Exploring Delay-Aware Transactions in Heterogeneous Mobile Environments

Brahim Ayari, Abdelmajid Khelil, Neeraj Suri
Technische Universität Darmstadt,
Dependable, Embedded Systems and Software Group,
Hochschulstr. 10, 64289 Darmstadt, Germany
Email: {brahim,khelil,suri}@informatik.tu-darmstadt.de

***Abstract*— In the expanding e-society, mobile embedded systems are increasingly used to support transactions such as for banking or database applications. Such systems entail a range of heterogeneous entities - both the devices and the networks connecting them. While these systems are exposed to frequent and varied perturbations, the support of atomic distributed transactions is still a fundamental requirement to achieve consistent decisions. Guaranteeing atomicity and high performance in traditional fixed wired networks is based on the assumption that node and link failures occur rarely. This assumption is often not supported in current and upcoming mobile environments where the heterogeneity and mobility often result in link and node failures as a dominant operational scenario.**

In order to continue guaranteeing strict atomicity while providing for high efficiency (low resource blocking time of transaction participants and message overhead) and acceptable commit rate, transactional fault-tolerance techniques need to be revisited perhaps at the cost of transaction execution time. In this paper, we provide a comprehensive classification of perturbations for a wide range of mobile environments including infrastructure-based, ad-hoc, and hybrid environments. We also investigate the impact of these perturbations on the design of mobile transactions. In particular we argue for the delay-awareness of mobile transactions to allow for the fault-tolerance mechanisms to ensure resilience to the various and frequent perturbations.

***Index Terms*— Transactions, mobile database systems, dependability**

I. Introduction

Mobile embedded systems are increasingly characterized by frequent and varied perturbations in the mobile devices and the networks linking them. These are directly apparent as resource constraints and operational failures over the delivery of mobile services. Mobile systems are constrained by the processing, storage and energy capacity of mobile devices, and also the continuously varying properties of wireless channels. Most of the failures which occur in such systems manifest at the nodes (given their mobility and size) or as communication failures. These failures can last for seconds, minutes or even hours e.g., network partitioning. Increasingly, the mobile environments are supporting applications that require strict atomicity. A typical example being banking/stock transactions in the m-commerce applications. Atomic commit protocols ensure strict atomicity of database transactions and consequently play a major role for the design of these applications. Most existing atomic commit protocols show a restricted perturbation-tolerance resulting in either poor transaction commit rate or high resource blocking time which consequently decreases the efficiency of the mobile system. In [1], we showed that sacrificing latency (time needed to decide about the outcome of the transaction) is necessary to cope with frequent perturbations without sacrificing performance in terms of efficiency and commit success rate.

Our Contributions and Paper Organization

Mobile transactions are increasingly the focus of extensive ongoing research. A variety of transaction models have been proposed such as [2]–[10] with an excellent survey appearing in [11]. In the literature database transactions are usually delay-sensitive. A limited body of research exists for real-time transactions [12], [13]. However, to the best of our knowledge, delay-aware (i.e. also delay-tolerant) transactions have not yet been addressed.

In this paper we argue for the necessity of delay-awareness of mobile transactions in networked embedded systems. Our work in [1] investigated primarily infrastructure-based system models. In this paper, we extend this core model to cope with a more generalized mobile system that also involves ad-hoc communication scenarios. Ad-hoc mobile environments are mainly deployed when an infrastructure is unavailable. For instance ad-hoc networks are used to support crisis management applications, such as in disaster recovery, where the entire network is destroyed and communication between different entities should be set up within few hours. Increasingly ad-hoc communication is used to supplement an existing infrastructure for example to reduce communication expenses. In this paper we briefly summarize our previous work tailored for infrastructure-based mobile environments. Next we present a framework to support

This paper is based on "Delay-Aware Mobile Transactions," by B. Ayari, A. Khelil, and N. Suri, which appeared in the Proceedings of the 6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS) , Capri Island, Italy, October 2008. © 2008 Springer.

Research supported in part by EU COMIFIN, INSPIRE and DFG GRK 1362 (TUD GKMM).

doi:10.4304/jsw.4.7.634-643

strict atomicity in pure ad-hoc scenarios. Finally we provide preliminary integration efforts for providing atomic transactions in generalized mobile environments.

The remainder of this paper is organized as follows. In Section II, the system model is described along with a set of application scenarios and a classification of perturbations. The design requirements for mobile transaction protocols and systems are presented in Section III. In Section IV, delay-aware mobile transactions are introduced along with a discussion of the main challenges of introducing delay-awareness in mobile systems. Section V concludes the paper and briefly outlines future issues.

II. System Model, Scenarios and Perturbations

We first present the system model of the generalized mobile distributed environment. Next we describe application scenarios for upcoming embedded and ubiquitous systems where strict atomicity is essentially desired. Finally, we identify the relevant perturbations, i.e., constraints and failure modes that can occur in the considered environment to affect commit functionality.

A. System Model

In order to consider a broad class of mobile and networked embedded systems, we develop a generalized mobile distributed environment consisting of a set of battery-powered mobile hosts (MH), a set of fixed hosts (FH) and a set of Wireless Sensor Networks (WSNs) composed of a number of sensor nodes (SN) and mobile sink(s). The mobile sink collects data from SNs about a monitored area or goods etc. We assume that every host in this environment has a unique ID. The architecture of the environment considered is illustrated in Fig. 1. The MHs intermittently connect to the wired network through *Mobile Support Stations (MSS)* via wireless channels (Fig. 1). The MHs can communicate directly with each other in an ad-hoc manner for instance using Bluetooth or WLAN. The coverage of MSSs is much higher than the transmission area of ad-hoc communication technologies (e.g., if we compare GSM to Bluetooth). A subset of MHs can also communicate with mobile sink(s) of involved WSNs. This generalized mobile distributed environment mainly consists of three basic system models which are often considered discretely by commit protocol developers. In this work we progressively tackle the complexity of the generalized system model by stepwise considering these sub-systems and subsequently consolidating them into the proposed generalized system model:

1) *Infrastructure-based* scenarios involve only FHs, MSSs and a subset of MHs of the model of Fig. 1. This subset of MHs can only communicate with each other or with FHs using the services of MSSs. They are also not able to communicate with each other in an ad-hoc fashion.
2) *Ad-hoc scenarios* involve only a subset of MHs of Fig. 1 and mobile sinks of WSNs. These MHs can communicate with each other or with mobile sinks of WSNs only in ad-hoc manner, i.e. they can not communicate with MSS or other FHs. In this scenario, mobile sinks construct a gateway for communication with WSNs but only mobile sinks (not SNs) can be participants in transactions.
3) *Hybrid scenarios* are a combination of both the infrastructure-based and the ad-hoc scenarios representing our generalized mobile distributed model. In these scenarios, MHs can ad-hoc communicate with each other and also with MSS to reach other MHs or FHs.

We refer to a distributed transaction where at least one MH participates in its execution as a *Mobile Transaction (MT)*. Commit protocols are generally based on the existence of at least one *coordinator (CO)*, which is responsible for coordinating the execution of the corresponding transaction. For different transaction and mobile system models, different nodes may play the CO role. One CO for each transaction in infrastructure-based scenarios may be sufficient and more than one CO in ad-hoc or hybrid scenarios may be needed. This key issue of CO selection will be discussed in more detail in Section IV. The CO is responsible for storing information concerning the state of the transaction execution. Based on the information collected from and about the participants of the transaction, the CO takes the decision to commit or abort the transaction and informs all participants about its decision.

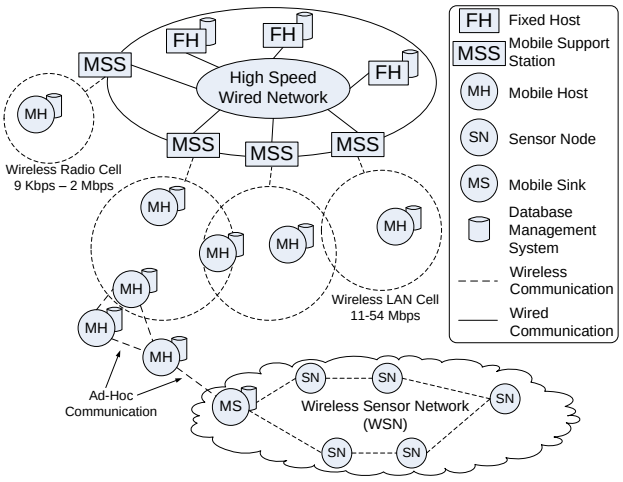


Figure 1. Architecture of environment

B. Application Scenarios

We now classify the main application scenarios for embedded and ubiquitous systems, where strict atomicity for mobile transactions is desired.

a) Bank/Stock Transactions: This type of application scenarios includes mobile commerce (m-commerce) applications where users can buy or sell goods using their mobile devices and involving bank servers in fixed networks to accomplish their transactions. These applications usually run in infrastructure-based scenarios and

increasingly in hybrid scenarios involving some WSNs checking the availability of goods. The role of WSNs in this scenario is to keep track on availability of certain type of goods and their corresponding quantity. Transactions can be in this case processed automatically without human intervention but using an on-demand availability check.

b) Coordination across Autonomous Networked Vehicles: In such an application scenario (representing ad-hoc networks) we present a potential application where mobile transactions are needed for the purpose of coordination for safe navigation of unmanned autonomous networked vehicles. Similar to black boxes for airplanes, autonomous vehicles can be equipped with black boxes which are basically mobile databases. The following example describes a scenario where these black boxes can be useful to deploy in cars also and more precisely in unmanned vehicles. Fig. 2 shows a scenario of four unmanned vehicles at a crossing. These vehicles need to agree on an order how they will pass the crossing. Prior to their actual passing, this order information needs to be agreed upon and recorded atomically to their corresponding black boxes. This information may be needed by insurance companies or police in case an accident occurs.

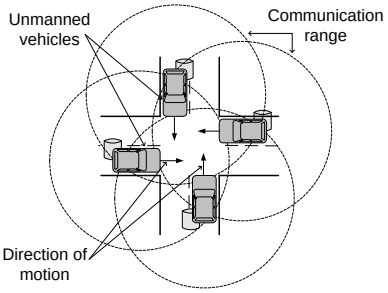


Figure 2. Coordination across autonomous vehicles (livelock scenario)

Other scenarios for ad-hoc environments include payment and mobile commerce services. A detailed description of a commercial mobile ad-hoc application which represents a radio dispatch system between taxis can be found in [14]. The radio dispatch system is described as a novel and plausibly realistic application scenario for mobile ad hoc networks. The proposed system is then evaluated from both financial and technical perspectives to provide a complete picture of its feasibility.

c) Health-Care Ubiquitous Systems: For insurance purposes, in order to monitor old people living alone in their homes a set of WSNs could be deployed in their homes and transactions are needed to react to certain situations where some actuators need to be activated together either all of them or none and this data should also be written somewhere on MHs and/or on FHs belonging to hospitals or police etc. This illustrates a hybrid scenario as defined in our system model.

C. Classification of Perturbations

Within these networked embedded systems supporting such transactional applications, we now consider two main classes of perturbations: Operational constraints (battery power, computing, connectivity etc.) and failures (Fig. 3). We classify the environmental constraints relevant to mobile transactions as heterogeneity (of nodes and links), unstable storage and energy constraints. Failures of the mobile environment are classified into communication and node failures.

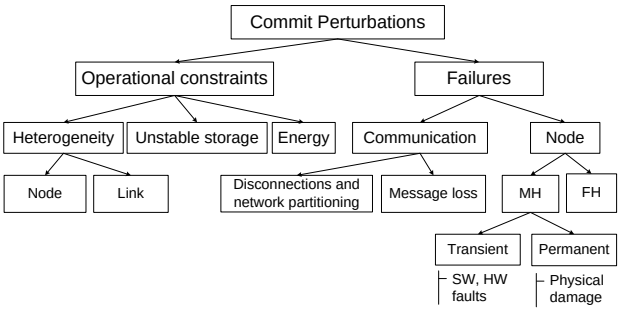


Figure 3. Classification of MT commit perturbations

1) Constraints: The considered mobile environment is constrained mainly by the characteristics of *MHs* and *wireless links*. MHs intuitively possess less computational resources than FHs, such as processor speed and storage capacity. Especially some MHs possess limited disk space which restricts the amount of storable data. These resource constraints increase the time MHs need to execute transaction fragments or may even lead to execution failures. Furthermore, as MHs are carried by users, they incur operational wear and tear, and can also be easily lost or stolen. Two of the most important sources of power consumption for MHs are transmissions and disk accesses [15]. We note that transmitting data consumes around three times as much energy as receiving the same amount of data by a MH. Furthermore, MHs may run in different energy modes or be turned-off to save energy.

Wireless network characteristics are changing more frequently than those of wired links. For example, the effective bandwidth available for MHs over a wireless link is highly dynamic. This depends on the wireless technology (like GSM, GPRS, UMTS, WLAN, and Satellite), access coverage, and number of MHs that have to share the wireless medium. Other key characteristics of the wireless links are higher latency and communication charges. These characteristics lead to considerably varied reliability/availability and connectivity of MHs.

Mobile nodes are considered to have *unstable storage* due to high vulnerability of these entities to catastrophic failures, e.g., loss, theft or physical damage and the immature replication strategies used in the mobile environment to replicate data like in [16].

The limitations and characteristics listed above outline the variation of constraints for the mobile environment as being different from those in fixed networks. These constraints also complicate the design of mobile transaction protocols. For example, aborting a MT because of a slow participant is not a suitable design choice in mobile environments.

2) Failures: We now outline the common failure modes which we classify into primary classes of communication and node failures.

a) Communication Failures: These constitute the majority of failures in the mobile environment. We distinguish between two types of communication failures:

Message Loss: Messages exchanged across MHs or between MHs and MSSs are highly vulnerable to loss due to the high bit error rate of wireless links and from network congestion and collisions. Message loss is much more probable to occur in mobile environments than fixed ones. This needs to be explicitly taken into consideration in the design of mobile systems.

Disconnections and Network Partitioning: While moving, the MH can enter a geographical area out of coverage of any MSS or any other MH (to communicate in ad-hoc manner) so that it loses its connection to the network. Especially in ad-hoc scenarios network partitioning is frequent and unpredictable [17]. Causes for network partitioning are also the frequent and various perturbations characterizing mobile environments. So a link disruption e.g. can lead to network partitioning as well as the absence of neighbors in communication range of a MH participating in the execution of a MT. While partitioned from the network, the MH is not able to send or receive messages. As network partitioning is not exceptional but rather part of the normal mode of operation, it needs to be explicitly considered in the system design.

b) Node Failures: We distinguish between MH, FH and CO failures. For MHs, we identify two main failures classes, as *transient* and *permanent* failures. The CO can theoretically be implemented either on a MH or FH, and correspondingly exhibits either MH or FH failure modes. However, we separate CO failures from MH and FH failures given the central role the CO plays in commit protocols. In Section IV, we will fix the entity implementing the CO role and subsequently discuss the CO failures in detail. Currently, we do not consider deliberate failures such as Byzantine attacks or intrusions.

Transient MH failures: These occur from either software or hardware faults and usually disappear if the MH reboots. A further common cause of transient failures is the lack of battery power to sustain operation of the mobile device. Transient failures are the most probable failures of MHs in the mobile environment. In contrast to network partitioning, in the case of a transient MH failure the content of the volatile storage of the MH and consequently the state of its recent computations is lost. In this work we concentrate only on network partitioning.

Permanent MH failures: These are irreparable failures such as loss, theft or physical damage of the MH itself or its non-volatile storage, where the data and logs are stored (media failure). Consequently, all the data stored in the MH is lost.

FH failures: We assume a crash-recovery model, i.e., if the FH crashes it stops receiving, sending and processing messages until it recovers after a finite amount of time. Volatile storage of the FH is checkpointed periodically to stable storage and the FH logs its computations and received/sent messages between two checkpoints. Once a backup is done the log is deleted and a fresh logging process is initiated. The FH corresponding DBMS takes care about the recovery from transaction and media failures. The recovery includes also all logging operations which need to be done when the FH is executing a transaction.

III. Design Requirements for Mobile Transactions Protocols

We now present the design requirements of transactions in the considered generalized mobile environment. A basic issue is the need for new design requirements for mobile transactions in mobile environments. The issue being if it suffices to abort a MT when a perturbation or anomaly appears and then to restart it later? The problem with this methodology is that perturbations in mobile environments are increasingly the normal case than an exceptional situation. Another important argument is the fact that restarting the transaction involves other costs in term of energy consumption and charges for using the wireless links, which are not always tolerable in mobile environments. For this reasons we need to clearly define the boundaries in terms of design requirements. We identify the following main requirements and design issues:

Resilience to Perturbations: (Fault-tolerance and recovery) To build resilient MT 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 categorization of perturbations assists the protocol designer in identifying the main concerns and developing appropriate solutions. The overall objective for fault-tolerance is to maximize the number of committed MTs. As mentioned above, a naive approach to provide for fault-tolerance is to abort the MT each time a failure occurs and to restart it later (e.g., after a back-off time or after the failure disappears). However, this simplistic approach introduces a large overhead for the successful participants (due to frequent re-execution of the fragments) and requires some external intelligence (either from the user or from the ability of the system to detect failures). Therefore, we introduce the delay-tolerance design requirement for 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. In this paper, we expect that the application/user is able to specify an appropriate (tolerable) *lifetime* for each initiated MT. 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 messages and 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 blocking time. The reason behind minimizing blocking time is that transactions, especially executing on FHs, often lock expensive resources. These resources can not be accessed by other transactions as long as they are locked by an uncommitted one. This transaction is isolated from the rest of the transactions by locking all relevant data needed by it. As long as the locks are held, no other transaction can access the same data. This data or resources are *blocked*. The more transactions per second an application can process, the better its scalability and throughput are. 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, blocking time, especially of FH resources (because they are frequently much more loaded than MHs), should be minimized.

Scalability: Transaction protocols are said to be scalable if they support growing number of participants without sacrificing efficiency. The resource blocking time as well as the capabilities and choice of the CO are the main factors that determine the scalability of commit protocols.

IV. Delay-Aware Mobile Transactions: Overview of the Basic Approach

In the considered generalized mobile environment, network partitioning (due to either node or link failures or due to mobility) is the dominant case to consider. We investigate the impact of network partitioning on mobile transactions especially with respect to their delay-awareness and design challenges for commit protocols resilience to such failures. We also discuss the choice of the CO which is a major design decision for commit protocols in general and especially for MTs.

We proceed progressively in this section. First, we consider the existence of powerful fixed participants besides mobile participants (Infrastructure-based scenario). In this first scenario we review existing solutions which consider delay-tolerance for mobile transactions. Next, we consider scenarios involving only mobile participants that use ad-hoc wireless communication to communicate multi-hop with each other (Ad-hoc scenario). We survey existing solutions and propose a new approach that minimizes and controls the blocking behavior of mobile participants while providing for efficiency and strict atomicity. Finally, we consider a generalized MT, where both mobile and fixed participants are involved in its execution and mobile participants may communicate with each other in ad-hoc manner (Hybrid scenario).

A. MT in Infrastructure-based Scenarios

For infrastructure-based scenarios, we investigated the problem of network partitioning and heterogeneity in nodes and links in [1]. In [1], we presented FT-PPTC as a new commit solution for infrastructure-based scenarios and developed a set of efficient and generic techniques to provide MT's resilience to these fundamental perturbations. In the following we briefly summarize these techniques. First we start with *decoupling* the commit of MHs from that of FHs. The execution of the transaction is then split into two phases: *(1)* a mobile data gathering phase called pre-commit phase where the votes (either to commit or abort the MT) and the logs of the MHs (containing all operations done by the MH during the execution of its part of the MT) are collected to provide progress, and *(2)* a core Two-Phase-Commit [18] (2PC) phase, which involves only FHs for the commit action as we represent MHs by agents (which are proxy entities) in the fixed part of the network. As shown in Fig. 4, these agents (Agent_1 and Agent_2) representing MT participant MHs (respectively P-MH_1 and P-MH_2) in the fixed network store messages sent to the MHs participating in the MT and forward them to their corresponding MHs when they reconnect to the network. These agents can also act on behalf of their corresponding MHs trying to mask their disconnections. Decoupling prohibits network partitioning of MHs to affect FHs especially their resource blocking times. We concentrated our efforts on reducing the resource blocking times especially of FHs because they are frequently much more loaded than MHs in infrastructure-based scenarios. FHs are generally involved in much more transactions at a time than MHs which are not expected to participate in more than one or few transactions in parallel. We have shown in [1] how delay-awareness can help in reducing the costs of mobile transactions and in decreasing the number of aborted transactions in infrastructure-based mobile environments.

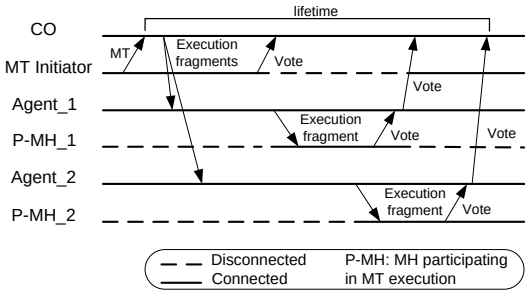


Figure 4. Lifetime in infrastructure-based scenarios

As network partitioning in this class of scenarios usually leads to the isolation of some MHs from the rest of the participants, the CO is chosen to run on one FH in the fixed part of the network. This is also beneficial for storage and overhead savings. The CO is always able to take a decision about the outcome of the MT and inform all participants which are connected to the network. The CO usually waits for a specified time (TO_{CO}) to receive the vote (either to commit or abort the MT) from each MH participating in the MT. Obviously this time depends on the slowest mobile participant since the CO needs to receive votes from every participant in order to be able to decide about the outcome of the MT. In order to have a good estimation of TO_{CO}, each participant is requested to send an estimation of the time it needs to execute its part

of the MT and send its vote and its logs to the CO. This estimation can also be updated when needed. This strategy allows the CO to easily cope with both heterogeneity of participants and their network partitioning by waiting for the maximum of received timeouts.

This timeout concept introduces delay-awareness to mobile transactions. This awareness is driven by the heterogeneity of MT participants and their connectivity. Some applications may impose a certain maximum execution time of the initiated MT. This maximum execution time is referenced in this work as *lifetime*. The lifetime of a MT models the time the user can sacrifice to receive the MT result. The lifetime can either be set by the application or estimated by the initiator or CO based on previous experiences, observations, and data collected from MTs executed previously. The initiator of the MT hands the estimated lifetime of the initiated MT to the CO or the CO estimates the lifetime by itself. The CO aborts the MT when the lifetime expires. The optimal lifetime value should account for how long disconnections of the participants can last (see Fig. 4). This value is not trivial for a generalized system model, however easier for certain systems such as closed systems. The optimal lifetime value depends also on the heterogeneity of participants. As the user can only decide about its desired waiting time, recommendations may support the user deciding for an appropriate lifetime value. In order to allow for recommendations, the system should keep a history of system properties such as the average disconnection time of mobile participants. The application may also extend this lifetime if needed.

For infrastructure-based scenarios the choice of the CO is easy due to the presence of FHs in the system. FHs can be used to implement the CO role because of their communication, computation and storage capabilities required for the implementation of the CO role on a single node. This design choice becomes much more difficult in ad-hoc scenarios due to absence of such FHs. In the next section this issue is discussed in more detail.

B. MT in Ad-hoc Scenarios

Mobile ad-hoc environments are characterized by the lack of infrastructure and also by the self-organization of the network. Ad-hoc scenarios show frequent and unpredictable network partitioning as mentioned in Section II-C. MHs in such scenarios are the only participants in the execution of MTs. We review the design challenges for commit protocols in the ad-hoc environment. Subsequently we outline existing solutions with their limitation and then present our solution approach.

1) Commit Design Challenges: As MHs do not connect to any infrastructure, the CO of the MT is required to be a MH. A MH is not assumed to have stable storage and is therefore not able to play the CO role alone unless specific assumptions on the capabilities of such a MH can be made (however same capabilities as a FH are in general not realistic). Failures of the single mobile CO usually lead to the blocking of all participants. Two extreme cases are possible: Only one CO is defined by introducing a more powerful MH (with additional assumptions on it like stable storage) or every single participant in the MT is CO. We believe that only a subset of the participants should play the CO role as justified later in this Section.

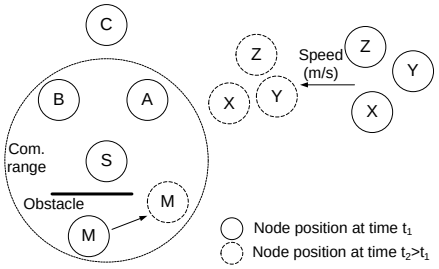


Figure 5. Factors for estimating MT lifetime in ad-hoc scenarios

The lifetime concept for infrastructure-based scenarios introduced in Section IV-A can also be used in ad-hoc scenarios. Estimating the appropriate lifetime value depends on multiple factors. A key issue is the network connectivity, which primarily depends on mobility parameters such as speed of MHs, and their communication parameters as depicted in Fig. 5. These variables make estimating lifetime in ad-hoc scenarios a challenge. Applications initiating delay-aware mobile transactions should be at least able to compute how long they will be able to wait before receiving the results of the initiated MT. This time can be used as the lifetime of the initiated mobile transaction or can be adapted to the current state of the underlined mobile ad-hoc network. In this work we do not assume synchronized clocks across the mobile entities. Thus the lifetime can elapse on different times for different participant MHs. This issue should be considered while designing an appropriate MT solution.

Given frequent network partitioning, it is challenging for ad-hoc scenarios to disseminate the fragments of the MT to their corresponding MHs. For this, partition-aware dissemination protocols can be used such as Hypergossiping [19]. Next, we review the existing commit solutions and sketch our new approach [*3) & 4)*] to deal with these challenges.

2) Existing Cluster-based MT Commit: [20], [21] propose the use of a *cluster of coordinators* preferably in single-hop distance from each other to avoid blocking of mobile participants in case one CO fails. The cluster of COs elects a single *main coordinator* and uses the 3PC protocol [22] 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 [23] to elect a new main coordinator. The termination protocol succeeds only if a majority of the COs in the cluster of coordinators does not fail and also belongs to the same partition. The assumption on the mobility of the cluster of COs made here is not valid in most of ad-hoc scenarios. Targeting a more general solution, we relax this assumption and consider a generalized arbitrary mobility model.

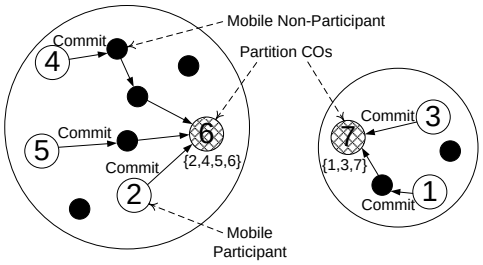
3) Proposed MT Commit based on Partition Membership: We now present a commit solution assuming that every MH in a partition knows all the members of the partition it belongs to. Later we will relax this assumption and present a corresponding solution. This solution is based partly on the work presented in [24]. Given the partition membership information, the participants in every partition elect a CO and send their votes to the elected CO which takes a pre-decision on the outcome of the MT. The pre-decision can be different from the final decision. This temporary decision is communicated to all participants within the partition. If the pre-decision is Abort, then every participant MH that receives this pre-decision can safely abort the MT. If the pre-decision is Commit, every participant in the partition should wait until all pre-decisions are collected. Alternatively, when two partitions merge, then the pre-decisions are exchanged and if all pre-decisions are collected the outcome of the MT can be safely decided since these include the votes of all participants in MT. Now all the MH participants must be informed about this outcome which can be achieved through partition-aware communication protocols similar to fragment dissemination. The correctness of the basic solution described above is assured by the partition membership assumption [24]. If this assumption is not valid a participant can be a member of a partition but the CO of that partition is not aware about the membership of this participant and subsequently the vote of this participant can be lost i.e., not included in any pre-decision and consequently not in the final decision.

The assumption that every MH in a partition knows all the members of its partition is crucial for the proposed solution. Some works [25], [26] addressed the problem of group membership in mobile ad-hoc environments, however a general solution for mobile environments remains a challenge. Furthermore, the blocking time of participants is often not considered and in worst case all participants may be blocked forever if one of the participants disappears forever. As shown in [27], there exists no non-blocking atomic commit protocol if network partitioning may occur for an unpredictable duration. Fortunately, the number of blocked participants can be minimized as we will discuss later. This approach based on partition membership information is independent from the mobility of nodes in contrast to [20], [21]. However it is based on the assumption that partition membership is available to all its members. Partitions in mobile ad-hoc scenarios are usually very dynamic as nodes may leave and join partitions arbitrarily. Therefore acquiring the global partition membership information becomes very inefficient.

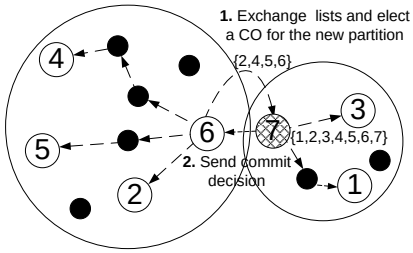
4) Proposed MT Commit without Partition Membership Information: In this paper, we present a new approach which *(a)* does not require partition membership information, and *(b)* limits the resource blocking time on participants by defining a lifetime for each initiated MT and electing a CO in each partition. The lifetime information is communicated to every participant upon initialization of the MT but can only be used by partition COs. Thus each partition CO can abort a MT if its lifetime expires. Transaction progress of the proposed approach is guaranteed because in each partition a pre-decision is agreed upon as soon as all participants in the partition communicate their votes to a priori elected partition CO. Any election algorithm can be used for the selection of the partition CO such as election of a random participant or an election based on node IDs, e.g., selecting the participant with the highest ID as a partition CO (nodes 6 and 7 in Fig. 6 (a) are elected as partition COs because they have the highest ID in their corresponding partitions). Existing election algorithms for mobile ad-hoc networks like [28], [29] can be also used for the election of the CO. The election can also be optimized according to some factors like the connectivity to other MH participants in the same partition, communication, storage and computation capabilities. The COs can also be selected upon initialization of the MT. In this case the COs are not tied to partitions that dynamically change over time, hence the notion of partition CO is not appropriate. For this work we assume that sufficient time exists to elect a partition CO before the partition composition changes.

As we do not assume the existence of partition membership information, we require that every participant sends its vote to each partition CO it encounters (i.e. it is able to communicate with either directly or multi-hop) as long as there is no final decision. In this way even if the original partition CO was not aware about this participant's vote, e.g., due to message loss, then the vote information is not lost. The vote information is communicated to other partition COs when these are encountered. The pre-decision taken by the partition CO can propagate from one partition to another as the partition CO moves to another partition. If at least one participant votes to abort the MT, the partition CO decides to abort the MT and the final decision is propagated to the rest of the participants. If the pre-decision is Commit, the partition CO can not take a decision about the outcome of the MT locally and proceeds as follows.

The CO of each partition maintains a list of all participants from which the partition CO received a vote (e.g., in Fig. 6 (a), node 6 maintains the list {2,4,5,6}). As long as the received votes are Commit votes the pre-decision is Commit. As soon as a participant votes to abort the MT, the CO decides to abort the MT as discussed above. The partition CO propagates its pre-decision and the list of participants which voted to commit the MT (if the pre-decision is Commit) to other partitions on partition joins. As shown in Fig. 6 (b), if two partitions join, the corresponding partition COs (nodes 6 and 7 in in Fig. 6 (b)) exchange their lists and elect one CO among themselves (e.g. using the same strategy as for election of the partition COs). In the example of Fig. 6 (b), node 7 is elected as partition CO because it has a higher ID than node 6. Using this schema for the election of a new partition CO if two partitions join, we guarantee that no two or more partition COs have the same knowledge about



(a) Intra-partition decision (Pre-decisions)



(b) Inter-partition decision (Final decision)

Figure 6. Partition-aware ad-hoc commit

which participants voted to commit the MT. In the latter case these partition COs can take different decisions about the outcome of the MT which violates the correctness of the proposed solution. Every CO can abort the MT if the lifetime expires before reaching a decision. Once the list of participants that voted for Commit contains all the participants of the MT, the CO (which is the single CO in the system by this time) takes the decision to commit the MT and sends the decision to the participants in its current partition. Only one CO remains in the system because all the partition COs should meet together in order to exchange their lists and by this time they select one CO among them. The lists are merged only if the election succeeds. This guarantees the uniqueness of the taken decision. The final decision is communicated to other participants when they encounter participants which already know this decision. For the dissemination of the decision and for the communication between the participants inside a single partition, either flooding or a routing protocol like AODV [30] are used depending on the ratio of participants to non-participants which exist in that partition.

Our proposed approach reduces the resource blocking time of mobile participants because the partition COs do not wait arbitrarily long to connect in order to decide about the outcome of the MT. If the lifetime expires at at least one partition CO before reaching a final decision, the MT is aborted. This is not viable in any existing solution as mobile participants have to meet asynchronously to be able to reach a final decision.

a) Evaluation of the Proposed Approach: We have conducted simulations for the case of MT Commit without Partition Membership Information to validate our approach. We briefly summarize our simulation settings and provide an overview on the simulation results.

Simulation Settings: For our conducted simulations, we have used J-Sim [31], [32], a component-based, compositional simulation environment that is entirely developed in Java and increasingly used in the mobile ad-hoc networks community [33]. We consider a representative range of parameter values to assess our approach. We selected the commonly used random waypoint mobility model [34] (node speed is uniform in [0.5, 1.5] m/s). We fix the geographical mobility area (2km x 2km) and the communication range (250m), and vary the total number of nodes to consider scenarios where the network is heavily partitioned (10 MHs) and others where it is most of the time (simulation time) composed of a single partition (100 MHs). We generate the mobility model of the nodes using the BonnMotion mobility simulator [35].

We generate transactions of similar length (transaction parts or fragments of MH participants are of similar length). We initiate one transaction at the beginning of each simulation. We fix the number of participants to 10 MHs. We assume that the participant which initiates the MT fixes the lifetime to 120s for all scenarios and selects 2 COs to play the role of partition COs as described above. Each simulation is repeated 10 times for statistical significance.

Simulation Results: We investigated in our simulations the efficiency and availability of the approach. *Efficiency* is measured in terms of resource blocking time. The resource blocking time is the time interval, where the resources at the MHs remain locked waiting for the final decision of the initiated MT. We can observe that resources at MHs are blocked as long as there is no final decision for the initiated transaction. As shown in Fig. 7, the resource blocking time is majorally determined by the lifetime. The lifetime plays a major role in determining the maximum blocking time in case of frequent and volatile network partitioning. Mobility is another important parameter in determining the resource blocking time. As the lifetime is defined by the user/application, other transactions are not blocked only for tolerable time bounds in case of highly partitioned mobile ad-hoc networks (N between 10 and 30 in Fig. 7). If the network conditions improve (N between 20 and 30), the resource blocking time decreases as more and more transactions are committed. For N between 30 and 100, the resource blocking time is very low given the fact that in some scenarios the majority and sometimes all participants are located in a single partition and in other scenarios frequent partition joins are occurring. A bounded resource blocking time allows for high scalability in terms of number of transactions and number of participants.

Availability is assessed using the rate of committed MTs to the number of initiated ones. Fig. 8 shows how the commit rate behaves when the total number of nodes N varies. The increase of N results in less number of partitions. We observe in Fig. 8 that we have 3 classes of partitioning levels. In the first level (N between 10 and 20), network partitioning is frequent and some partitions remain isolated for the whole lifetime considered (120s).

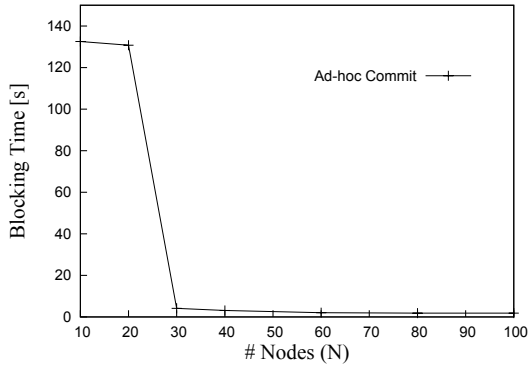


Figure 7. Resource blocking time of MHs

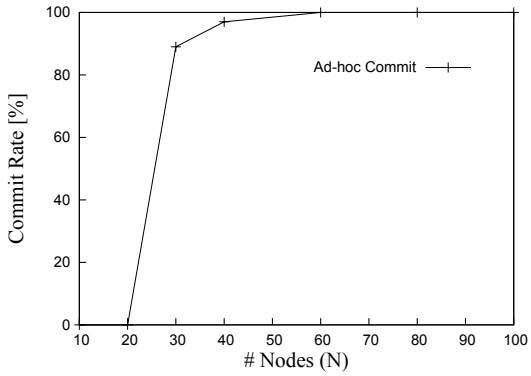


Figure 8. Commit rate as a function of number of nodes

In the second level (N between 20 and 40), network partitioning remains dominant but some transactions are committed due to network joins within the lifetime of the initiated MTs. In the third level (N between 40 and 100), a very high number of the initiated MTs are committed. This is due to the fact that the fewer existing partitions join frequently in comparison to the other two levels.

C. MT in Hybrid Scenarios

As a combination of both the infrastructure-based and the ad-hoc scenarios, hybrid scenarios can use the advantage of infrastructure-based scenarios when possible for example choosing the CO to run on a FH or defining agents as representatives for some of MH participants which can connect to fixed networks using the services of MSSs. When it is impossible to take some of these advantages the system will behave like in ad-hoc scenarios. Mobile initiators that are partitioned can also exploit multi-hop communication in order to reach a pre-decision. This is particulary helpful if one (or more) participant in the same partition as the initiator aborts the MT.

It is also important in hybrid scenarios to investigate the suitability of ad-hoc solutions involving *all* MH participants before involving new entities from the fixed network like the CO and agents of MH participants. For example a MH which initiates a MT should be given the possibility to check whether all other MH participants are in the same partition or not. If it is the case the initiator can accomplish the pre-phase of [1] in ad-hoc mode before involving other FH entities in the MT. For hybrid scenarios adaptation strategies to the current connection state of mobile participants should be developed.

V. Conclusion and Future Work

In this work we have investigated the notion of delay-aware transactions for heterogeneous and generalized mobile environments. Delay-awareness can help in providing perturbation resilience in generalized mobile embedded systems. Furthermore, we explored the problem of atomic transaction commit in different mobile environments: Infrastructure-based, ad-hoc, and hybrid. We have presented the main challenges atomic transaction protocols face in such mobile systems and also divided the generalized mobile embedded system into sub-classes. We presented a framework to provide strict atomicity for the identified sub-classes especially the mobile ad-hoc sub-class which represents the novel contribution of this work. For the ad-hoc sub-class, we examined in detail the existing work in the area and presented the description of a new atomic transaction commit solution valid for a wider spectrum of applications because it is based on fewer assumptions compared to existing solutions. We have evaluated the new approach and shown how delay-awareness can help in reducing the costs of mobile transactions and in decreasing the number of aborted transactions in ad-hoc mobile environments.

In our future work, we plan to formalize and implement the framework presented for mobile ad-hoc scenarios and assess its suitability to provide the desired requirements in ad-hoc networks. Further the aggregation of the proposed solutions to present a generalized solution for the generalized and hybrid mobile embedded environment introduced in this work will be investigated.

References

[1] B. Ayari, A. Khelil, and N. Suri, "FT-PPTC: An efficient and fault-tolerant commit protocol for mobile environments," in *Proc. of SRDS*, October 2006, pp. 96–105.

[2] M. H. Dunham, A. Helal, and S. Balakrishnan, "A mobile transaction model that captures both the data and movement behavior," *Mobile Networks and Applications*, vol. 2, no. 2, pp. 149–162, 1997.

[3] S. K. Madria and B. Bhargava, "A transaction model to improve data availability in mobile computing," *Distributed and Parallel Databases*, vol. 10, no. 2, pp. 127–160, 2001.

[4] R. Alonso and H. F. Korth, "Database system issues in nomadic computing," in *Proc. of COMAD*, 1993, pp. 388–392.

[5] E. Pitoura and B. Bhargava, "Data consistency in intermittently connected distributed systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 6, pp. 896–915, 1999.

[6] G. D. Walborn and P. K. Chrysanthis, "Transaction processing in pro-motion," in *Proc. of SAC*, 1999.

[7] G. Walborn and P. Chrysanthis, "Supporting semantics-based transaction processing in mobile database applications," in *Proc. of SRDS*, 1995, pp. 31–40.

[8] L. H. Yeo and A. Zaslavsky, "Submission of transactions from mobile workstations in a cooperative multidatabase processing environment," in *Proc. of ICDCS*, 1994, pp. 372 – 379.

[9] R. Karlsen, "An adaptive transactional system - framework and service synchronization," in *Proc. of DOA*, 2003, pp. 1208–1225.
[10] N. Nouali-Taboudjemat and H. Drias, "A policy-based context-aware approach for the commitment of mobile transactions," in *Proc. of NOTERE*, 2008, pp. 1–11.
[11] P. Serrano-Alvarado, C. Roncancio, and M. Adiba, "A survey of mobile transactions," *Distributed and Parallel Databases*, vol. 16, no. 2, pp. 193–230, 2004.
[12] Y. S. Liu, G. Liao, G. Li, and J. Xia, "Relaxed atomic commit for real-time transactions in mobile computing environment," in *Proc. of WAIM*, 2002, pp. 397–408.
[13] J. R. Haritsa, K. Ramamritham, and R. Gupta, "The prompt real-time commit protocol," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 2, pp. 160–181, 2000.
[14] E. Huang, W. Hu, J. Crowcroft, and I. Wassell, "Towards commercial mobile ad hoc network applications: A radio dispatch system," in *Proc. of MobiHoc*, 2005, pp. 355–365.
[15] G. H. Forman and J. Zahorjan, "The challenges of mobile computing," *IEEE Computer*, vol. 27, no. 4, pp. 38–47, 1994.
[16] D. Pradhan, P. Krishna, and N. Vaidya, "Recovery in mobile wireless environment: Design and trade-off analysis," in *Proc. of FTCS*, 1996, pp. 16–25.
[17] J. Hähner, D. Dudkowski, P. J. Marrón, and K. Rothermel, "A quantitative analysis of partitioning in mobile ad hoc networks," in *Proc. of SIGMETRICS/Performance*, 2004, pp. 400–401.
[18] J. Gray, "Notes on data base operating systems," in *Operating Systems, An Advanced Course*, 1978, pp. 393–481.
[19] A. Khelil, P. J. Marrón, C. Becker, and K. Rothermel, "Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks," *Ad Hoc Networks*, vol. 5, no. 5, pp. 531–546, 2007.
[20] J.-H. Bose, S. Bottcher, L. Gruenwald, S. Obermeier, H. Schweppe, and T. Steenweg, "An integrated commit protocol for mobile network databases," in *Proc. of IDEAS*, 2005, pp. 244–250.
[21] S. Böttcher, L. Gruenwald, and S. Obermeier, "A failure tolerating atomic commit protocol for mobile environments," in *Proc. of MDM*, 2007, pp. 158–165.
[22] D. Skeen and M. Stonebraker, "A formal model of crash recovery in a distributed system," *IEEE Transactions on Software Engineering*, vol. 9, no. 3, pp. 219–228, May 1983.
[23] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 133–169, 1998.
[24] W. Xie, *Supporting Distributed Transaction Processing Over Mobile and Heterogeneous Platforms. Dissertation*. Georgia Institute of Technology, 2005.
[25] G.-C. Roman, Q. Huang, and A. Hazemi, "Consistent group membership in ad hoc networks," in *Proc. of ICSE*, 2001, pp. 381–388.
[26] L. Briesemeister and G. Hommel, "Localized group membership service for ad hoc networks," in *Proc. of IWAHN*, 2002, pp. 94–100.
[27] D. Skeen, "Nonblocking commit protocols," in *Proc. of COMAD*, 1981, pp. 133–142.
[28] N. Malpani, J. L. Welch, and N. Vaidya, "Leader election algorithms for mobile ad hoc networks," in *Proc. of DIALM*, 2000, pp. 96–103.
[29] S. M. Masum, A. A. Ali, and M. T. youl Islam Bhuiyan, "Asynchronous leader election in mobile ad hoc networks," in *Proc. of AINA*, 2006, pp. 827–831.
[30] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. of WMCSA*, 1999, pp. 90–100.
[31] "The J-Sim website," http://www.j-sim.org/.
[32] "The J-Sim wireless extension tutorial," http://www.j-sim.org/v1.3/wireless/wireless_tutorial.htm.
[33] H.-Y. Tyan and J. Hou, "Design, realization, and evaluation of a component-based, compositional network simulation environment," in *Proc. of CNDS*, 2002.
[34] J. Broch, D. A. Maltz, D. B. Johnson, Y. chun Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. of MobiCom*, 1998, pp. 85–97.
[35] "The BonnMotion Mobilty Simulator," www.cs.uni-bonn.de/IV/BonnMotion/.

Brahim Ayari received his M.Sc. degree in computer science in 2004 from the University of Kaiserslautern, Germany. Currently he is Ph.D. in computer Science at the Technical University of Darmstadt, Germany, in the Dependable Embedded Systems and Software Group. His main research interests include the area of Dependable Distributed Systems & Protocols and Mobile Wireless Networks, in particular fault-tolerant transaction processing in mobile environments. He is a student member of the IEEE and the IEEE Computer Society.

Dr. Abdelmajid Khelil received his M.Sc. degree in Electrical Engineering in 2000 and his Ph.D. in Computer Science in 2007 from the University of Stuttgart, Germany. Currently he is a post doctoral fellow at the Technical University of Darmstadt, Germany, in the Dependable Embedded Systems and Software Group. His main research interests include the area of Dependable Distributed Systems & Protocols and Mobile Wireless Sensor Networks. He is a member of the IEEE and the IEEE Computer Society.

Prof. Dr. Neeraj Suri received his Ph.D. from the University of Massachusetts at Amherst. He currently holds the TU Darmstadt Chair Professorship in "Dependable Embedded Systems and Software" at TU Darmstadt, Germany. His earlier appointments include the Saab Endowed Professorship, faculty at Boston University and sabbatical at Microsoft Research. His research interests focus on design, analysis and assessment of distributed-dependable systems and software. His research emphasizes composite issues of dependability and security for SW/OS, verification/validation of protocols and especially "trusted/secure systems by design". His group's research activities have garnered support from the European Commission, NSF, DARPA, ONR, Microsoft, Hitachi, IBM, NASA, Boeing, Saab, Volvo, SSF, Vinnova, Daimler Chrysler among others. He is also a recipient of the NSF CAREER award and the 2008 IBM Faculty Award.

Suri serves as the associate Editor in Chief for IEEE Trans. on Dependable and Secure Computing, on the editorial boards for: IEEE Transactions for Software Engineering, ACM Computing Surveys, Journal of Security and Networks, and has been an editor for the IEEE Trans. on Parallel and Distributed Systems. He is a member of IFIP WG 10.4 on Dependability, and a member of Microsoft's Trustworthy Computing Academic Advisory Board.