Delay-Aware Mobile Transactions ⋆

Brahim Ayari, Abdelmajid Khelil and Neeraj Suri

Technische Universität Darmstadt,
Dependable, Embedded Systems and Software Group,
Hochschulstr. 10, 64289 Darmstadt, Germany.
Tel: +49-6151-16-7066, Fax: +49-6151-16-4310
{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, stock or database applications. Such systems entail a range of heterogeneous entities - both the embedded 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 faults like node and link failures occur rarely. This assumption is not supported in current and future mobile embedded systems 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 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, a comprehensive classification of perturbations and their impact on the design of mobile transactions is provided. 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.

Key words: Transactions, mobile database systems, dependability

1 Introduction

Future mobile embedded systems are increasingly characterized by frequent and varied perturbations. These are directly apparent to the delivery of services as constraints and failures. Mobile systems are also constrained by the scarcity of processing, storage and energy resources of mobile devices, and the continuously varying properties of wireless channels. Most of the failures which can occur in such systems are caused by node (given the mobility and size of these nodes) or communication failures. These failures can last from seconds, minutes or even hours e.g., network partitioning. Increasingly, the mobile environments are supporting applications that require strict atomicity like health-care home systems, coordination across autonomous networked vehicles,

⋆ Research supported in part by EU NoE ReSIST, EU COMIFIN and DFG GRK 1362 (TUD GKMM)

m-commerce etc. Atomic commit protocols ensure strict atomicity of database transactions and play therefore a major role for the design of these applications. Most existing atomic protocols show a restricted perturbation-tolerance leading to either poor transaction commit rate or to high resource blocking time which consequently deceases the efficiency of the mobile system. In our previous work [1], we showed that sacrificing latency (time needed to decide about the outcome of the transaction) is necessary to cope with frequent and enduring perturbations without sacrificing performance in terms of efficiency and commit success rate.

In the literature computer transactions are usually delay-sensitive. A limited body of research exists for real-time transactions [2, 3]. 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 [4] in networked embedded systems. Our work in [1] investigated primarily infrastructure-based system models. We extend this base model here to cope with a more generalized mobile system that also involves ad-hoc communication scenarios.

The remainder of this paper is organized as follows. In Section 2, the system model is described along with a set of application scenarios and a classification of perturbations in mobile environments. The design requirements for mobile transaction protocols and systems are presented in Section 3. In Section 4, delay-aware mobile transactions are introduced along with a discussion of the main challenges of introducing delay-awareness in mobile systems. Section 5 concludes the paper and briefly outlines the future work.

2 System Model, Perturbations and Scenarios

2.1 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 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 a sink. The sink collects data from SNs about a monitored area or goods etc. The architecture of the environment considered is illustrated in Fig. 1. The coverage of MSSs is much higher than the transmission area of ad-hoc communication technologies (e.g., if we compare GSM to bluetooth). 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 using Bluetooth or WLAN. Some MHs can also communicate with the sink(s) of involved WSNs. This generalized mobile distributed environment mainly consists of three basis system models which are usually tackled separately by commit protocol developers. In this work we will progressively tackle the complexity of the generalized system model by stepwise considering these sub-systems and finally combining them to our 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.

2. *Ad-hoc scenarios* involve only a subset of MHs of the model of Fig. 1 and WSNs. These MHs can communicate with each other or with mobile sinks of WSNs only in ad-hoc manner.
3. *Hybrid scenarios* are a combination of both the infrastructure-based and the ad-hoc scenarios representing our generalized mobile distributed model.

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 transactions and mobile system models, different nodes may play the CO role. 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.

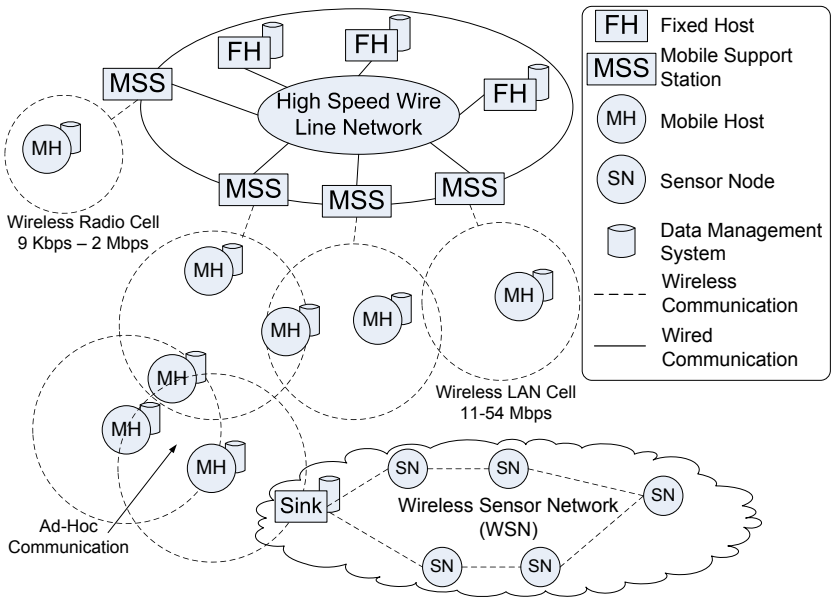

Fig. 1: Architecture of environment

2.2 Application Scenarios

In this section we classify the main applications scenarios for future embedded and ubiquitous systems, where strict atomic mobile transactions are required.

Bank/stock transactions: This type of application scenarios include mobile commerce (m-commerce) applications where users can buy or sell goods using their mobile de-

vices and involving bank servers in fixed networks to accomplish their transactions. This is an example of application for infrastructure-based scenarios.

Coordination between autonomous networked vehicles: In such an application scenario (which is a pure ad-hoc scenario) we present a potential future application where mobile transactions are needed for the purpose of coordination for safe navigation of unmanned autonomous networked vehicles. Like black boxes for airplanes, autonomous vehicles can be equipped with such black boxes which are basically mobile databases. 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 information needs to be agreed upon and recorded atomically to their corresponding black boxes. This information is needed e.g., afterwards by assurance companies in case an accident happens between these vehicles to determine which vehicle was responsible for the accident.

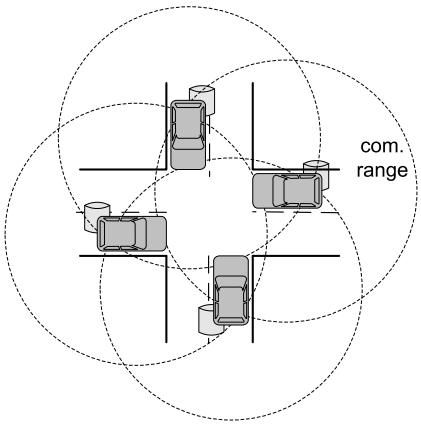

Fig. 2: Coordination between networked autonomous vehicles (livelock scenario)

Health-care ubiquitous systems: For insurance purposes, in order to monitor old people living alone in their homes a set of WSNs should be deployed in these houses and transactions are needed to react to certain situations where some actuators e.g. need to be activated together either all of them or none and this data should also be written somewhere on MHs or on FHs belonging to hospitals or police etc. This application scenario is an example of the hybrid scenarios defined in the system model.

2.3 Classification of Perturbations

Within these networked embedded systems supporting such transactional applications, we now consider two main classes of perturbations: operational constraints (power, computing, connectivity etc.) and failures. We classify the environmental constraints

relevant to mobile transactions into heterogeneity (of nodes and links), unstable storage and energy constraints. Failures of the mobile environment are classified into communication and node failures.

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 data to store on them. 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. MHs are often powered by batteries. Two of the most important sources of power consumption are transmissions and disk accesses [5]. 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 put-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, 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 [6]. Due to these issues the disk storage on a MH can not be considered as a stable storage.

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

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

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

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

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. 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.

Node Failures: We distinguish between MH, FH and CO failures. For MHs, we identify two main failures classes, i.e., *transient* and *permanent* failures. The CO can theoretically be either implemented 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 4, we will fix the entity implementing the CO role and subsequently discuss the CO failures in detail. We do not consider deliberate failures such as Byzantine faults or intrusions, but in future work we want to extend the fault model incorporating deliberate faults.

Transient MH failures: These occur mainly due to 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. Opposite 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.

3 Design Requirements for Mobile Transactions Protocols

We now present the design requirements of transactions in the considered generalized mobile environment. A basic issue is on the need for new design requirements for mobile transactions in mobile environments? Is it not sufficient to abort a mobile transaction when a perturbation or anomaly appears and then 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:

Efficiency: The efficiency of mobile transaction 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 of the CO are the main factors that determine the scalability of commit protocols.

Resilience to perturbations: (Fault-tolerance and recovery) To build resilient mobile transaction protocols, defining a comprehensive set of perturbations (constraints and failures) and a set of techniques to cope with constraints and recover from failures is mandatory. 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 mobile transactions. A naive approach to provide for fault-tolerance is to abort the MT each time a failure occurs and to restart it (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 MT.

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 our framework.

4 Delay-Aware Mobile Transactions: Overview of the Basic Approach

In the considered generalized mobile environment, network partitioning (due to either node or link failures) is the most important and frequent failure that needs to be taken particularly into consideration. We investigate the impact of this failure on mobile transactions especially with respect to their delay-awareness and the challenges of the design of commit protocols resilience to such type of failures. We proceed progressively

in this section. First we consider the existence of powerful fixed participants besides mobile participants (Infrastructure-based scenario). Then we consider only mobile participants that use ad-hoc wireless communication to communicate multi-hop with each other (Ad-hoc scenario). Finally, we consider a generalized MT, where both mobile and fixed participants are involved and some Mobile participants can communicate in ad-hoc manner with each other while being partitioned from the rest of the network (Hybrid scenario).

4.1 Infrastructure-based Scenarios

For infrastructure-based scenarios, we investigated the problem of network partitioning and heterogeneity in nodes and links in [1] 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 [7] (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. 3, these agents representing MHs in the fixed network store messages sent the MHs participating in the MT and forward them to their corresponding MHs when they reconnect to the network. Decoupling prohibits network partitioning of MHs to affect FHs especially their resource blocking times.

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 not the only reason why the CO is chosen to run on a FH, stable storage and energy overhead are also further reasons which consolidate this choice. So 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 from each MH participating in the MT. Obviously this time depends on the slowest mobile participant. In oder to have a good estimation of TO_{CO}, everyone of these participants 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.

The timeout concept described above introduces delay-awareness to mobile transactions. This awareness is driven by the heterogeneity of the MT participants and their connectivity. Some applications may impose a certain maximum execution time of the initiated MT. This models the time the user can sacrifice to receive the MT result. The initiator of the MT then estimates a lifetime for the MT and hand it to the CO. The CO aborts the MT when the lifetime expired. The optimal lifetime value should account for how long disconnections of the participants can last (see Fig. 3). This value is not trivial for a generalized system model, however easier for certain systems such as closed systems. The optimal lifetime value depends on the heterogeneity of participants and

the duration of their disconnections. Since the user can only decide about his 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 can also be given the possibility to extent this lifetime if needed.

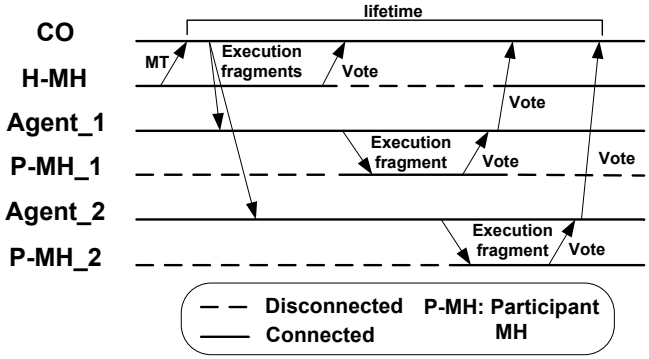

Fig. 3: Infrastructure-based scenario

4.2 Ad-hoc Scenarios

For ad-hoc scenarios only MHs are participants in the MT and can only communicate in ad-hoc manner building a mobile ad-hoc network (MANET). Since the MHs are not connected to the fixed part of the network, the CO of the MT can not be chosen to be a FH. A MH is also not assumed to have a stable storage and therefore can not play the CO role alone. Failures of the CO in this case will also lead to the blocking of all participants. As shown in [8], there exists no non-blocking atomic commit protocol if network partitioning may occur. [9] proposes to use 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 CO is represented by one member called *main coordinator*. The cluster of COs use 3PC protocol [10] to agree on a single 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 [11]. Two extreme cases that need to be considered are whether only one CO can be defined in these ad-hoc scenarios e.g. introducing a more powerful MH (with additional assumptions on it like stable storage) or the other extreme is whether it is possible to consider every single participant in the MT as a CO. In the following we illustrate the challenges network partitioning introduces in the case of ad-hoc scenarios.

Fig. 4 shows that estimating the lifetime of a MT in ad-hoc scenarios mainly depends on network connectivity, which in turn depends on different parameters like speed of the MHs, their communication range and obstacles in their vicinity. This makes estimating lifetime in ad-hoc scenarios a real challenge taking into consideration all these

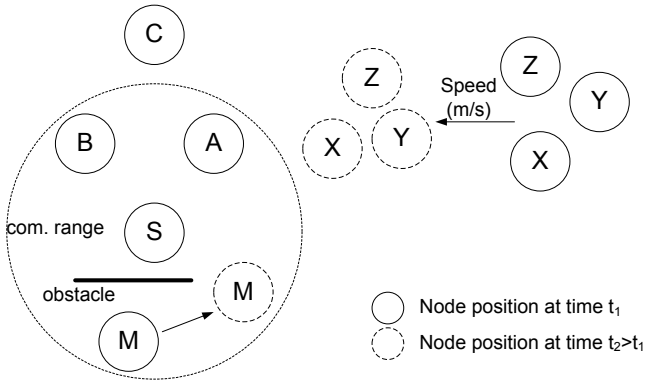

Fig. 4: Parameters for estimating lifetime in ad-hoc scenarios

parameters. Another challenge for ad-hoc scenarios is the dissemination of parts of the MT to their corresponding MHs. For this partition-aware broadcast/multicast protocols can be used such as Hypergossiping [12].

Assuming that every MH in a partition knows all the members of the partition it is belonging to like in [13], then the members of every partition can exchange their votes (either to commit or abort the MT) and take a pre-decision on the outcome of the MT (Fig. 5 (a)). The pre-decision can be different from the final decision and is only a temporary decision inside one partition which is communicated to every member of the partition. If the pre-decision is to abort the MT, then every MH participant can safely abort the MT. If the pre-decision is to commit the MT, every member should wait until all participants are in the same partition. Alternatively, when two partitions merge or join (Fig. 5 (b)) then the pre-decisions are exchanged and if no further partition exists the outcome of the MT can be safely decided and all the MH participants can be informed about this outcome which is challenging as partition-aware protocols are required. The assumption that every MH in a partition knows all the members of the partition it is belonging to is a real challenge especially in the considered ad-hoc scenarios. Some works addressed the problem of group membership in MANETs like [14, 15],but a customized solution to mobile transactions remains a challenge. The real challenge is to guarantee atomicity even if this knowledge is not available in the scenario under consideration.

4.3 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

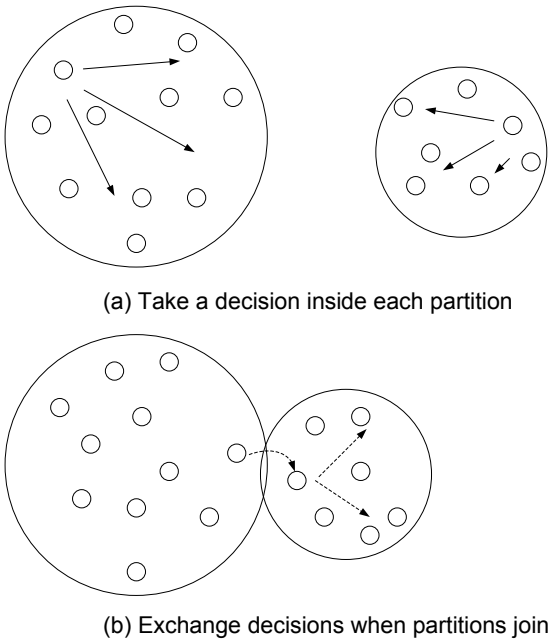

Fig. 5: Network partitioning in ad-hoc scenarios

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.

5 Conclusion and Future Work

In this work we have introduced the notion of delay-aware mobile transactions. We have shown how delay-awareness can help in reducing the costs of mobile transactions and in deceasing the number of aborted transactions in mobile environments. Delay-awareness can also help in providing perturbation resilience in generalized mobile embedded systems. 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.

In our future work, we plan to address the spectrum of mobile ad-hoc scenarios and find solutions that can be aggregated to present a generalized solution for the mobile embedded environment introduced in this work.

References

1. Ayari, B., Khelil, A., Suri, N.: FT-PPTC: An efficient and fault-tolerant commit protocol for mobile environments. In: SRDS'06. (2006) 96–105
2. Liu, Y.S., Liao, G., Li, G., Xia, J.: Relaxed atomic commit for real-time transactions in mobile computing environment. In: WAIM '02: Proceedings of the Third International Conference on Advances in Web-Age Information Management. (2002) 397–408
3. Haritsa, J.R., Ramamritham, K., Gupta, R.: The prompt real-time commit protocol. IEEE Trans. Parallel Distrib. Syst. **11**(2) (2000) 160–181
4. Serrano-Alvarado, P., Roncancio, C., Adiba, M.: A survey of mobile transactions. Distrib. Parallel Databases **16**(2) (2004) 193–230
5. Forman, G.H., Zahorjan, J.: The challenges of mobile computing. IEEE Computer **27**(4) (1994) 38–47
6. Pradhan, D., Krishna, P., Vaidya, N.: Recovery in mobile wireless environment: Design and trade-off analysis. In: Proc. of the 26th International Symposium on Fault-Tolerant Computing. (1996) 16–25
7. Gray, J.: Notes on data base operating systems. In: Operating Systems, An Advanced Course. (1978) 393–481
8. Skeen, D.: Nonblocking commit protocols. In: SIGMOD '81: Proceedings of the 1981 ACM SIGMOD international conference on Management of data. (1981) 133–142
9. Bose, J.H., Bottcher, S., Gruenwald, L., Obermeier, S., Schweppe, H., Steenweg, T.: An integrated commit protocol for mobile network databases. In: IDEAS '05: Proceedings of the 9th International Database Engineering & Application Symposium. (2005) 244–250
10. Skeen, D., Stonebraker, M.: A formal model of crash recovery in a distributed system. IEEE Transactions on Software Engineering **9**(3) (1983) 219–228
11. Lamport, L.: The part-time parliament. ACM Trans. Comput. Syst. **16**(2) (1998) 133–169
12. Khelil, A., Marrón, P.J., Becker, C., Rothermel, K.: Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. Ad Hoc Netw. **5**(5) (2007) 531–546
13. Xie, W.: Supporting Distributed Transaction Processing Over Mobile and Heterogeneous Platforms. Dissertation. Georgia Institute of Technology (2005)
14. Roman, G.C., Huang, Q., Hazemi, A.: Consistent group membership in ad hoc networks. In: ICSE '01: Proceedings of the 23rd International Conference on Software Engineering. (2001) 381–388
15. Briesemeister, L., Hommel, G.: Localized group membership service for ad hoc networks. In: IWAHN '02: Proceedings of the 1st International Workshop on Ad Hoc Networking. (2002) 94–100