizes why FT-PPTC and ParTAC fail in delivering a solution for hybrid environments and briefly highlights our requirements on the new GMTC approach. Transaction commit protocols developed for infrastructure-based mobile environments in general (and FT-PPTC in particular) consider only the choice of mobile transaction COs as FNs because of the availability and perturbation-resilience of these nodes compared to MNs. These studies do not consider at all the choice of MNs as transaction COs because of the nature of the mobile environment. To apply these approaches in mobile ad hoc environments as a special case of hybrid environments, the choice of the CO becomes a challenge. Not only MNs are not having stable storages to rely on them for the complete time of transaction execution but also their availability and reachability is not guaranteed even for small periods of time as opposed to FNs. Subsequently, these developed solutions simply fail in ad hoc environments mainly because they do not define how to choose the MT CO in this environment. In hybrid mobile environments, these approaches might work provided the MT initiator is able to connect to the infrastructure to select an FN there to play the CO role. Unfortunately, this solution is in many scenarios inefficient since all the traffic between the CO and P-MNs will flow in this case through the MT initiator.

If we now consider the transaction commit protocols developed for mobile ad hoc environments (e.g., ParTAC), we observe that COs can only be chosen among MNs as no FN is available. These approaches rely on choosing more than one P-MN to play the CO role, and thus overcoming the vulnerabilities of the P-MNs by replicating the CO role. Though these commit protocols can be deployed in infrastructure-based environments as a special case of hybrid environments, they are inefficient due to an unnecessary overhead caused by the replication of the CO role and also long blocking time of FNs participating in the MT execution. It is noteworthy that the mobile ad hoc approaches do not explicitly consider the existence of P-FNs and therefore do not provide required mechanisms to minimize their blocking times. In hybrid mobile environments, these protocols might be inefficient especially when some transaction participants are FNs or in case all participants are MNs but the access to the infrastructure is available.

TABLE 1
Requirements on the Proposed GMTC Approach

	Infrastructure-based mobile environment (possibly P-FNs)	Ad-hoc mobile environment	Hybrid mobile environment (possibly P-FNs)
FT-PPTC [1]	Perturbation-resilient and efficient solution	Fails due to lack of infrastructure	Works only if the MT initiator can connect to an infrastructure
ParTAC [4]	Ignoring the decoupling of P-MN commit and P-FN commit, extra overhead and long blocking time of P-FNs frequently occur, rendering ParTAC extremely inefficient	Perturbation-resilient and efficient solution	Wastes valuable resources on MNs (energy, bandwidth) as it ignores communication shortcuts through the infrastructure
GMTC	Perturbation-resilient solution, requires additional message overhead	Perturbation-resilient and efficient solution (no extra message overhead)	First perturbation-resilient and efficient solution

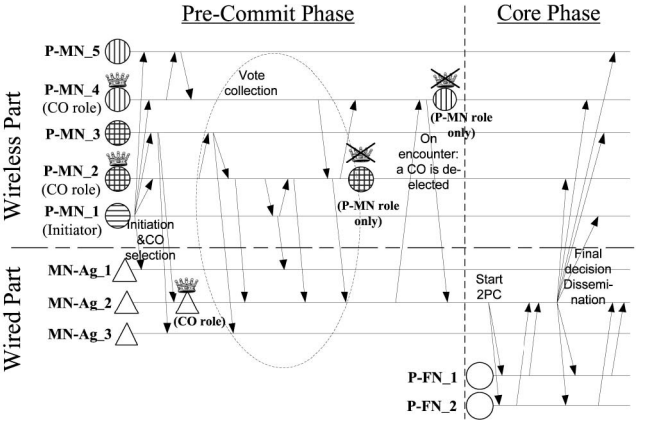


Fig. 5. GMTC flow diagram.

4 The Proposed GMTC Approach

We describe GMTC, our solution towards a perturbation-resilient transaction atomic commit protocol for hybrid mobile environments. GMTC is an efficient integration of the ParTAC and FT-PPTC approaches. GMTC converges to the FT-PPTC approach if deployed in an infrastructure-based mobile environment with an additional message overhead over FT-PPTC needed to elect one single CO among the preselected ones (FT-PPTC selects upfront a single CO). GMTC converges to the ParTAC approach if deployed in ad hoc mobile environments without any additional message overhead. The GMTC approach introduces a modified CO election strategy that takes into consideration a prioritization scheme between the preselected COs as will be explained in the following. Note that in hybrid environments, the CO role can be played by P-MNs and/or their corresponding MN-Ags if the P-MNs can communicate with the available infrastructure.

4.1 Overview

Targeting an atomic transaction solution which covers all three main classes of environments as illustrated in Table 1, we propose the GMTC protocol as a solution for hybrid mobile environments. This solution delivers the same perturbation tolerance of ParTAC and FT-PPTC in ad hoc and infrastructure-based mobile environments, respectively. GMTC adds a tolerable performance overhead if deployed in infrastructure-based mobile environments due to the new modified election strategy used in GMTC.

Fig. 5 shows that the GMTC protocol inherits from the FT-PPTC (see Fig. 3) its two phases, i.e., the precommit phase and the core phase. In the first precommit phase, GMTC follows the multiple CO strategy of ParTAC (see Fig. 4) with one main difference: The multiple CO strategy in GMTC favors P-MNs which can connect to the infrastructure while electing a new CO on encounter. If two COs, which are only able to communicate in ad hoc mode, encounter each other then the election is done like described in ParTAC, for example, the one having the highest ID is elected. If one CO, which is able to access the infrastructure, encounters a CO which is able only to communicate in ad hoc mode, the former is elected and its MN-Ag becomes one CO of the MT in the wired part of the environment. In case two MN-Ags are selected in the above described way to be COs, then any election algorithm for wired networks is convenient. We detail in the following the protocol operations of GMTC that describe the activities of P-MNs, COs and MN-Ags, respectively, in Algorithms 1, 2, and 4. P-FNs activities are described in Section 4.5. We distinguish between the activities performed during the precommit phase and the ones performed during the core phase.

4.2 Protocol Operations: Activities of P-MNs

Precommit phase. Algorithm 1 mainly describes how a P-MN sends its vote to its MN-Ag or to an encountered CO. Upon receiving its execution fragment, the P-MN begins the processing of its execution fragment $e_i(P\text{-}MN)$. If the P-MN can communicate with a BS, it sends a "No" vote to its MN-Ag whenever it decides to abort the MT and its updates if it successfully completes the execution of its fragment. In case the P-MN is not able to access the infrastructure, it sends its vote to each CO it encounters as long as no ACK is received from that CO and there is no final decision (Algorithm 1, L 17 and L 19-24). P-MNs know that they are encountering a CO when they receive a beacon from that CO as described in Section 4.3. Hence even if one CO was not aware about the P-MN's vote, for example, due to message loss, then the vote information is not lost, but communicated to the next encountered CO. It is noteworthy that a P-MN is not allowed to change its vote once it is sent to a CO.

Alg. 1: P-MN's Activities in GMTC

```
1  wait for receiving a mobile transaction T_i;
2  extract the corresponding execution fragment e_i(P-MN) and
   the set of preselected COs;
3  start executing the received execution fragment;
4  if P-MN decides to abort T_i then
5      abort T_i;
6      if P-MN has MN-Ag then
7          send "No" vote to corresponding MN-Ag
8      else
9          send "No" vote to all COs in C_m;
10         exit;
11     end
12 else                          /* P-MN decides to commit T_i */
13     write updates to the local log;
14     if P-MN has MN-Ag then
15         send updates to the corresponding MN-Ag;
16     else
17         send "Yes" vote to all preselected COs;
18         while waiting for the final decision about the outcome
           of T_i do
19             if beacon is received from a CO then
20                 send "Yes" vote to the CO from which the
                   beacon was received;
21             end
22             if Ack is received from a CO then
23                 stop reacting on beacons received from
                   that CO;
24             end
25         end
26     end
27     if final decision is Commit then
28         commit T_i;
29         exit;
30     else                              /* decision is Abort */
31         abort T_i;
32         exit;
33     end
34 end
```

4.3 Protocol Operations: Activities of COs

Precommit phase. Algorithm 2 details how a GMTC CO 1) manages the state of an MT depending on received votes

and the MT lifetime, 2) discovers surrounding P-MNs, and 3) may turn to a simple P-MN if it encounters another CO. Upon receiving an MT T_i, a CO creates a *Token* for the received MT, which includes all information about the participants and COs of the MT and the state of execution. The possible execution states are: Idle, active, precommitted, committed or aborted. The state of T_i is set to "active." If a CO is a P-MN in the MT, it starts executing its execution fragment upon receiving the MT T_i. Every CO starts a timer to detect/watch the expiration of the lifetime of the MT (Algorithm 2, L 9). If a CO receives initial or updated E_t and/or S_t from one P-MN, the lifetime of the MT is updated if mandatory, i.e., if the lifetime needs to be increased. The updated information is stored in the Token of the CO which received that information. If a CO decides to vote for aborting the MT and it is a P-MN or receives a "No" vote, it sends an Abort decision to all P-MNs of the MT. The COs periodically send presence beacons to allow other P-MNs and COs in their partition to discover their presence (Algorithm 2, L 11). These beacons are those already being sent by the underlying ad hoc routing protocol to avoid additional wireless messages. Every preselected CO maintains a commit-list L of all P-MNs from which it has received a "Yes" vote. If the CO is a P-MN and decides to vote for committing the MT or if it is an MN-Ag and receives the updates of its corresponding P-MN, it adds also its ID to its own commit-list (Algorithm 2, L 24). As soon as a CO receives a "No" vote it decides to abort the MT and sends an Abort decision to all P-MNs. If the lifetime of the MT expires on a CO before receiving a final decision, the CO decides also to abort the MT (Algorithm 2, L 59-60).

If two COs encounter each other (e.g., if the corresponding network partitions join) these two COs exchange their commit-lists (Algorithm 2, L 33-43 and 46-47) and elect one CO among themselves, for example, based on highest ID (Algorithm 2). The other CO becomes either a normal P-MN or an MN-Ag (Algorithm 2, L 44) and behaves from this point in time and onwards according to Algorithms 1 or 4, respectively. If one of the COs is an MN-Ag, it is elected automatically and if both are MN-Ags the highest ID schema or another election algorithm can be used. COs are allowed to give their list of votes only to other COs and only after they complete the election process. The nonelected CO sends its commit-list to the elected one that merges it with its own list. Thus, the lists are merged only if the election succeeds. If the list does not reach the elected CO, for example, due to message loss, it is still possible that GMTC commits the MT since the nonelected CO will behave as a normal P-MN and sends its commit-list L to every CO it encounters or it will send the commit-list to the elected FN CO if it is an MN-Ag.

The election process as described above guarantees the uniqueness of the decision. We note that the votes of COs can only be given to other COs after the election process. Using this schema for the election of a new and single CO guarantees that no two or more COs have the complete knowledge about which P-MNs voted to commit the MT. In the latter case, these COs could take different decisions about the outcome of the MT which violates the correctness of the proposed solution.

Alg. 2: CO's Activities in GMTC

```
1  wait for receiving a mobile transaction T_i;
2  extract the corresponding execution fragment if the CO is a
   P-MN or the fragment of its corresponding P-MN if it is a
   MN-Ag, the lifetime of the MT, the set of P-MNs and
   preselected COs;
3  create a Token for T_i ;
4  set the state of the MT to "active" in T_i's Token;
5  let P_n = {P-MN_1, ..., P-MN_n} the set of all P-MNs;
6  let C_m = {CO_1, ..., CO_m} the set of all preselected COs;
7  let L = ∅ the ID list of all P-MNs which sent "Yes" vote to
   the CO;
8  start executing the received execution fragment if CO is a
   P-MN;
9  while waiting for lifetime to expire do
10     if CO is a P-MN then
11        broadcast periodically own ID;
12     else                                  /* The CO is a MN-Ag */
13        if value of E_t and/or S_t (initial or extended values) of
          one of the P-MNs is received then
14           update lifetime value only if it needs to be
             increased;
15           update the Token of T_i with the received
             value(s);
16        end
17     end
18     if CO decides to abort T_i or receives "No" vote then
19        abort T_i;
20        send Abort decision to all P-MNs in P_n;
21        exit;
22     end
23     if CO decides to commit T_i or receives updates from
       corresponding P-MN then
24        add own ID to L if CO is a P-MN or add ID of
          corresponding P-MN if CO is a MN-Ag;
25        checkList(L);
26     end
27     switch message M is received do
28        case M is a "Yes" vote from a P-MN
29           send Ack to P-MN;
30           add ID of sending P-MN to L;
31           checkList(L);
32        endsw
33        case M is a beacon from another CO
34           compare the received ID with the own ID;
35           if (both COs are either P-MNs or MN-Ags and
             Own ID > received ID) or (CO is MN-Ag and
             other CO – from which the beacon is received – is
             a P-MN) then
36              send request to CO asking for list L
                (include own list L in the request);
37           else
38              send own list L;
39              change role to normal P-MN or MN-Ag;
40           end
41        endsw
42        case M is a request to send list L
43           send own list L;
44           change role to normal P-MN;
45        endsw
46        case M contains a list L from another CO
47           send own list L if not already done;
48           add all IDs of received L to own list;
49           checkList(L);
50        endsw
51        case M is a Commit decision
52           commit T_i; exit;
53        endsw
54        case M is an Abort decision
55           abort T_i; exit;
56        endsw
57     endsw
58 end
59 abort T_i;          /* T_i is aborted if lifetime expires
   before reaching a decision */
60 send Abort decision to all P-MNs in P_n;
```

Core phase. Each time a CO election is performed, the new elected CO checks whether its list contains all P-MNs of the MT (Algorithm 3). If this is the case, it sets the state of the MT to "precommitted" and starts a 2PC session to collect the votes from P-FNs if any. If the CO receives a "Yes" vote from all the P-FNs, it decides to commit the transaction and sends Commit decision to all the participants. If it receives at least one "No" vote (or no reply) it aborts the transaction and sends Abort decision to all participants. Recall here that our CO selection and election strategies result in that the remaining final CO has access to the infrastructure (if at least one P-MN has access to the infrastructure which is a condition to initiate a meaningful generic transaction). If all P-MNs voted for committing the MT and only one CO remains for the MT, then this unique remaining CO might have a list that does not contain the IDs of all P-MNs because some votes were lost or the corresponding P-MN did not send any vote due to a transient MN failure or communication failure. If the CO does not collect all required votes before the transaction lifetime expires, the MT is aborted and the core phase is subsequently not started at all. P-MNs share the final decisions on encounter. The final decision is inherently replicated onto the CO that turned to either a P-MN or MN-Ag since the lists of the COs are exchanged (Algorithm 2, L 36 and 47) before electing a new CO among them. This replication is needed to recover from a failure of the last remaining CO.

Alg. 3: CheckList Procedure

```
1  procedure checkList(L)
2    if (L contains the IDs of all P-MNs) and (commit set
     contains P-FNs) then
3      set the state of the MT to "pre-committed" in T_i's
       Token;
                          /* Starting of the Core Phase */
4      start a 2PC protocol to collect the votes from all
       P-FNs;
5      if all votes were "Yes" then
6        commit T_i;
7        set the state of the MT to "committed" in T_i's
         Token;
8        send Commit message to all members of the
         commit set;
9        return;
10     else        /* at least one of the votes is No */
11       abort T_i;
12       set the state of the MT to "aborted" in T_i's
         Token;
13       send Abort to all members of the commit set;
14       return;
15     end
16   else if (L contains the IDs of all P-MNs) then
17     commit T_i;
18     send Commit decision to all P-MNs in P_n;
19     exit;
20   end
21   return;
```

Our proposed approach reduces the transaction decision time. Consequently, the resource blocking time of participants is reduced as the COs have bounded waiting time given by the transaction lifetime for the MT outcome. If the transaction lifetime expires at one CO before reaching a final decision, the MT is aborted. This is not viable in any existing solution as P-MNs should be able to asynchronously reach a final decision or proceed with the core phase if P-FNs exist.

4.4 Protocol Operations: Activities of MN-Ags

Precommit phase. Algorithm 4 describes how an MN-Ag interacts with the corresponding P-MN to 1) cope with its disconnections, and 2) to manage its commit state. Upon receiving the execution fragment of its corresponding P-MN from a CO, the MN-Ag forwards it to the corresponding P-MN. After receiving E_t and S_t from the P-MN, the MN-Ag forwards this information to the CO. After receiving a "Yes" or "No" vote from the P-MN, the MN-Ag forwards the vote to the CO. Upon receiving the decision from the CO, the MN-Ag forwards it to the P-MN as soon as it is available (connected to the network). After receiving the Ack for decision reception from the P-MN, the MN-Ag acknowledges the CO. It is key to mention that the MN-Ag is not an active participant in the execution of the MT, since it does not have to know any information about the application and does not need to process any part or fragment of the MT.

Alg. 4: MN-Ag's Activities in GMTC

```
1  wait for receiving execution fragment e_i(P-MN) of the
   corresponding P-MN from CO;
2  create a Token for e_i(P-MN);
3  set the state of e_i(P-MN) to "idle" in T_i's Token;
4  send an estimation of the timeouts of corresponding P-MN
   to the CO;
5  for any received message do
6    if message contains the timeouts of the P-MN then
7      update T_i's Token with the received timeouts;
8      set the state of the e_i(P-MN) to "active" in T_i's
       Token;
9      forward the timeouts to the CO;
10   else if message contains the updates of the corresponding
     P-MN then
11     update the Token with the received updates;
12     send "Yes" vote to CO;
13   else if message contains possible disconnection of the
     corresponding P-MN and its reasons then
14     recompute the timeouts based on disconnection
       reasons;
15     update the token with this information;
16     send extended timeouts to the CO;
17   else if message is sent by the CO then
18     update the Token with the received message;
19     send the received message to the corresponding
       P-MN as soon as it is available;
20   else if message is sent by P-MN then
21     update the Token with the received message;
22     send the received message to CO;
23
24 end
```

The MN-Ag can take some decisions on behalf of its corresponding P-MN. These decisions include the extension of the timeouts of the P-MN in case of a transient disconnection. The MN-Ag is also given the responsibility to send an estimation of the timeouts of the corresponding P-MN direct after receiving the execution fragment of this P-MN (L 4). This estimation can be corrected after receiving new timeout values (E_t and S_t) from the P-MN.

4.5 Protocol Operations: Activities of P-FNs

Core phase. P-FNs behave as per the established 2PC protocol, i.e., a P-FN executes its fragment, waits for the Prepare message, sends its vote and waits for the decision. Upon receiving the decision, the P-FN acknowledges the CO. Note that any existing protocol like 3PC or Paxos Commit can be used here.

5 Correctness Basis

To show the correctness of the proposed GMTC protocol composed of Algorithm 1, 2, 3, and 4, we demonstrate that it satisfies the required five *atomicity properties* [6]:

1. *Stability.* A participant cannot reverse its decision after it has reached one.
2. *Consistency.* All participants that reach a decision reach the same one.
3. *Validity.* The "Commit" decision can only be reached if *all* participants voted "Yes."
4. *Nontriviality.* If no failure occurs and all participants voted "Yes," then the final decision should be "Commit."
5. *Termination.* At any point in execution, if all existing failures are repaired and no new failures occur for sufficiently long time, then all participants will eventually reach a decision.

While the properties of stability and nontriviality naturally follow from the GMTC protocol description in Section 4, we now show that it also satisfies the consistency, validity and termination properties.

Consistency. This is satisfied as only the last active CO decides the outcome of the transaction in case the final decision is "Commit" and distributes the final decision to every participant. Hence, the last remaining CO is the single one which can have the final eventual complete list of P-MNs since at least its vote was not communicated to any other CO or P-MN according to the specification of the GMTC protocol. This CO is the single one able to start the core phase. If more than one CO are still remaining in the system, they can only take an "Abort" decision and no "Commit" since no one of them can have a full list of participating nodes (see detailed description of GMTC protocol). Thus, the consistency property is guaranteed by our protocol.

Validity. We assume that one of the preselected COs decides to commit the transaction when at least one participant has not decided yet. If this participant is a P-MN then its ID cannot appear in any list L (Algorithm 2, line 7) of the preselected COs according to the specification of the GMTC protocol. Obviously, no preselected CO can then take the decision to precommit the MT since this contradicts with the protocol specification (Algorithm 3, lines 1-21). In the case that at least one of the P-MNs decides to abort the transaction, the preselected COs cannot decide to precommit the transaction because this decision will violate the protocol specification (Algorithm 2, lines 18-22). If this participant is a P-FN, then the 2PC protocol decides to commit the transaction before receiving all the votes from the P-FNs, which again contradicts the specification of the 2PC protocol. In the case that at least one of the P-FNs decides to abort the transaction, the CO cannot decide to commit the whole transaction because this decision will also violate the 2PC protocol specification. Hence, the commit decision can only be reached if all P-MNs voted "*Yes*," i.e., decided to commit the transaction.

Termination. We consider any execution containing the failures listed in the perturbation model detailed in Section 2.2. From the GMTC protocol specification, we can observe that because we are using a timeout concept the protocol cannot block forever (the blocking of the protocol forever leads to a nontermination of the protocol). If at any point in execution all existing failures are repaired and no new failures occur for sufficiently long time, then all participants will eventually reach a decision. Especially in this situation all P-MNs (including COs) can meet each other eventually and progressively the lists of COs are filled and the number of COs is reduced until only one CO remains having a list L containing the IDs of all P-MNs. This complete list allows this CO to take a precommit decision (Algorithm 3, lines 1-21) and to start the core phase which is executed only on the wired part of the environment. The protocol terminates as soon as 2PC reaches a final decision (2PC also implements a timeout strategy to avoid blocking). If the lifetime expires at any CO before reaching the final decision, the MT is aborted (Algorithm 2, lines 59-60) leading also to the termination of the protocol.

6 Performance Evaluation

We use simulations to validate our approach. We present the used performance metrics, the simulation model and our results on the high commit rate, the bounded decision time and the efficiency of GMTC. Our simulation studies show the feasibility of our approach and concentrate on investigating the benefit of accessing the infrastructure in MT, where only MNs are participating in their execution.

6.1 Methodology and Simulation Settings

For the evaluation of the GMTC protocol, we focus on three major performance metrics: 1) *Commit rate* as it determines the service availability, 2) commit latency or *transaction decision time* as it determines the service response time, and 3) *message complexity* as it determines the scalability and efficiency of our approach. We measure the commit rate as the ratio of number of successfully committed MTs to total number of initiated MTs. The transaction decision time is the time needed to take a decision about the outcome of the initiated MT, i.e., the time between the initiation of the MT and the time, when the final decision is reached at the CO. The blocking time of P-MNs is majorally determined by the transaction decision time. However, the time needed for the final decision to reach the P-MNs plays a role especially in mobile ad hoc and hybrid environments (blocking time = decision time + time needed to disseminate the final decision). This time depends on the partitioning degree and the implementation of the message dissemination protocols used to disseminate the final decision [19], and therefore will not be further investigated in our performance evaluation. The message complexity of GMTC is defined as the number of wireless messages sent and received in average by each P-MN during the execution of the MT.

The performance of the GMTC approach is evaluated in this work based on the service delivery level assured by the protocol and defined basically by the commit rate and the decision time. The costs of assuring a certain service delivery level are measured in terms of message complexity. We focus in our performance evaluation on the impact of BS coverage and network partitioning degree of MNs

TABLE 2
Simulation Settings

Parameter	Value(s)
Geographical area	$2km$ x $2km$
Communication range	$250m$
Mobility model	Random Waypoint (RWP)
Node speed	uniform in [0.5, 1.5] m/s
# Nodes	$\in$ [20,200]
# Pre-selected COs	$\in \{3,5,10\}$
# P-MNs	10
Lifetime	$\in \{60s,120s,300s\}$

that can only communicate in ad hoc manner on the identified performance metrics.

For our simulation studies we have used J-Sim [18], a component-based, compositional simulation environment that is developed in Java. For the performance evaluation of the GMTC protocol, we consider a representative range of parameter values. Table 2 summarizes our simulation settings. We selected the commonly used Random Waypoint mobility model [12] (node speed uniform in [0.5, 1.5] m/s). We fix the mobility area (2 km $\times$ 2 km) and the communication range (250 m). We generate the mobility scenarios using the BonnMotion mobility simulator [8]. Given its importance, for all our simulation studies we vary the partitioning degree through varying the number of nodes. The partitioning degree or degree of separation is provided by BonnMotion and reflects how likely it is that two randomly chosen nodes are not within the same partition at a randomly chosen point in time.

BSs have in our simulations the same communication range as the MNs. We place the BSs uniformly in the simulated area and vary the number of deployed BSs to vary the coverage area of these BSs. For 4, 9, 16, 25, and 36 deployed BSs the percent of the simulated area covered by these BSs is 19.63, 44.17, 78.53, 98.2, and 100 percent, respectively.

We consider that all deployed MNs in the simulation area can communicate with the BSs and also with other MNs (in ad hoc mode) if they are in their communication range. We generate transactions of similar number and size of execution fragments. We initiate one transaction at the beginning of each simulation. We fix the number of P-MNs to 10 and vary the number of preselected COs, the transaction lifetime and the number of BSs to study the impact of these parameters on the performance of GMTC. Each simulation is repeated 200 times for statistical significance of the results.

Besides the detailed qualitative comparison of GMTC to FT-PPTC and ParTAC throughout the paper, we show the best case performances of these protocols in our simulation results. For this, we note that in case the coverage of BSs is 0 percent, the GMTC protocol converges to the behavior of ParTAC, and the performance of GMTC and ParTAC are identical. When the coverage of BSs is 100 percent, the behavior of GMTC converges to that of FT-PPTC with a more than preselected coordinator, which can immediately agree with negligible delay and message overhead on one single coordinator. Accordingly, we consider that the performance of GMTC is almost the same as the of FT-PPTC for full base station coverage. In [1], a comprehensive comparison of FT-PPTC to 2PC has been provided, which we skip due to space limitation.

6.2 Simulation Results

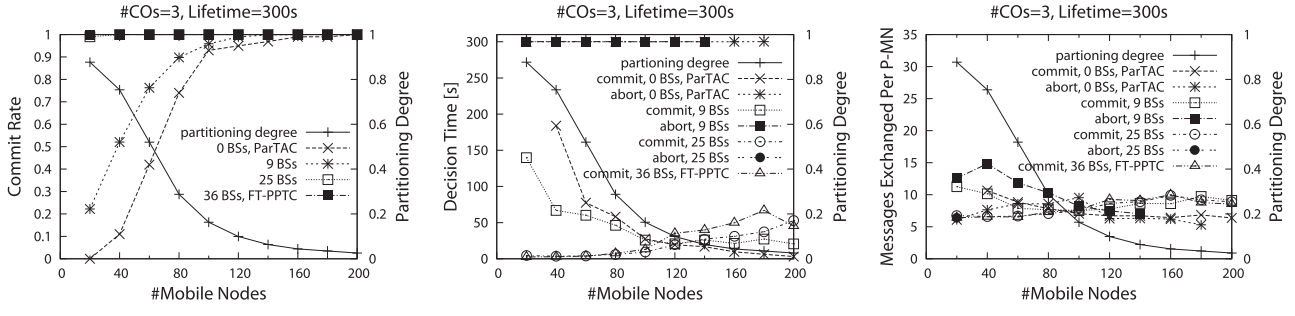
Now, we present the results of our conducted simulation studies for the defined performance metrics. As mentioned before, we simulate GMTC under different network conditions and vary important protocol parameters to study the behavior of our approach in a wide range of possible deployment scenarios. Overall, we split the statistics for "Abort" and "Commit" cases to have better insights to GMTC performance.

6.2.1 Impact of BSs' Coverage and Node Density

We arbitrarily use in this study three preselected COs and a transaction lifetime of 300 s. To assess the influence of the BSs' coverage area on the GMTC protocol, we vary the number of the BSs deployed in the simulation area (i.e., we vary the percentage of the simulated area covered by these BSs).

Fig. 6a shows that GMTC benefits from accessing BSs to increase the number of committed transactions compared to ParTAC. The infrastructure indeed allows to bridge partitions. For 36 BSs no transaction is aborted. Accordingly, we focus on statistics for the "Commit" case in the remainder of this section.

In Fig. 6b, we observe also that accessing the infrastructure reduces the decision time, especially in sparse deployments (less than 100 MNs in our settings). This is due to the fact that bridging partitions through the infrastructure is much faster than bridging partitions through node mobility like it is the case in ParTAC. For dense node deployments



(a) Commit rate (b) Decision time (c) Message complexity

Fig. 6. Impact of BSs' coverage on (a) commit rate, (b) decision time, and (c) message complexity.

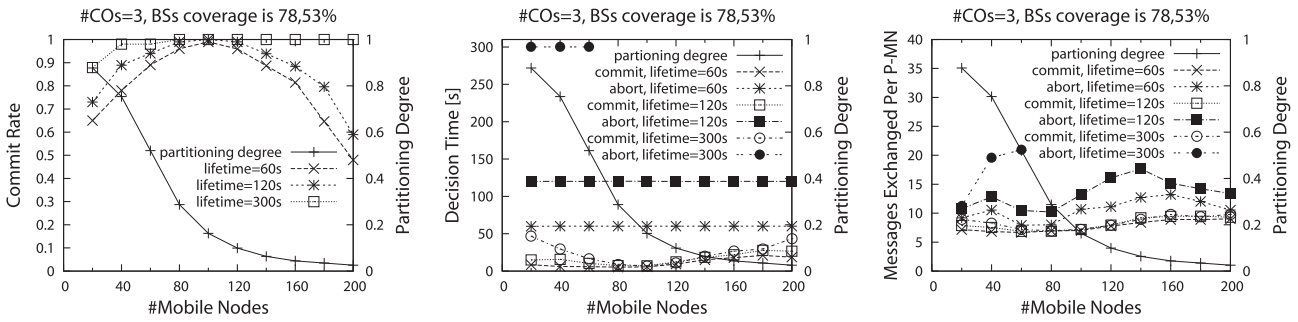


Fig. 7. Impact of transaction lifetime on (a) commit rate, (b) decision time, and (c) message complexity.

(more than 100 MNs), the commit decision time surprisingly increases. We explain this behavior as follows: If the number of MNs increases, the number of MNs sharing the access to a BS also increases. Consequently, the infrastructure-based communication suffers from higher latencies.

Apart from a few "Abort" cases when the partitioning degree is very high and resulting in exchanging more messages, the overhead of the MTs in term of exchanged wireless messages remains more or less constant in our simulations (see Fig. 6c).

The curves "abort, 0 BSs," "abort 9 BSs," and "abort 25 BSs" in Figs. 6b and 6c have less measurement points than the others. This is due to the fact that when commit rate as shown in Fig. 7a becomes 100 percent, the number of aborted transactions is 0, which translates in 0 s decision time and 0 exchanged messages per P-MN. For these cases, we do not plot the values for readability reasons.

6.2.2 Impact of Transaction Lifetime

We arbitrarily fix in this scenario the number of preselected COs to 3 and number of deployed BSs to 16 (i.e., 78.53 percent of the simulated area is covered by the BSs). To assess the impact of the lifetime on the GMTC performance, we select the following lifetime values: 60 s, 120 s, and 300 s.

Fig. 7a shows that also in the case of GMTC, the commit rate depends on the MT lifetime. The commit rate increases with the lifetime value. If the number of deployed MNs in the simulated area is larger than 100, the commit rate starts even to decrease because the decision time starts to increase as illustrated in Fig. 7b and as explained before in Section 6.2.1.

Fig. 7c highlights that the number of exchanged messages in the "Abort" case increases if the lifetime increases since an increase in the lifetime implies an increase in the number of exchanged messages (more votes and lists can be sent to encountered COs if lifetime increases). For the "Commit" case the message complexity does not change if the lifetime is changed.

Similar to Fig. 6, the curves "abort, lifetime $= 300$ s" in Figs. 7b and 7c do not show measurement points for the cases, where all transaction are committed.

6.2.3 Impact of Number of Preselected COs

We arbitrarily fix in this scenario the lifetime to 300 s and the number of deployed BSs to 9 (i.e., 44.17 percent of the simulated area is covered by the BSs) and vary the number of preselected COs.

The number of preselected COs does not impact the commit rate of GMTC as illustrated in Fig. 8a. This is due to the fact that as soon as two COs encounter each other only one of them remains active and the other one becomes a normal P-MN. After a certain point in time only a few (2 to 3) COs remain and all the simulated scenarios behave from this instant onwards similarly. This point in time is closer to the initiation time of the MT in the "Commit" case as from all the COs present in one partition only one remains active as soon as they receive beacons from each other. Fig. 8b shows that the number of preselected COs does have only a slight impact on the decision time also because of the same reasons given above.

However, the number of preselected COs has a minor impact on the efficiency of the GMTC protocol as shown in Fig. 8c. The slight increase of the number of messages

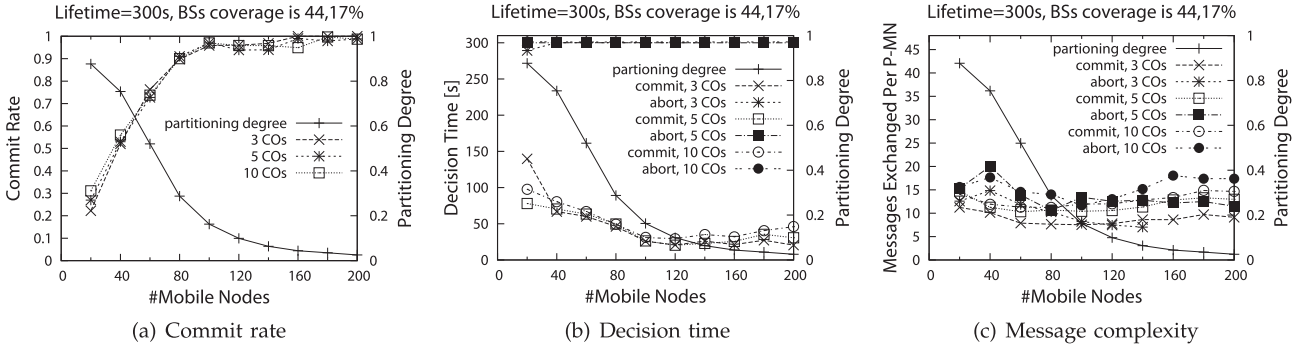


Fig. 8. Impact of number of preselected COs on (a) commit rate, (b) decision time, and (c) message complexity.

exchanged per node is due to the fact that every P-MN needs to send its vote to more COs as the number of COs increases. It is noteworthy to mention that selecting higher number COs is primarily to tolerate CO failures during the MT execution. Our simulations show that a higher CO failure-tolerance does only slightly impact the message efficiency.

6.3 Discussion

Our simulation studies show that the GMTC approach takes advantage of the access to the infrastructure whenever possible to achieve better performance especially w.r.t. the transactional service availability by increasing the commit rate of initiated MTs and w.r.t. commit latency by reducing the commit decision time of these transactions. Similar to FT-PPTC and ParTAC, GMTC takes also into consideration application requirements by defining an appropriate lifetime for each initiated MT. Our evaluation studies has shown the existence of a tradeoff between the chosen lifetime and the performance of GMTC in terms of commit rate, decision time and message complexity. By defining an appropriate lifetime, the application also limits and controls the cost of the initiated MTs. To choose the appropriate lifetime, the application should consider 1) the user tolerable transaction decision delay, and 2) the time scale to transitively full-connect the network mainly determined by the partitioning degree and the mobility of nodes. The time scale factor can for instance be estimated from the experienced decision time of prior transactions.

7 Related Work

Given the need for correct data management in mobile environments, MTs have increasingly become the focus of extensive ongoing research. Some existing atomic commit protocols are designed for infrastructure-based mobile environments [7], [20], [26], [22], [1]. Unilateral commit for mobile (UCM) [7] provides support for disconnections and offline executions on mobile devices. UCM is a one-phase protocol where the voting phase of 2PC is eliminated to reduce the wireless message complexity. The single CO acts as a "dictator" imposing its decision on all participants. UCM guarantees atomicity. However, UCM is based on strict and hard assumptions of local pessimistic concurrency control such as strict two-phase locking [6] which is required for all participants, as well as immediate integrity control and homogeneity of participating database systems. Transaction commit on timeout (TCOT) [20] uses timeouts to provide a nonblocking protocol that limits the amount of communication between the participants during the execution of the protocol. Instead of exchanging messages to reach a final decision, the single CO waits for timeouts to expire. Overall, TCOT provides only semantic atomicity as defined in [14]. Semantic atomicity requires the existence of a compensating transaction for every initiated mobile transaction which is not possible for every transaction. Compensating transactions undo semantically the transaction effects. This type of atomicity is weaker than the strict atomicity [17] needed for transactions in general, which limits the applicability of TCOT to a limited class of applications. The CO2PC protocol [27], [26] combines an optimistic approach with 2PC. Like TCOT, the CO2PC protocol provides only semantic atomicity limiting the applicability of the protocol only for a restricted set of applications. In mobile two-phase commit (M-2PC) [22], a P-MN delegates its commit duties to its agent on an FN, which is the BS the P-MN is connected to. Unfortunately, M-2PC assumes that all P-MNs are connected at transaction initiation and that network disconnections are allowed only after the mobile node delegates its commitment duties. Similar to our discussion of the fault-tolerant prephase transaction commit (FT-PPTC) protocol [1] in Section 3.2, all these infrastructure-based protocols are based on single CO in the wired network, which makes them unsuitable for ad hoc environments and therefore for hybrid environments.

Other works address the atomic transaction commit problem in mobile ad hoc networks [29], [9], [10], [23]. In [23], a cross layer commit protocol for mobile ad hoc networks (CLCP) is presented. This protocol employs all participants as COs and uses consensus to ensure failure tolerance. CLCP is directly instantiated from the application layer, but operates on both network and application layers. Consensus introduces a considerable message overhead which makes it undesirable especially on mobile devices. Bose et al. [9] and Böttcher et al. [10] propose the use of a cluster of COs preferably in single-hop distance from each other to avoid blocking of mobile nodes in case one CO fails. The cluster of COs elects a single main CO and uses the 3PC protocol [28] to agree on a consistent decision either to commit or abort the MT. If the cluster of COs is partitioned or the main CO fails the authors use a termination protocol based on the Paxos Consensus protocol [21] to elect a new main CO. The assumption in [9] and [10] that the COs are moving together in a group (forming a one hop cluster) is not valid in most of mobile scenarios. In [29], a commit solution is presented which assumes that every mobile node in a partition knows all the members of the partition it belongs to. However, this solution is briefly sketched and lacks a detailed description of the proposed group based commit protocol. Given the partition membership information, every partition elects a leader and uses the 2PC protocol inside the partition to decide whether the transaction should be tentatively committed or aborted. This temporary decision is communicated to all mobile nodes within the partition. When a mobile node joins a new partition, the tentative decision (obtained in its original partition) is communicated to the new partition. As described in [29], the correctness of the proposed solution is assured by the partition membership assumption, i.e., the fact that partitions can be detected. The assumption that every MN in a partition knows all the members of its partition is crucial for mobile environments. Some works [24], [11] addressed the problem of group membership in ad hoc environments, however, a generic solution remains a challenge. We showed in Section 3.2 the unsuitability of the partition-tolerant atomic commit (ParTAC) protocol [4] for hybrid environments. Similarly, all the above discussed infrastructure-less protocols are inefficient (extra overhead and long blocking time of FNs) in infrastructure-based environments and therefore fail in delivering a solution for hybrid environments.

8 Conclusion

As the evolving mobile environments necessitate new commit constraints, the current approaches geared toward dedicated scenarios, often do not provide comprehensive and generic commit capabilities. Thus, our developed generic and evolvable atomic commit solution benefits from the presence of an infrastructure, if available, and delivers best effort results in its absence. We have introduced the main challenges for designing atomic commit protocols faced in hybrid mobile environments. GMTC is developed as an efficient perturbation-resilient commit protocol that provides strict atomicity in spite of frequent mobile environment perturbations. GMTC especially fills the gap between solutions provided for infrastructure-based and infrastructure-less mobile environments.

References

[1] B. Ayari et al., "FT-PPTC: An Efficient and Fault-Tolerant Commit Protocol for Mobile Environments," *Proc. IEEE 25th Symp. Reliable Distributed Systems (SRDS),* pp. 96-105, 2006.

[2] B. Ayari et al., "Delay-Aware Mobile Transactions," *Proc. Sixth IFIP WG 10.2 Int'l Workshop Software Technologies for Embedded and Ubiquitous Systems (SEUS),* pp. 280-291, 2008.

[3] B. Ayari et al., "Exploring Delay-Aware Transactions in Heterogenous Mobile Environments," *J. Software,* vol. 4, no. 7, pp. 634-643, 2009.

[4] B. Ayari et al., "ParTAC: A Partition-Tolerant Atomic Commit Protocol for MANETs," *Proc. 11th Int'l Conf. Mobile Data Management (MDM),* pp. 135-144, 2010.

[5] B. Ayari et al., "On the Design of Perturbation-Resilient Atomic Commit Protocols for Mobile Transactions," *ACM Trans. Computer Systems,* vol. 29, no. 3, pp. 7:1-7:36, 2011.

[6] P.A. Bernstein et al., *Control and Recovery in Database Systems.* Addison-Wesley, 1987.

[7] C. Bobineau et al., "A Unilateral Commit Protocol for Mobile and Disconnected Computing," *Proc. 12th Int'l Conf. Parallel and Distributed Computing Systems (PDCS),* 2000.

[8] *BonnMotion,* http://iv.cs.uni-bonn.de/wg/cs/applications/bonnmotion/, 2013.

[9] J. Bose et al., "An Integrated Commit Protocol for Mobile Network Databases," *Proc. Ninth Int'l Database Eng. and Application Symp. (IDEAS),* pp. 244-250, 2005.

[10] S. Böttcher et al., "A Failure Tolerating Atomic Commit Protocol for Mobile Environments," *Proc. Int'l Conf. Mobile Data Management (MDM),* pp. 158-165, 2007.

[11] L. Briesemeister and G. Hommel, "Localized Group Membership Service for Ad Hoc Networks," *Proc. Int'l Workshop Ad Hoc Networking (IWAHN),* pp. 94-100, 2002.

[12] J. Broch et al., "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proc. ACM MobiCom,* pp. 85-97, 1998.

[13] S.B. Davidson et al., "Consistency in a Partitioned Network: A Survey," *ACM Computing Surveys,* vol. 17, no. 3, pp. 341-370, 1985.

[14] H. Garcia-Molina, "Using Semantic Knowledge for Transaction Processing in a Distributed Database," *ACM Trans. Database Systems,* vol. 8, no. 2, pp. 186-213, 1983.

[15] J. Gray, "Notes on Data Base Operating Systems," *Proc. Operating Systems, An Advanced Course,* pp. 393-481, 1978.

[16] T. Härder and A. Reuter, "Principles of Transaction-Oriented Database Recovery," *ACM Computing Surveys,* vol. 15, no. 4, pp. 287-317, 1983.

[17] T. Härder and A. Reuter, *Principles of Transaction-Oriented Database Recovery.* Morgan Kaufmann, 1994.

[18] "The J-Sim Website," *J-Sim,* http://www.j-sim.org/, 2013.

[19] A. Khelil, "A Generalised Broadcasting Technique for Mobile Ad Hoc Networks," PhD thesis, Univ. of Stuttgart, 2007.

[20] V. Kumar et al., "TCOT-A Timeout-Based Mobile Transaction Commitment Protocol," *IEEE Trans. Computers,* vol. 51, no. 10, pp. 1212-1218, Oct. 2002.

[21] L. Lamport, "The Part-Time Parliament," *ACM Trans. Computer Systems,* vol. 16, no. 2, pp. 133-169, 1998.

[22] N. Nouali et al., "A Two-Phase Commit Protocol for Mobile Wireless Environment," *Proc. 16th Australasian Database Conf. (ADC),* pp. 135-143, 2005.

[23] S. Obermeier et al., "CLCP-A Distributed Cross-Layer Commit Protocol for Mobile Ad Hoc Networks," *Proc. IEEE Int'l Symp. Parallel and Distributed Processing with Applications (ISPA),* pp. 361-370, 2008.

[24] G.C. Roman et al., "Consistent Group Membership in Ad Hoc Networks," *Proc. 23rd Int'l Conf. Software Eng. (ICSE),* pp. 381-388, 2001.

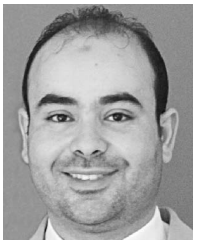
[25] G. Samaras et al., "Two-Phase Commit Optimizations in a Commercial Distributed Environment," *Distributed Parallel Databases,* vol. 3, no. 4, pp. 325-360, 1995.

[26] P. Serrano-Alvarado, "Transactions Adaptables Pour Les Environments Mobiles," PhD thesis, Université J. Fourier, 2004.

[27] P. Serrano-Alvarado et al., "Context Aware Mobile Transaction," *Proc. IEEE Int'l Conf. Mobile Data Management (MDM),* p. 167, 2004.

[28] D. Skeen and M. Stonebraker, "A Formal Model of Crash Recovery in a Distributed System," *IEEE Trans. Software Eng.,* vol. 9, no. 3, pp. 219-228, May 1983.

[29] W. Xie, "Supporting Distributed Transaction Processing Over Mobile and Heterogeneous Platforms," PhD thesis, Georgia Inst. of Tech., 2005.



Brahim Ayari received the PhD degree in computer science from Technische Universität Darmstadt, Germany. He is currently with the Information Systems Department at ABB.

Abdelmajid Khelil is a research team leader in the Department of Computer Science, Technische Universität Darmstadt. He has recently joined the Huawei European Research Center as a senior researcher.

Neeraj Suri is a professor it the Department of Computer Science, Technische Universität Darmstadt. His professional details can be found at http://www.deeds.informatik.tu-darmstadt.de.