Freie Universität Berlin
Institute of Mathematics and Computer Science

Dissertation

Atomic Transaction Processing in Mobile Ad-Hoc Networks

Joos-Hendrik Böse

in partial fulfillment of the requirements for the degree of
doctor rerum naturalium

Supervisors:
Prof. Dr. Heinz Schweppe
Prof. Dr. Stefan Böttcher

Submission: November 12, 2008
Disputation: January 30, 2009

Abstract

Mobile Ad-Hoc Networks (MANETs) are self-organized wireless networks where mobility and limited energy resources cause frequent communication and node failures. Guaranteeing consistency and integrity of distributed data in such a volatile environment is challenging. A key concept to assure these guarantees are distributed atomic transactions. Transferring this concept to a MANET environment raises several new research questions due to high failure probabilities. This work analyzes blocking risks of distributed transactions in MANETs and provides solutions to control these risks.

It is well known that a non-blocking atomic commit protocol cannot exist in presence of communication and node failures. This impossibility has little impact on transaction processing in fixed networks, since communication and node failures are so rare that transaction processing is not significantly affected or delayed; however, the situation in MANETs is not clear. Research has not answered yet which transactions show high blocking risks and how blocking risks are influenced by different transaction models. Therefore, a controlled risk management in MANET transaction processing is not possible yet. This thesis contributes towards a better understanding of atomic transaction processing in MANETs by presenting:

- A probabilistic model to predict the abort and blocking risks for arbitrary transaction and MANET scenarios caused by communication or node failures. The model is used to analyze strict and semantic transaction models.
- A solution to control blocking risks caused by participant failures called Shared Log Space (SLS). The SLS system allows to preserve decision logs of a transaction at a defined availability within a MANET for recovering participants. It is shown how the SLS is embedded in commit processing of strict and semantic transactions and how blocking risks can be decreased to a desired level. Two implementation approaches of the SLS are described and evaluated.
- A probabilistic model to analyze the use of a backup coordinator (BC) to reduce blocking risks caused by a node failure of the transaction coordinator. It is shown that an integration of a BC is not necessarily beneficial but may increase the blocking risk in some situations. The presented probabilistic model allows to identify such situations. Additionally, I propose a scheme to integrate the BC with the SLS to assure a required availability of decision logs if the transaction coordinator fails.

These contributions are fundamental, as they provide a comprehensive model to predict and control blocking risks in MANETs. Such a model is useful as it allows for adaptive risks management during transaction processing, i.e. it can be decided whether the use of a more reliable protocol, e.g. the SLS or BC, is indicated and to what level blocking risks can be reduced.

Acknowledgments

This thesis is the result of my research at the Database and Information Systems Group of the Freie Universität Berlin. While conducting research I had the pleasure of working with excellent people.

In the first place, I want to thank my primary supervisor Professor Dr. Heinz Schweppe. Without his support and advice as well as the freedom and confidence he has given me, this work would have never been possible. I am also very pleased to have Professor Dr. Stefan Böttcher as the second examiner of my thesis. He encouraged me to work on atomic transaction processing in MANETs and welcomed me in Paderborn several times for productive discussions.

I thank all my current and former colleagues at the Freie Universität for the cooperative and pleasant work environment. Especially, I want to thank Jürgen Bross for numerous hours of technical discussions on weekends and late evenings. I also want to thank all my graduate students, in particular Andreas Thaler, Stefan Murawski, and Petra Fiedler who contributed to this work.

I want to thank my family for always supporting and encouraging me and finally, I thank my girlfriend Alexandra for her patience during all phases of this work.

Contents

1 Introduction **3**
1.1 Motivation and Problem . . . 3
1.2 Mission and Contributions . . . 4
1.3 Thesis Structure . . . 5

2 MANETs: Background and Preliminaries **7**
2.1 Principles and Enabling Technologies . . . 8
2.2 MANET Modeling and Simulation . . . 14
2.3 System and Failure Model . . . 18
2.4 Summary and Conclusion . . . 20

3 Atomicity: Background and Preliminaries **23**
3.1 The Transaction Concept . . . 23
3.2 Transaction Models . . . 25
3.3 Atomic Commit Problems . . . 30
3.4 Atomic Commit Protocols . . . 36
3.5 Commit Protocols for Mobile Environments . . . 48
3.6 Transaction Models of this Work . . . 54
3.7 Summary and Conclusion . . . 57

4 Atomic Transactions in MANETs **59**
4.1 MANET Parameters . . . 61
4.2 Transaction Parameters . . . 67
4.3 Preliminary Considerations . . . 69
4.4 Abort Probability . . . 72
4.5 Blocking caused by Participant Failures . . . 80
4.6 Blocking Caused by Coordinator Failures . . . 95
4.7 Case Study - Mission Coordination . . . 102
4.8 Summary and Conclusion . . . 105

5 Shared Log Space (SLS) **109**
5.1 Idea and Related Work . . . 110
5.2 SLS Architecture . . . 113
5.3 Integration of SLS in Recovery Protocols . . . 118
5.4 SLS Lightweight Approach . . . 124
5.5 Log Availability Model . . . 130
5.6 Implementation of the SLS Operations . . . 144
5.7 Simulative Evaluation . . . 156

5.8 Overlay-based Implementation Approach 166
5.9 Summary and Conclusion 171

6 Backup Coordinator 173
6.1 Multiple Coordinators 174
6.2 Single BC with Veto Right Protocol 175
6.3 SLS Escalation Strategy 185
6.4 Summary and Conclusion 194

7 Summary and Conclusion 195

A Symbols and Abbreviations 199

B Simulation Tools 205
B.1 The *NS2* Simulation Process 205
B.2 MarNET Emulator 207

Bibliography 213

C Anhang gemäß Prüfungsordnung 227

Chapter 1

Introduction

Mobile Ad-Hoc Networks (MANETs) are self-organized wireless networks in which mobile devices communicate with each other in a peer-to-peer fashion without requiring fixed infrastructure. Such networks become more realistic with the increasing availability of affordable wireless devices. Communication in MANETs is inherently unreliable due to mobility and limited energy resources. Providing transactional support to guarantee consistency and integrity of distributed data and applications is therefore more demanding than in fixed networks. One challenge is to overcome high blocking risks of distributed transactions. This work analyzes and proposes solutions to compensate these risks.

1.1 Motivation and Problem

MANETs can be imagined wherever there is a certain concentration of mobile computers, such as gatherings of people in a public place or vehicular networks on highways. Possible applications are, for example, sharing of music between persons or the exchange of traffic and road information among vehicles. Additionally, MANETs are especially attractive in situations where a network infrastructure is not available or has been destroyed like in a military or disaster scenario.

Applications deployed in a MANET face the inherent challenge of having to deal with frequent communication and node failures. Preserving consistency and integrity of distributed data in presence of failures is difficult and received a lot of attention in the research field of distributed atomic transactions.

The basic rationale of atomic transactions is to model a set of operations as one larger portion of work that is treated as a single operation. This operation is executed in an atomic manner, meaning that either all embraced operations are executed successfully or none. In a distributed transaction, operations are assigned to multiple nodes of a network. To coordinate the atomic execution of these operations, a so-called Atomic Commit Protocol (ACP) is required. In a MANET where communication is unreliable and nodes can fail at any time, an ACPs cannot completely avoid so-called *blocking* situations. A node experiencing a blocking situation cannot decide whether a transaction was successful or not and has to wait for an undetermined time to reach a decision.

Although distributed atomic transactions are susceptible to blocking, numerous applications demand for atomicity guarantees in MANETs and therefore have to consider blocking risks. One example of such an application is mission coordination in a military or disaster scenario. Distributed resources, such as medical and technical units, must commit to missions in unison, because missions only succeed if all of the required resources are available. Another example of an application requiring atomic transactions are trading transactions, where an electronic good (e.g. a music file) is exchanged for some virtual currency. Here, atomicity is required in the sense that either both, goods and money are exchanged or nothing at all. Several other MANET applications demanding atomicity guarantees are described in [108].

In research, blocking is considered a major problem of distributed atomic transaction processing. The practical relevance of this problem is expected to be different for MANETs compared to fixed networks. While in fixed networks, practice showed blocking situations to be so rare that transaction processing is not significantly affected or delayed, in MANETs the situation is not clear. The general assumption is that, due to more frequent communication and node failures, blocking problems require more attention than in the fixed world. However, it has not been answered yet which kind of transactions show high blocking risks and how different transaction models influence these risks. Such information is important to decide for a given transaction whether the use of a more reliable ACP or recovery scheme to reduce blocking risks is indicated. At the time of writing this thesis, the common way to quantify these risks have been tedious simulation studies. This approach is obviously not feasible to derive blocking probabilities for an adaptive risk management during transaction processing. Hence, calculation models are necessary to predict the blocking risks of a transaction analytically. Additionally, schemes to reduce blocking risks are required when blocking risks are high. For such recovery schemes it is desirable to be able to predict their benefit.

Besides increased blocking risks, transactions in MANETs also exhibit high abort risks, since a single communication or node failure can cause the complete transaction to be aborted. To decide during transaction processing whether the abort risk of a transaction is acceptable, a calculation model is needed. In addition, being able to predict the abort risk of a transaction is required to derive its blocking probability since blocking is a subsequent problem to abort, i.e. a blocking situation can only occur if the transaction is not aborted before.

These observations motivated the main contributions of this thesis.

1.2 Mission and Contributions

The main objectives of this thesis are: (i) to examine the blocking and abort probabilities of atomic transactions in MANETs and to develop probabilistic models to derive these risks analytically; and (ii) to develop schemes to compensate for blocking risks in a predictable way. Towards these goals I present three significant contributions in the area of atomic transaction processing in MANETs. The first contribution is:

- A probabilistic model which allows to predict the abort and blocking probabilities of a transaction processed in a MANET. I consider such a model to be crucial because it allows to answer the fundamental question of whether transaction processing is possible in a MANET scenario or if a high abort risk renders transaction processing impossible. The model also reveals which blocking situation requires special attention and whether blocking is a problem at all in a transaction. The presented prediction model is developed for basic transaction models providing strict or semantic atomicity and considers recovery from failures as well as cooperative recovery in calculations of blocking risks.

The calculation models presented in this thesis allow for any given combination of a MANET and a transaction to compute the probability for blocking. Based on these probabilities, additional schemes can be integrated to reduce the blocking risk. Throughout the work, I distinguish blocking situations caused by a participant and by a coordinator. I present a compensation scheme for both blocking situations, allowing to reduce blocking risks in a controlled manner. These schemes are the second major contribution of this work:

- Blocking situations caused by a participant failure are reduced by the so-called Shared Log Space (SLS), allowing participants to leave blocking situations at a defined probability. One can think of the SLS as a distributed shared storage that preserves a transaction decision at a desired availability within a MANET. Controlling the availability of the transaction decision for blocked participants allows to adjust the probability that a blocked participant can learn the transaction decision. Blocking is therefore compensated with a defined probability. I present and evaluate two implementation approaches of the SLS. The main focus of this work is on a lightweight approach that disseminates the decision log once and does not maintain its availability any further. The second approach presented distributes the decision log within a cluster-overlay structure and actively maintains its availability.

- Blocking situations caused by a node failure of the coordinator are compensated by a backup coordinator (BC). I provide a probabilistic model to calculate the benefit of a single BC. Such a model is crucial because the integration of a BC is not necessarily beneficial but harmful in some MANET scenarios. In a second step, I propose a scheme that integrates the BC protocol with the SLS. This allows participants which are blocked due to a node failure of the coordinator to leave blocking at a defined probability.

I consider these contributions to be fundamental, as they provide a comprehensive model to predict and control blocking risks in MANETs. Providing an analytical approach to these problems is useful as it allows controlled risk management during transaction processing.

1.3 Thesis Structure

The remainder of this dissertation is structured as follows: First, relevant background information on MANETs and atomic transactions is given in Chapter 2 and Chapter 3.

In Chapter 2, I present several important routing and broadcasting schemes required for the understanding of subsequent chapters. Additionally, the system and failure model used within this work is defined.

Chapter 3 introduces the concept of atomic transaction processing and discusses different transaction models. The atomic commit problem is defined formally, and important ACPs are presented and evaluated according to their applicability in MANETs. At the end of Chapter 3, the strict and semantic transaction models are defined, which are used in subsequent chapters to reason about abort and blocking risks as well as compensation schemes.

Chapter 4 presents the first major contribution of this thesis. Here, the calculation model for abort and blocking probabilities of strict and semantic transactions is presented. The chapter also demonstrates how parameters of the system model such as the probability of communication failures can be derived for a given MANET scenario. The derived formulae for abort and blocking probabilities are applied to an example scenario and compared to experimental results obtained by simulations using the *ns2* network simulator. The presented calculations of blocking probabilities also consider simple recovery schemes. Parts of the calculation model presented in Chapter 4 have been published in [28, 26].

Chapter 5 gives a formal description of the Shared Log Space (SLS) architecture and its integration into strict and semantic transaction models. Two implementation approaches of the SLS are described, while the main focus of Chapter 5 is on a lightweight approach and its underlying availability model. An overlay-based implementation approach is only briefly described. In the end, a *ns2* simulation study shows the benefit of the SLS for blocked participants in an example MANET scenario.

In Chapter 6, compensation for blocking situations caused by a node failure of the coordinator with a backup coordinator (BC) is described. I provide a detailed calculation model to compute the decrease in blocking probability if a BC is used. Additionally, a strategy is presented that combines the SLS with the BC approach. Calculation models presented in this chapter have been published in [27].

Finally, Chapter 7 summarizes and concludes this thesis. Appendix B contains detailed information about simulation parameters and a description of the simulation environments used for evaluation of the schemes proposed in this work.

Chapter 2

MANETs: Background and Preliminaries

A Mobile Ad-hoc Network (MANET) is an infrastructure-less network formed by autonomous mobile devices that are equipped with short-range radio devices. The key characteristics of MANETs are node mobility, causing frequent topology changes of the network, and limited energy and computational resources. These characteristics raise many challenges for network protocol design in all layers of the protocol stack. E.g. the physical layer has to deal with rapid changes in link quality, while the media access layer has to minimize collisions and deal with hidden and exposed terminal problems. At the network layer, mobile nodes have to calculate paths to allow for packet exchange over multiple hops. Although research has made great advances, communication in MANETs is inherently unreliable and applications deployed in a MANET have to deal with frequent node and communication failures.

Even though in recent years a large research community has actively worked on MANETs, they are still mainly an academic topic. An exception are personal area networks (PANs) using the 802.15 (Bluetooth) standard [59]. Here, up to eight devices can be connected in an ad-hoc manner in a so-called piconet. However, such PANs are not at the center of this work, as they are mainly intended to pair mobile devices at very close distances, such as a headset with a mobile phone or a PDA with a desktop PC. The type of MANET that is considered in this work are networks of larger scale, covering complete university campuses or even complete districts of a city and consisting of 10–100 mobile nodes. Except for small lab deployments with just a few nodes, real-world MANET experiments are not practical. Simulation of MANETs has become the preeminent approach to evaluating proposed schemes, while the potential of analytical modeling is often underrated.

In this chapter, I briefly present principles and enabling technologies of MANETs to provide important background information. Besides technical issues like radio standards and routing algorithms, I introduce standard modeling concepts of MANETs. However, I only present a selection of topics important to this work; for a more comprehensive introduction to MANETs I refer the reader to [115, 73]. Especially important for the remainder of this work are the system and failure models derived from the MANET principles presented in the end of this chapter in Section 2.3.1.

2.1 Principles and Enabling Technologies

The main driving factor of MANETs is the rapid progress in wireless communication. Open standards and the availability of cheap chipsets have resulted in a wide proliferation of mobile devices that are technically capable to cooperatively form MANETs. The most widely accepted standard for wireless ad-hoc technology is the IEEE 802.11 protocol family [58] briefly described in the following, as the 802.11 physical and MAC layers are used in simulations in later chapters.

At the network layer, nodes must cooperate to calculate communication paths, and forward packets along these paths to allow for communication between nodes not in each other's direct radio range. Discovery and maintenance of such routes is the task of multi-hop routing algorithms. The main challenge of such multi-hop schemes is to deal with node mobility, which leads to sudden path breaks and network partitioning. The frequency of such events varies widely between different MANET scenarios, as they depend on the grade of node mobility as well as on the network density. It is a major problem in research on MANETs to derive general insights that are valid for a wide range of MANET scenarios. A common classification is to differentiate between *dense* and *sparse* MANETs. Although this is a raw informal classification, it is widely used and most protocols developed for MANETs focus on either dense or sparse networks. General solutions that fit all classes of MANETs are rare.

In this section I will present the important multi-hop routing algorithms as well as broadcast schemes. This is important because routing information provided by routing schemes is used later by the Shared Log Space (SLS) presented in Chapter 5, and broadcast schemes are a fundamental building block of the SLS to increase data availability in a MANET. Additionally, the current MANET research landscape is briefly outlined and the mission of this work is related to other research activities.

2.1.1 Radio Technology

The most widely accepted wireless standard is the IEEE 802.11 protocol family. Members of this family are the 802.11a/b/g/p protocols. The radio range of mobile nodes using 802.11 is approximately a few 100 m outdoors and less than 100 m indoors. IEEE 802.11 provides an ad-hoc mode in addition to the commonly used infrastructure mode. However, the ad-hoc mode defined by IEEE 802.11 does not define any multi-hop routing schemes. Other standards, e.g. the hiperLAN family (the european counterpart to IEEE 802.11) provides a centrally controlled packet-forwarding mechanism, allowing nodes to receive packages that are in deadspots of a mobile support station, while a distributed path calculation for multi-hop routes is not part of this standard. To provide such a multi-hop routing in MANETs, a special routing extension on the network layer like AODV [117] or DSDV [116] is mandatory.

	802.11a	802.11b	802.11p	hiperLAN-1	hiperLAN-2
Frequency	5 GHz	2.4 GHz	5.9 GHz	5.15+17.1 GHz	5 GHz
Radio range	100 m	some 100 m	1000 m	50 m	50–100 m
Transm. rate	54 Mbps	11 Mbps	27 Mbps	23.5 Mbps	54 Mbps

Table 2.1: Comparison of wireless standards.

Table 2.1 compares transmission rates and radio ranges of different 802.11 and hiperLAN standards. While IEEE 802.11a/b are commonly found in standard consumer products, 802.11p is an evolving standard for vehicular wireless access, supporting high node speeds up to 200 kph. Other standards assume only node speeds of about 50 kph.

2.1.2 Research Issues in MANETs

As MANETs are a young research field, manifold research activities are found in this area. This subsection gives a brief overview of the main research fields in this area and shows how the research presented in this thesis integrates into the MANET research agenda and how it relates to other activities in this area. Figure 2.1 shows the different layers of a MANET and lists important research issues at each layer.

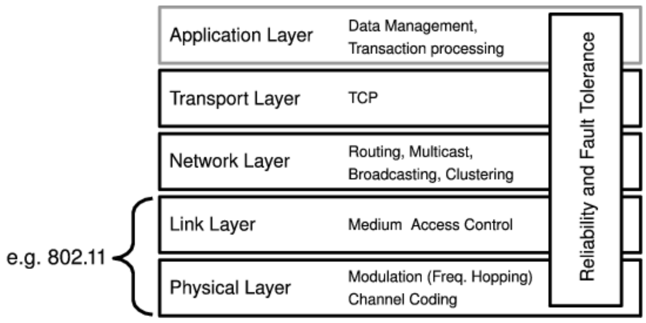

Application Layer
Data Management, Transaction processing
Transport Layer
TCP
Network Layer
Routing, Multicast, Broadcasting, Clustering
Link Layer
Medium Access Control
Physical Layer
Modulation (Freq. Hopping) Channel Coding
e.g. 802.11
Reliability and Fault Tolerance

Figure 2.1: Research issues related to MANET layers.

The link and physical layers are considered by the wireless standards described above, e.g. 802.11 or hiperLAN. Here, research focuses on medium access, e.g. solving the hidden terminal problem and frequency modulation at the physical layer. Most current research activities are found at the network layer of the MANET protocol stack. Here, research activities are concerned with routing schemes and other communication-related problems such as reliable multicast or efficient broadcast schemes. Clustering protocols are examined to allow for logical grouping of mobile nodes and to establish an overlay hierarchy that can be used e.g. for routing or service discovery. At the transport layer, the main challenge is to provide for more convenient transport protocols like TCP. The application layer of the MANET protocol was recently considered by scholars examining data management problems like data consistency and transaction processing, which is also the main concern of this thesis.

The main challenge here is to design schemes that are capable of tolerating frequent communication and node failures of a MANET. Coordination problems such as consensus and leader election attracted the interest of scholars in [21, 132, 155]. The main problem of research activities on the application layer in MANETs is that they are heavily influenced by subjacent layers. As no standardized protocol stack for MANETs exists, the performance of schemes proposed at the application layer is influenced by the configuration e.g. of routing schemes, MAC layer and transport layer. Another observation in this context is that many problems affect multiple layers of the protocol stack. While in fixed networks the strict separation of layers is beneficial as it hides complexity and allows for interoperability of different implementations of the different layers, it is often desirable in MANETs to comprise multiple layers into the solution of a problem. For example, the general problem of *reliability and fault tolerance* affects all layers of a MANET. At the application layer, certain reliability requirements may be posed that are to be considered by the transport and network layer, e.g. by prioritizing certain packets and by modifying the modulation at the physical layer to minimize the loss rate of these packets. Figure 2.1 shows how the issue of reliability and fault-tolerance cuts through all MANET layers. This general approach is known as *cross-layer* architecture and applies to many other problems, such as assuring *cooperation and fairness* in MANETs or *energy efficiency*.

Generally, the solutions in MANETs are application driven; the development of "one-size-fits-all" solutions is not feasible for MANETs. For example, a vehicular MANET must deal with fast-changing topologies, while a MANET of mobile sensors is more susceptible to failures caused by exhausted energy resources. Hence, research in MANETs often considers a certain class of MANET scenarios, making it difficult to evaluate the relevance of a proposed approach for other scenarios. Here, analytical models significantly simplify the evaluation of schemes for different classes of MANET scenarios compared to simulation based evaluation.

2.1.3 Routing Algorithms

The most commonly used routing protocols are *topology-based*. Here, nodes use knowledge about the current topology of the network to decide which neighbor a packet must be forwarded to for it to reach a remote node. In contrast, e.g. *geographic routing* uses the geographic location of the destination node to decide how to forward a message. *Hierarchical routing* protocols establish an overlay network, for example, by clustering used for routing.

The main problem to be solved by topology-based routing schemes is how to discover the constantly changing network topology. To learn about the current network topology, mobile nodes periodically announce their presence and keep track of their current neighbors. By sharing collected knowledge, the complete network topology can be learned over time. However, the learned topology must undergo constant maintenance to delete broken routes and detect new ones.

Topology-based routing schemes either constantly keep track of the current network topology or discover routes only on demand, i.e. when an application actually issues a message, the routing scheme initiates route discovery. Approaches following the first strategy are called *proactive*, while the second approach is denoted as *reactive*. The important point for this work is that proactive as well as reactive routing schemes provide a local routing table that allows the direct neighbors of a node to be discovered. In Table 2.1(a) an example routing table for the topology depicted in Figure 2.1(b) is shown.

(a) Routing table of node ***A***.

DESTINATION	NEXT HOP	HOPS	SEQ. # DSDV	SEQ. # AODV
E	A	1	E 08	10
C	B	2	C 02	24
B	A	1	B 06	32
F	B	2	F 04	14
A	A	0	A 12	68
G	E	2	E 05	12

(b) Example MANET topology.

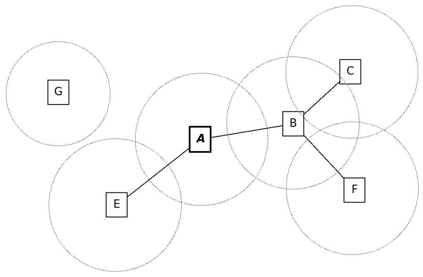


Table 2.2: Example MANET routing topology.

Route entries are annotated with sequence numbers and other protocol specific flags, to indicate the age and state of a route as well as to anticipate cycles of routes.

In the following, the most important proactive and reactive routing schemes for this work are briefly presented. For implementation and simulation of approaches developed in this work I use the DSDV (proactive) [116] and AODV (reactive) [117] routing protocols that are briefly presented in the following, while advanced variants of AODV like the Reliable AODV protocol [79] are not considered.

2.1.3.1 Proactive Routing

In a proactive routing scheme, nodes constantly keep track of topology changes. Hence, a node maintains routes to every node at any time. The most notable proactive routing algorithm is the *Destination Sequence Distance Vector Routing* (DSDV) protocol. A DSDV routing-table entry contains the destination, the next hop to use as relay, the hop count of the route and a sequence number. The sequence number of a route is issued by the destination node and indicates the age of the entry. A node periodically announces its routing table to its neighbors with a sequence number increased by two. A node detecting a broken link increases the sequence number of its routing entry by one.

Table 2.1(a) shows an excerpt of an example routing table of the node with id A for the example topology shown in Figure 2.1(b). From its routing table, A can learn that node E and B are in its direct neighborhood, while the route to G via E is currently broken. For a more detailed description of DSDV, see [116]. The important issue for this work is the fact that nodes in direct hop range can be directly extracted from the routing table.

In practice, DSDV has been found to be practical for smaller networks with few nodes. Its main drawback is that routes are maintained that are possibly never used, which causes a higher message overhead compared to reactive schemes.

2.1.3.2 Reactive Routing

In reactive routing schemes, topology information is only acquired if a route to a certain destination is requested. Hence, unnecessary route maintenance is avoided in exchange for a short delay induced by route discovery. The most prominent reactive routing schemes are *Dynamic Source Routing* (DSR) [74] and *Ad-hoc On Demand Distance Vector* routing (AODV) [117]. While DSR is a rather old protocol, AODV is currently one of the most used-schemes and is introduced here briefly.

In AODV every node keeps track of its direct neighbors by listening to periodical HELLO broadcasts issued by each node to announce its presence to its direct neighborhood. In the case where a route to a destination is unknown, the source node issues a route request (RREQ) message to all its direct neighbors. The RREQ message contains the network identifier of the source node, the identifier of the destination node, a lifespan value and an increasing sequence number, serving as the unique id of the message. A node receiving a RREQ message rebroadcasts the message to its neighbors if it does not know a route to the destination of the RREQ. If it has a route to the destination, it answers with a route reply (RREP) message to the source node. In both cases, the node saves the new route learned through the RREQ message. Hence, the RREP message can be transferred back on the path it took to the node that provided a valid route to the destination. The RREQ message is rebroadcast until its lifespan expires. If the source node does not receive an answer to its RREQ message, it reissues a new RREQ message with a higher lifespan value and a new id.

Routing entries are adjusted by distributing route error (RERR) messages that are issued when a node recognizes that one of its routes has become invalid.

Table 2.1(a) shows an sample excerpt from a routing table; the fifth column contains AODV sequence numbers. These sequence numbers are used to decide whether a piece of received routing information is newer than the current entry. A higher sequence number indicates a more recent route.

2.1.4 Broadcasting in MANETs

Delivering a message to all nodes in a MANET is the goal of a class of broadcast protocols called flooding protocols. Broadcasts are used to discover routes or resources in MANETs. For example, in this work I will use broadcasts to discover nodes that have participated in a transaction and to distribute log data within a MANET.

I borrow the classification of broadcast schemes proposed by [77], distinguishing *broadcast-in-space* and *broadcast-in-time* protocols. Broadcast-in-space protocols deliver a message within milliseconds to all nodes within a network partition. Nodes that reside in other partitions are not reached. These protocols are designed for delay-sensitive applications such as service or route discovery and have to deal with the broadcast storm problem [105], which causes a complete blocking of the wireless medium when too many nodes try to forward a package simultaneously. Proposals to avoid broadcast storms are to use a random waiting time before a node forwards a received packet (*Random Access Delay* (*RAD*)), probabilistic schemes that forward a packet-based on a predetermined probability [105], counter based approaches that rebroadcast only if a the number of redundant packages received falls below a certain threshold value, and protocols using information about the current number of neighbors [114, 33] or the neighbor change rate [146] to decide whether a package is going to be rebroadcasted. For an overview and discussion of the numerous proposed broadcasting-in-space schemes, I refer to [77, 154]. In this work, I use broadcasting protocols based on the probabilistic approach.

In contrast to broadcast-in-space schemes, broadcast-in-time protocols rely on node mobility to distribute data in a MANET. Such protocols are based on a store-and-forward strategy where packets are transported from one partition to another partition by moving nodes and are primarily designed for sparse MANETs. The main problem to be solved here is not the broadcast storm problem but to detect network partitions. To initiate a rebroadcast of a data item in a partition where the data has not been distributed yet, either negotiation-based strategies or repetitive-broadcasting strategies are used. In negotiation-based protocols such as SPIN (*Sensor Protocols for Information via Negotiation*) [70] or NADD (*Negotiation based Ad-Hoc Data Dissemination Protocol*) [66], nodes advertise data they have stored to surrounding nodes, which in turn answer with a request for data they have not received yet. Hence, data is only rebroadcast on request by nodes that are missing some data. Repetitive broadcasting of a data item omitting a negotiation phase is the central approach of the so-called hiper-flooding protocols like [109, 149]. Rebroadcast of data is initiated, if partitions previously separated are merged.

Recently, scholars have proposed integrated protocols that cope with the broadcast storm problem and with network partitioning. The most prominent approach of this class is hypergossiping [78]. Hypergossiping applies different strategies to reach all nodes in a partitioned network. Within partitions, a so-called intra-partition forwarding strategy is used, which is based on a probabilistic forward scheme. Broadcast repetitions are used when merging of partitions is detected.

Broadcasting schemes in MANETs is an active research area that constantly generates more message efficient flooding protocols. The SLS proposed later in this work requires flooding schemes but does not make any assumptions on how they are implemented.

2.1.5 Summary - Principles and Enabling Technologies

In this section, I have presented the MANET research landscape and integrated my research activities. As multi-hop routing and broadcasting are relevant to this work, some key characteristics and concepts have been presented for these areas. The presented MANET characteristics are captured by the system and failure models of this thesis presented at the end of this chapter.

The presentation of routing schemes also narrows down the class of MANETs I am interested in within this work, which are MANETs using topology-based routing schemes posing no further assumptions on node characteristics such as knowledge about the current position etc.

However, the schemes and models presented in the remainder of this work do not depend on certain multi-hop routing schemes, i.e. they are independent of the concrete multi-hop routing scheme used in a MANET scenario. More advanced routing schemes possibly available in future can be easily integrated into the probabilistic models presented in this work.

2.2 MANET Modeling and Simulation

Real-world MANET experiments are tedious and expensive in terms of required equipment and resources. Hence, examination of MANETs is commonly done analytically or by means of simulations. In both cases, abstract models are required to describe the real-world behavior of a MANET at an abstract level.

Analytical investigation of MANETs is mostly based on geometric graph models that are used to derive certain static characteristics, such as connectedness or path probability, while dynamic topology changes are hard to express analytically. To understand the influence of these network dynamics, simulation-based approaches are typically used.

In this work analytical as well as simulation-based studies are presented. Hence, the basic properties of both approaches are briefly presented in the following.

2.2.1 Geometric Graph Models

To analytically reason about a MANET, it is commonly modeled as a geometric random graph. It is assumed that nodes are randomly placed in an area $\mathcal{A}$, while links between nodes are defined by a wireless link model. The spatial distribution of nodes and links between them defines the topology of the MANET at a random time. The wireless link model describes whether a direct link between two nodes exists and models the propagation of the radio signal. The commonly used link model is the *purely geometric link* model [121], which assumes that the received signal strength diminishes in proportion to some power of the geometric distance between two nodes. Hence, if omnidirectional antennas are assumed, a node has links to all nodes that are currently located within a certain radius r. In reality, objects such as buildings or walls shield the signal in different ways, so that two receivers that are within a radius r to a sender encounter different probabilities to receive the signal. This behavior is approximated by the *shadow fading link* model [121], which assumes a log-normal distributed probability for a link between two nodes in distance r. Based on a spatial distribution of nodes and the geometric link model, a random geometric graph can be defined, consisting of a set $V = \{n_0, n_1, \ldots n_m\}$ of vertices (nodes) and a set E of edges (links), described by node pairs $E = \{n_i n_j, \ldots, n_n n_m\}$. The geometric random graph G is then defined by $G = \{V, E\}$.

Within this work the central concept of *path probability* is derived from G. The path probability P_{path} describes the probability that for a random placement and given transmission capabilities, a multi-hop path exists between two randomly chosen nodes. While analytical derivation of P_{path} is complex for n-hop paths and not solved yet, there exist calculation models to estimate P_{path} for 1–2 hop paths. The probability that a communication path between two nodes survives for a period t is defined as *path-duration*. The path-duration is mainly influenced by the changing topology and is therefore hard to express analytically. Hence, simulation is commonly used to reason about path-durations in a MANET scenario. A MANET simulation is based on various models as described in the following.

2.2.2 Simulation Models

Simulation of MANETs is by far the most often used approach to investigate the inherent properties of MANETs and to evaluate the performance of protocols and applications. At the time of writing, simulation-based approaches are the only possibility for examining the influence of dynamic topology changes in a MANET. However, the main challenge of a simulation-based analysis is to define appropriate models that approximate reality at a reasonable level of abstraction. The dynamic topology of a MANET is influenced (i) by node mobility, defined by a *mobility model*, (ii) node failure characteristics defined by an *outage model,* and (iii) by the *link model* (see above) as depicted in Figure 2.2.

The mobility model has a significant influence on topology dynamics, as it directly defines speed and direction of movement for every node. In this work, I will use different mobility models, namely the *Area Graph Based* (AGB) model and the more common *Random Way-Point* (RWP) mobility model. I will dedicate the next subsection to these models. The outage model describes node outages e.g. caused by exhausted energy resources or technical failures of nodes. It commonly considers energy consumption required for packet transfer and hence is influenced by network traffic. The outage model can also be defined by assuming arbitrary probability densities for technical and other failure events. Based on the dynamic topology, the behavior of the physical, link and network layers can be simulated. On top of this setup application models are implemented like transaction models as in this work.

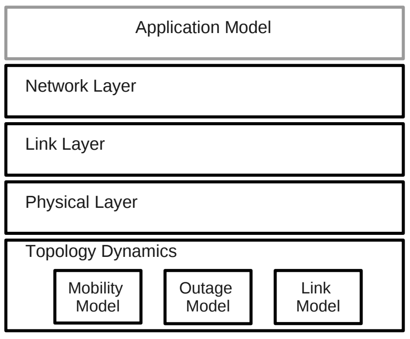


Figure 2.2: Models required for MANET simulation.

As an implementation of these sub-models and layers is tedious and error prone, correctly proved simulation frameworks like *ns2* [2, 23] or GloMoSim [11] are the first choice for simulative approaches. In this work, I use *ns2* as well as the MarNET emulation system described in Appendix B.2 for simulative evaluation. Even with a given simulation framework, the mobility, outage, and link models have to be adjusted by numerous parameters that may bias results. Deriving meaningful statistical results is therefore far from trivial, and an analytical approach should always be considered first. The credibility of MANET simulations has to be considered as critical if either one of the important models is not considered or if implementation and configuration of one model is not completely understood.

For this work, I consider analytical models, as the more valuable contribution for evaluation, as they abstract from the numerous parameters induced by simulation models. I regard simulation studies as useful tests whether the proposed approach can be efficiently implemented and to provide a "reality check", which allows to identify whether any important factors have been missed by an analytical model. The simulative evaluation is therefore kept brief compared to the model parts of this work. For a further discussion of pitfalls and problems of simulative evaluation, I refer to [8, 86].

In the following, the important mobility models used later are described.

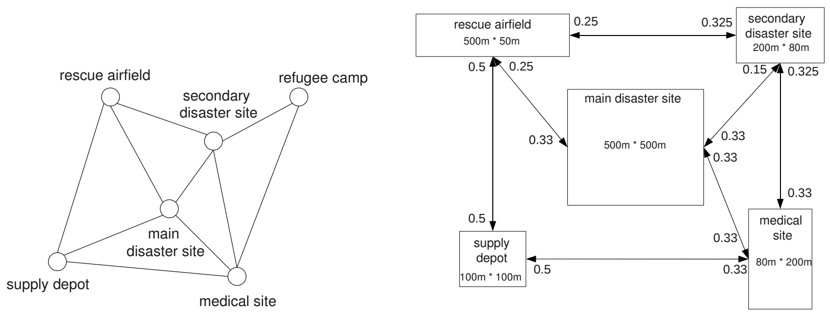
2.2.3 Mobility Models

The choice and understanding of mobility models used in simulations is crucial to evaluate results. While a large number of mobility models have been proposed, only a few are commonly used by the research community, mainly because they are implemented in the major simulation frameworks like *ns2*. For a comprehensive overview and classification of available mobility models, see [32, 15].

The main classification criterion of mobility models is their level of detail. At a microscopic level, the individual node motion is described, e.g. *"at time 100 node n_1 begins to move from point (x_1, y_1) towards point(x_2, y_2) with speed v"*. On a macroscopic level, movement is described on a larger scale, e.g. "*At time 100 node n_1 moves from building A to building B*".

The by far most commonly used mobility model is the Random Way-Point (RWP) mobility model, first described in [75]. RWP is a mobility model that describes movement on a microscopic level and does not assume any restrictions on movement paths of nodes such as roads. Variants like the Random Direction Model [127] overcome the problem of density waves in the center of the simulation area in the RWP model by allowing only direction changes if a node reaches the boundaries of the simulation plane. Other models sharing the same basic properties of the RWP model are described in [100, 62, 112]. Because the RWP model is used in this work, it is presented in more detail in Subsection 2.2.3.2. Another common model describing movement on a microscopic level that restricts movement to a defined grid of roads is the *Manhattan Mobility* (MH) model proposed in [96]. Other classes of models describing movement at a microscopic level are *group mobility models*, which take the correlation between individual nodes into account; see e.g. [72]. For a survey of such models mainly proposed in transportation theory, see [71].

Models describing node movement on a macroscopic level are commonly found in the area of cellular networks. Here, only the movement from one cell to another is of interest, in order to model the hand-over process of mobile clients between neighboring base stations. Because MANETs do not rely on a cell-based network structure, these models cannot be applied directly. A generalization is to model node movement as a random walk on an undirected graph as proposed in [144]. In a graph model, vertices of the graph represent locations or areas, while edges represent paths in the real world, e.g. streets between these areas. A mobility model based on such a graph describes the transition of nodes from one area to another. Hence, it is modeled how nodes *"jump"* from one location to another without describing the microscopic view of the transition process between areas. Additionally, the microscopic view on movement within areas



(a) Graph-based mobility model. (b) Area Graph–based mobility model.

Figure 2.3: Graph-based and Area Graph-based mobility model.

is neglected here. A model compensating these disadvantages is the so-called Area Graph–Based (AGB) [19] mobility model. The main idea of the AGB mobility model is to describe node motion on a microscopic as well as on a macroscopic level. In the following, the Area Graph-Based mobility model is described, followed by the RWP mobility model.

2.2.3.1 Area Graph Model

The Area Graph-Based model (AGB) is based on a directed graph $G = (\mathcal{A}, E)$. The vertices $\mathcal{A}$ of G represent areas with higher node density, like buildings or public places. In contrast to the graph model proposed by [144], where vertices are interpreted simply as points, the AGB model interprets vertices as areas with defined dimensions, e.g. the dimensions of a university campus or of a building. The edges E represent paths e.g. streets or gangways between these areas. Edges connecting two areas $\mathcal{A}_i$ and $\mathcal{A}_j$ are directed and weighted, while the weights influence the probability of which outgoing edge a node chooses. Figure 2.3(b) depicts the general idea of the AGB model.

Movement between areas is described on a macro level similar to the graph-based mobility model of [144] shown in Figure 2.3(a). In contrast to the graph-based model, the AGB model describes the movement of nodes also on a micro level within areas. When a node moves from one area to another, e.g. from the rescue airfield to the main disaster site, it joins the MANET formed by mobile nodes in the destination area $\mathcal{A}_i$ (e.g. the main disaster site). Nodes spend a random period in $\mathcal{A}_i$ before moving to another area $\mathcal{A}_j$. The sojourn time t a node remains in A_i is therefore modeled as a random variable described by a cumulative distribution function (cdf) $F_L(t)$. In this work, I will assume that $F_L(t)$ describes an exponential distribution.

The movement inside $\mathcal{A}_i$ is described by a microscopic mobility model. While this can be any microscopic model, I will use RWP in this work. The time a node needs to move from one area to another is defined by the length of the edge and the speed of the node, which is uniformly chosen from a defined interval $[v_{min}, v_{max}]$.

The main reason to choose the AGB model in my thesis is its ability to model a leave and join rate of nodes in a certain area. Nodes leaving an area $\mathcal{A}_i$ disconnect from the MANET in $\mathcal{A}_i$ and hence are not available anymore for transactions and recovery schemes examined in this work. Hence, the AGB model allows to simulate a *churn rate* of nodes in $\mathcal{A}_i$ that has various influences e.g. on data availability and transaction failures. The leave probability described by $F_L(t)$ is directly used in the failure model of this work described below.

2.2.3.2 Random Way-Point Mobility Model

The goal of the RWP model is to achieve a uniform random distribution of nodes within a bounded area. This is achieved by the following behavior of nodes: at the beginning of the simulation all nodes are randomly placed in the simulation area $\mathcal{A}$. Then every node randomly chooses a so-called way-point from all points in $\mathcal{A}$ and moves towards this point in a straight line with constant speed v, while v is randomly chosen from the interval $[v_{min}, v_{max}]$. When the node arrives at the way-point, it pauses for a time t_{pa} and then chooses a new way-point and speed.

Simulations have shown, that the goal of a completely uniform random distribution in $\mathcal{A}$ is not met by the RWP model. It has been shown that the node density in the center of $\mathcal{A}$ is slightly higher than in the border areas of $\mathcal{A}$ [18]. However, the RWP model approximates a uniform distribution quite well. As parameters it requires the dimensions $(x_{\mathcal{A}}, y_{\mathcal{A}})$ of $\mathcal{A}$, a speed interval $[v_{min}, v_{max}]$ from which v is uniformly chosen, and a pause time t_p.

2.2.4 Summary - MANET Modeling and Simulation

This section has presented background information on how MANETs are commonly examined and analyzed. As MANET research is based on simulations and analytical considerations rather than on real-world experiments, knowledge about the basic modeling principles is essential to the reader to interpret the results of this work.

Based on the models introduced, some important definitions like path probability P_{Path}, path duration, and expected sojourn times of nodes in $\mathcal{A}$ have been introduced, which are embraced in the system model of this work presented in the following.

2.3 System and Failure Model

Given the preliminary considerations about MANETs above, this section will define the system and failure model of my thesis. The system model is based on the standard partially synchronous system model, assuming communication and site failures of nodes and is enhanced with certain assumptions on reachability and availability of nodes in a MANET $\mathcal{A}$.

2.3.1 System Model

The system model considers a MANET $\mathcal{A}$ formed in a single area of a larger network described by the AGB mobility model. The macro view of the AGB model is used here to model the fact that a MANET is not a closed system, but new nodes can join as well as leave $\mathcal{A}$. I assume that the total number of nodes in $\mathcal{A}$, denoted by $n_{\mathcal{A}}$, shows a negligible variation and is quasi constant over time, which is feasible if nodes enter and leave $\mathcal{A}$ at equal rates.

The probability that a node disconnects from $\mathcal{A}$ because it moves into another area at time t is described by the probability density function (pdf) $f_L(t)$. Analogously, the probability that a node joins $\mathcal{A}$ after being disconnected at time t is described by the pdf $f_J(t)$. Nodes may have individual pdfs, with $f_{L,i}(t)$ and $f_{J,i}(t)$ describing the individual leave and join densities of node i. Note that if individual leave and join probabilities are assumed, $n_{\mathcal{A}}$ is still assumed to remain constant.

Nodes in $\mathcal{A}$ are assumed to have the same radio range and relay messages for each other to provide for multi-hop routing. Although message delays in $\mathcal{A}$ depend on the hop count of communication paths, for the sake of simplicity, I assume an average message delay δ_m for $\mathcal{A}$. Note that δ_m is not an upper bound for message delay, but rather a guideline describing the message delay of most messages if a path is available. A time-out value δ_{to} describes a reasonable period of time to wait for an expected message within a synchronous period of the system.

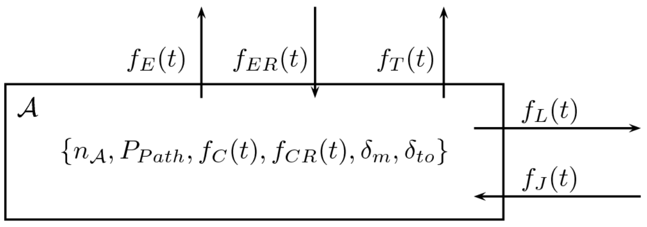

$f_E(t)$
$f_{ER}(t)$
$f_T(t)$
$\mathcal{A}$
$\{n_{\mathcal{A}}, P_{Path}, f_C(t), f_{CR}(t), \delta_m, \delta_{to}\}$
$f_L(t)$
$f_J(t)$

Figure 2.4: System Model.

Topology dynamics, mainly influenced by the mobility and link models, are captured in my system model by (i) the constant path probability P_{Path} of $\mathcal{A}$, (ii) by the pdf $f_C(t)$, describing the path duration for $\mathcal{A}$, and (iii) the pdf $f_{CR}(t)$ describing the probability that a broken path recovers after t. The probability that a broken communication path recovers is a conditional probability presuming that both communication partners remain in $\mathcal{A}$.

In addition to moving to other areas, nodes may disconnect from $\mathcal{A}$ forever if they experience a non-recoverable technical failure. The probability of this happening is described by the pdf $f_T(t)$. Recoverable failures causing a disconnection from $\mathcal{A}$ represent e.g. energy-related outages. The probability that a node disconnects due to a recoverable technical failure is described by pdf $f_E(t)$ and the density of energy-related outage times by pdf $f_{RE}(t)$. I use the subscripts E and RE here because I assume recoverable technical failures to be energy-related.

Figure 2.4 depicts the general idea of the system model. While the communication characteristics within $\mathcal{A}$ are described by the parameters $n_{\mathcal{A}}$, P_{Path}, $f_C(t)$, $f_{CR}(t)$, δ_m, and δ_{to}, the leave and rejoin probability of nodes is described by the pdfs $f_{ER}(t)$, $f_E(t)$, $f_T(t)$, $f_L(t)$, and $f_J(t)$.

The described system model is generic in the sense that it describes arbitrary MANET scenarios; to examine a concrete MANET scenario, the pdfs introduced here must be derived for the scenario under consideration. How these probabilities can be derived for a given scenario is shown in Chapter 4.

2.3.2 Failure Model

The failure model describes failures from a single node's perspective. Failures lead to situations where a node cannot communicate with another node anymore. A node in the system model described above can generally experience a *node failure* or a *communication failure*. Node and communication failures are defined as follows.

Node failure

A node failure describes all events that cause a node to disconnect from $\mathcal{A}$. Hence, cdf $F_N(t)$, the probability for a node to experience a node failure within time t, is given by the probability that (i) a node leaves $\mathcal{A}$, (ii) exhibits an energy-related failure or (iii) experiences a technical failure. Given the pdfs $f_L(t)$, $f_E(t)$ and $f_T(t)$ from the system model, $F_N(t)$ can be calculated by considering the complementary probabilities of the cdfs $F_L(t)$, $F_E(t)$ and $F_T(t)$ as

$$F_N(t) = 1 - \big[(1 - F_L(t)) \cdot (1 - F_E(t)) \cdot (1 - F_T(t))\big] \tag{2.1}$$

It is assumed that mobile nodes are equipped with some kind of stable storage that survives node failures. Hence, data written to stable storage is available after recovery from node failures.

Communication failure

A communication failure describes all events that lead to an outage of the communication between two nodes that are connected to $\mathcal{A}$. A communication failure causes the break of a communication path that was functional before and is induced by the dynamic network topology. The probability for a communication failure to happen within time t is given by the distribution of path durations described by the cdf $F_C(t)$, which is directly derived from the according pdf $f_C(t)$ provided by the system model. Hence, I denote the probability for a communication failure within time t by $F_C(t)$.

By $F(t)$ I denote the cdf of the *general failure* that either a communication or a node failure occurs until t, derived by considering the complementary probabilities of node and communication failures:

$$F(t) = 1 - \big[(1 - F_C(t)) \cdot (1 - F_N(t))\big] \tag{2.2}$$

From a single node's perspective, $F(t)$ describes the probability that communication with another node fails, because either the communication path breaks or because the communication partner disconnects from $\mathcal{A}$ within t.

2.4 Summary and Conclusion

In this chapter, I gave a short overview of the MANET research landscape and introduced MANET principles as well as related work important to this thesis.

These are multi-hop routing and broadcast schemes as well as basic modeling and simulation principles. I have also shown how this work integrates into research activities in the area of MANETs. At the time of writing, this work is one of the few that examines a problem on the application layer in MANETs.

From the general MANET characteristics, I extracted the system model and failure model of this work. Atomic transaction processing is analyzed in the following chapters based on these models. A unique characteristic of the system model in contrast to commonly proposed models is that I do not consider a MANET to be a closed system, but nodes are assumed to constantly leave and rejoin the network. This is modeled by considering one area $\mathcal{A}$ of a MANET defined by the AGB model. The numerous factors reducing reliability of communication in $\mathcal{A}$ are condensed in the pdfs $f_C(t)$, $f_{RC}(t)$, and the constant P_{path}.

The following chapter introduces principles of transaction processing and derives the application model of this work. Application and system models are then analyzed together in Chapter 4.

Chapter 3

Atomicity: Background and Preliminaries

This chapter provides an introduction to distributed atomic transaction processing and therefore introduces the application model investigated within this work. Based on the general principles of distributed transaction processing, the concepts of strict and semantic atomicity are introduced and formalized by defining different types of atomic commit problems. For each atomic commit problem, solvability is discussed and fundamental results available in the literature are presented.

For each atomic commit problem known protocols solving this problem are described and their applicability in MANETs is discussed. To give an overview of related work in the area of atomic commit processing, I will present some commit protocols recently proposed for infrastructure-based mobile networks. The discussion of minimal protocols solving certain atomic commit problems is the starting point for the examination of atomic commit in MANETs and motivates the general approach of probabilistic analysis of commit protocols followed by this work.

Besides presenting background information and related work of atomic commit processing, a major objective of this chapter is the formal definition of transaction models that are used within the remainder of this thesis to investigate abort and blocking risks of atomic transaction in MANETs.

The chapter is structured as follows: Section 3.1 introduces the basic transaction concepts, while Section 3.2 discusses different transaction models and atomicity notions. Section 3.3 formally defines important atomic commit problems and discusses their solvability in MANETs. In Section 3.4, important atomic commit protocols are presented solving these problems. Finally, Section 3.6 presents the transaction models used to examine atomic transactions in MANETs in the remainder of this work.

3.1 The Transaction Concept

Transactions are one of the most successful abstractions in information technology. Transactional systems make the developer's life easier by freeing him to explicitly deal with problems that stem from concurrent access to shared resources and by masking transaction and site failures that can occur at lower system levels, e.g. page failures, consistency violations, etc. This is achieved by grouping several computational tasks into a single consistent and reliable unit of work called a *transaction*, which is conceptually treated as a single operation masking failures of embraced operations and hiding their effects until the whole portion of work is considered to be correct.

In general, the notion of transactions is tightly bound to the ACID properties. Processing of a transaction is expected to meet the ACID properties (atomicity, consistency, isolation, and durability) [147, 80, 64]. Some transaction models weaken one or more of these properties, but generally the problem of guaranteeing ACID is inherent to all transactional systems. In short, the ACID properties are:

Atomicity is also known as the *all or nothing* property. This property requires that either all operations of a transaction are completed or none at all. To ensure atomicity, recovery mechanisms are required that undo executed operations or complete the remaining operations of a transaction if a transaction is aborted intentionally, e.g. due to consistency or concurrency violations (*transaction recovery*), or if a system failure occurs, such as media, process, or communication failures (*crash recovery*).

Consistency requires that a transaction always leaves a system in a consistent state and no inconsistent states are ever exposed. A state is consistent if all integrity constraints of a system are satisfied. Broadly speaking, the consistency property requires *correctness* of a transaction.

Isolation demands that not only the effect of a single transaction is correct, but also the concurrent execution of multiple transactions. If isolation is satisfied, the overall effect of a transaction schedule must be the same as if transactions had been executed in a serial order.

Durability ensures that modifications made on behalf of a successfully completed transaction persists even if the computer, or the medium on which the data is stored, subsequently crashes. Hence, durability ensures that no information is lost due to failures.

In its most simple form a transaction is completely executed on one server that hosts all of the required data and functionality. In a distributed system, functionality and data objects are allocated to different network nodes. A transaction which is processed among multiple computer nodes is naturally decomposed into portions of work by the allocation of functionality and data objects to the different network nodes. Hence, the internal structure of such a transaction is determined by the location of functionality or data within the network. All work processed at one node forms a subtransaction, also denoted as a *transaction branch*. Guaranteeing ACID for such a distributed transaction is considerably more difficult than in the centralized case, because coordination among sites processing transaction branches is required to ensure ACID at a global level. In the remainder of this work, I am solely concerned with such distributed transactions, henceforth simply called *transactions*.

To ensure atomicity at the global transaction level, Atomic Commit protocols (ACP) establish agreement on the termination decision (abort or commit of local transaction branches) among involved sites. As this work focuses on atomicity in MANETs, an understanding of the distributed atomic commit problem solved by an ACP is fundamental. I will describe this problem in detail in Section 3.3.

While the lack of an omniscient controlling entity in a distributed transaction also makes it difficult to ensure other ACID properties, e.g. providing isolation becomes more complicated as dependencies between concurrent distributed transactions have to be considered. However, I argue that these problems are only relevant at high transaction load, which is commonly not assumed in a MANET, since here transactions are processed in a peer-to-peer fashion where the transaction load is distributed among nodes. In contrast, Atomicity is the property that is directly affected by volatile communication and frequent node failures in MANETs, independently of the transaction load.

While the transaction concept was initially developed in the context of online transaction processing (OLTP) in database systems, which I call the *traditional transaction model*, transactions are also successfully applied in other application domains, such as CAD/CAM applications or transactional workflows. In these scenarios, atomic operations are not necessarily database operations but arbitrary method invocations or tasks that are assigned to a functional component or even a human. Numerous transaction models for these so-called *non-traditional* applications have been proposed in the literature posing a different atomicity semantic than known from OLTP. In contrast to OLTP transactions, non-traditional transactions are possibly long-living like transactional workflows and may show more complex nested or multi-level structures than the flat distributed transaction introduced above.

3.2 Transaction Models

In the following, I will briefly describe some selected transaction models important for this work. Besides the traditional transaction model, general concepts of so-called advanced transaction models (ATMs) are presented, with a focus on the different termination dependencies of subtransactions in these models that cause the atomic commit problem.

3.2.1 Traditional Transaction Model

The traditional flat distributed transaction model is by far the most relevant model in practice and is manifested in the de-facto standard *X/Open Distributed Transaction Processing (DTP)* [40]. It is basically concerned with transactions among distributed databases. A strong school of thought exists for this transaction model, providing solutions to implementation issues on concurrency and atomicity [57, 13, 153, 14].

The structure of a traditional transaction is solely determined by the allocation of accessed resources in the network. All units of work executed at a certain node are assembled into one subtransaction assigned to this node. Hence, one characteristic is that a traditional distributed transaction consists of one nesting level only. The transaction framing these subtransactions is called *global* *transaction* in X/Open terminology. In the X/Open standard architecture, operations of a subtransaction are received by a resource manager (RM), which must reside on each network node. The RM guarantees ACID at a local level, i.e. at the level of subtransactions. To ensure atomicity of the global transaction, the X/Open standard proposes an additional functional component called the *transaction manager* (TM), responsible for coordinating atomicity of the global transaction. Hence, X/Open proposes a strict separation of the execution coordinator and the TM. The role of the execution coordinator is taken by an application that executes operations of the global transaction at remote sites by means of RPC calls.

The most important characteristic of the traditional transaction model in the context of this work are its strict inter-transaction dependencies[1]. All transaction branches $S_0, \ldots, S_n$ of a global transaction T are bound to each other by termination dependencies, i.e. $\{\forall i, \forall j \mid S_i$ can only commit or abort until S_j aborts or commits$\}$. The dependencies between each transaction branch S_i and T are given by abort and commit dependencies, i.e. $\{\forall i \mid$ if S_i aborts, then T must be aborted$\}$ and $\{\forall i \mid$ if T aborts, then S_i must be aborted$\}$ and $\{\forall i \mid$ if S_i commits, then T must commit$\}$.

The termination dependencies among subtransactions induce a fundamental problem: the termination dependencies implicate that a transaction participant cannot unilaterally decide whether to abort or commit its local transaction branch if the state of another S_i is unknown. Therefore, the basic problem to be solved by an Atomic Commit Protocol (ACP) is to derive the state of remote transaction branches in cases of communication and site failures.

The abort dependency between subtransactions and the global transaction T force a transaction to be completely aborted if any subtransaction fails. Hence, even if only a single operation of a transaction fails, the successful parts of the transaction are also lost.

The dependencies defined above describe an atomicity semantic called *strict atomicity*. They are strict in the sense that no subtransaction S_i is allowed to terminate as long as it is not guaranteed that every other subtransaction terminates. To resolve the termination dependencies of strict atomicity, a local RM must be able to announce to the other transaction branches that it is *prepared* to commit its local branch, without actually committing that branch. Hence, another important characteristic of the traditional transaction model is that it requires RMs to provide such a *prepared state*.

The strict abort dependencies of the traditional model causes high rates of transaction abort in situations where failures occur frequently. Frequent failures may occur due to unreliable nodes and communication channels like in MANETs or because transactions are running very long[2]. So-called long-lived transactions motivated the development of advanced transaction models (ATMs), which diminish the strict abort and termination dependencies of the traditional model in order to tolerate failures of transaction branches without being forced to abort the complete transaction.

[1]Termination, commit, and abort dependencies are here understood as proposed by the ACTA framework [37].

[2]In [54] Jim Gray calculates that the probability of a deadlock (transaction failure) increases with the fourth power of the transaction size.

3.2.2 Advanced Transaction Models

ATMs are designed to be more failure resilient, because the effect of a failure is limited to smaller entities than the global transaction, which is desirable in volatile environments like a MANET.

Generally, ATMs provide special constructs that exploit the semantic relationship of a subtransaction with other subtransactions, allowing for more advanced failure handling than the strict model. ATMs emerged in the early 90s, but have not found their way into the real world and have mainly remained an academic topic. In contrast, workflow-management systems stem from real world requirements demanding for transactional features but lack the theoretical foundation that ATMs provide. Currently, research tends to combine both fields, I refer the interested reader to [22, 141, 52, 134]. Important to this thesis is that with ATMs, a new atomicity notion called *semantic atomicity* has been proposed, weakening the strict atomicity notion introduced above.

The handling of faulty subtransactions proposed by the five ATMs: (i) Sagas [50], (ii) the ConTract model [150], (iii) Flex Transactions [47], (iv) S-Transactions [148], and (v) the Multi-Level transaction model [152]; can be summarized by four approaches to cope with abort or unknown states of a subtransaction. Based on the semantic of the subtransaction, the following is proposed in ATMs:

1. Simply ignore the failure of a subtransaction and continue to execute the remaining correct subtransactions. Subtransactions allowing for such a scheme are called *non-vital* subtransactions.

2. Retry the aborted subtransaction (forward recovery). In cases where a temporal failure caused the abort of a subtransaction, it may be possible to re-execute the subtransaction. A subtransaction allowing for this semantic is called a *retryable* subtransaction.

3. Initiate another alternative subtransaction that implements a semantically similar action (*contingency subtransaction*).

4. Abort the global transaction and all running subtransactions as in the traditional model (*vital subtransaction*).

Options (1–3) allow a transaction to be committed successfully in spite of failures, while (4) is the traditional approach. Although ATMs and their strategies (1–3) were initially developed to overcome transaction failures due to concurrent transaction processing in long-lived transactions, these strategies naturally fit into a distributed setting where communication and node failures are assumed rather than transaction failures. For example, if a remote site executing a subtransaction no longer answers, the execution coordinator may execute a contingency transaction on another reachable node and hence commit the global transaction. However, an important aspect in the context of atomicity is that the global decision must still be consistent and every node executing a transaction branch must follow the global decision, i.e. also a non-vital subtransaction is not allowed to commit if the global decision is to abort. In other words, a non-vital subtransaction is non-vital for a global commit decision but vital for global atomicity. Hence, schemes 1–3 aim at allowing a global transaction to be committed in the case of failures, but does not ease the problem of providing atomicity.

In a MANET, assuring strict atomicity is susceptible to blocking, as subtransactions cannot unilaterally decide on commit, but require communicating over unreliable communication links to learn about the state of the other subtransactions. Local commit must be delayed until all transaction branches are ready. ATMs approach this undesirable delay with the concept of *semantic atomicity* described in the following.

Semantic Atomicity

The basic idea of semantic atomicity is to allow subtransactions to terminate independently without coordinating with other subtransactions by removing the termination dependencies among transaction branches. Resources held by a transaction branch can be immediately freed as soon as a local decision has been made.

Thus atomicity is given up for a weaker notion of atomicity (*semantic atomicity*) first introduced by [49] and adapted to distributed environments in [94, 158]. The basic concept of semantic atomicity is applied in most ATMs and is sometimes also called *relaxed atomicity* or *semi-atomicity* [158].

Here, atomicity is not provided on the physical level, but rather on a semantic level. The idea is to take temporal inconsistencies into account but eventually ensure atomicity semantically on the global level. To reach an unanimous decision a posteriori (after some participants have committed their local transaction branch), a recovery mechanism is required that can reverse the decision of a local RM and hence undo the effects of a committed transaction. The corresponding recovery concept is called *semantic recovery* and is based on the idea of *compensating transactions* [81, 49].

Compensating transactions semantically undo the effect of an already committed transaction. Hence, in semantic atomicity subtransactions are allowed to unilaterally commit, without considering the other subtransactions and the global transaction state. If the local decision opposes the global decision, a compensating transaction must be executed. Therefore, every subtransaction S_i is associated with a compensating transaction called CS_i that systematically removes all the effects of S_i.

If a strict abort rule is applied, i.e. if no failure is tolerated, the following inter-transaction dependencies must be maintained in a semantic scheme: (i) $\{\forall i \mid$ if S_i aborts, then T must be aborted$\}$, (ii) $\{\forall i \mid S_i$ aborts if still executing and T aborts. S_i can decide on commit if T has not aborted$\}$, (iii) $\{\forall i \mid CS_i$ is not allowed to begin work until S_i has committed. Hence, CS_i can only be started after S_i is committed$\}$, (iv) $\{\forall i \mid CS_i$ is only allowed to begin work if T has aborted. If T commits, CS_i is never executed$\}$, and (v) $\{\forall i \mid CS_i$ is forced to commit if T aborts. Hence, if T aborts, CS_i is started if not yet executing and is not allowed to abort$\}$.

Note that dependency (i) can be substituted with advanced constructs to tolerate failures as described above.

A major implication of the semantic atomicity scheme on the execution environment is that RMs are not required to implement a *prepared* state as in strict atomicity, which makes it especially appealing for non-database applications.

However, implementing the semantic scheme in reality poses several other problems. The main problem is to guarantee isolation in presence of temporal inconsistency as shown by the following example: consider the global transaction T that is composed of two subtransactions S_0 and S_1. Now, S_0 manipulates a data item a, while S_1 writes item b. Assume that the RM executing S_1 decides to commit while S_0 is aborted by its controlling RM. Another transaction S_i reading a and b is now exposed to an inconsistency. If the global decision on T is to abort, a compensating transaction CS_1 is executed undoing the effects of S_1, the problem is then to deal with the depending transaction S_i that has seen the effects of S_1 and S_0. One option would be to compensate S_i, as its effects are based on inconsistent premises. But this approach would result in the cascading execution of compensating transactions, which must not happen. Delaying S_i until S_1 receives the final decision of the coordinator would contradict the initial idea of semantic atomicity, as this would lead to a strict scheme. Hence, the main goal is to undo the effects of transactions using compensating transactions but leave the effects of dependent transactions intact.

Implementation of a general approach to maintain the possibility for compensation for arbitrary transactions is hard. Whether the use of compensating transactions is feasible mainly depends on the given application scenario. A general model proposed by academia is the *soundness* criterion [94, 81]. Soundness is achieved if a compensating transaction CS_i undoes the effects of transaction S_i cleanly, leaving the effects of all transactions depending on S_i denoted by the set $dept(S_i)$ intact. It is shown in [81] that a history is sound if the compensating transaction commutes with every transaction in $dept(T)$. Several scholars have proposed protocols to assure soundness, e.g. the Polarized Protocol proposed in [94] or ε-Serializability [120]. It is obvious that enforcing soundness will still require transactions to be delayed or to be rejected. Generally, all dependent transactions using conditional constructs are problematic. Hence, whether a protocol enabling soundness is beneficial depends on the transaction load and type of depending transactions, which depends on the application and MANET scenario under investigation.

Semantic atomicity is an appealing concept to increase the autonomy of transaction participants, because unilateral decision making is now possible. The price to pay is the overhead to maintain soundness and the requirement to provide compensating transactions. This concept is tightly bound to application semantics and relies on the assumption that compensating operations can be found, which is not always the case (e.g. firing a missile is obviously not a compensatable action). Additionally, designing compensating transactions increases the development cost of a transactional systems. In fact the need to implement compensating transactions requires the developer to explicitly deal with failure handling, which in a way contradicts the initial intent of the transaction concept.

However, there are a lot of application scenarios where compensation actions are found intuitively and therefore semantic atomicity is a natural fit. E.g. in a mission-control scenario, a rescue unit that has committed itself to a mission can easily compensate this action by dropping its commit to a mission and continuing to answer requests of other missions, and no dependent transactions have to be considered in this application.

3.2.3 Summary - Transaction Models

In this section, I have introduced the traditional transaction model and the general ideas behind advanced transaction models (ATMs) and their different termination dependencies. These dependencies defined the notions of *strict* and *semantic atomicity*. Abstractions of both models are later used in this chapter to define the transaction models examined in a MANET environment.

Since communication and node failures are assumed to occur frequently in MANETs, the failure handling of ATMs can be considered beneficial for MANETs with regard to transaction abort. The inherent problem of ensuring termination dependencies among transaction branches to assure strict and semantic atomicity remains for both models. While a communication failure does not necessarily cause abort in an ATM, the RM executing the subtransaction which state is unknown has eventually to follow the global decision. Removing uncertainty about the global decision is a central problem in both models in presence of node and communication failures.

It is the central focus of this thesis to examine the problems that occur when the termination dependencies of strict and semantic atomicity have to be assured in a MANET environment. These are generally maintained by a coordinating entity that processes an ACP solving the atomic commit problem. While in this section the general atomicity semantic was introduced together with a general discussion of transaction models, the resulting atomic commit problems are defined formally in the following.

3.3 Atomic Commit Problems

To examine strict and semantic atomicity in MANETs, the theoretical foundation of the underlying atomic commit problems is discussed first. The objective of this section is to give a formal description of the atomic commit problem and to discuss its general solvability, before protocols solving the different commit problems are presented.

Depending on the transaction model, i.e. whether strict or semantic atomicity is considered and how RMs process local transaction branches, different atomic commit problems are distinguished. Namely, I will consider the standard *atomic commit* problem, the *dictatorial atomic commit* problem, and the *semantic atomic commit* problem.

At the heart of all commit problems lies the *blocking problem*, which describes the situation that a participant is forced to wait for an unforeseeable length of time to be able to learn the global transaction decision.

The atomic commit problem is a fundamental coordination problem extensively examined by the database and distributed systems community. In the following solvability results proposed in the literature for the asynchronous and partially synchronous system models are briefly reviewed and their impact on this work is discussed.

3.3.1 Atomic Commit (AC)

The standard atomic commit problem, mostly just called the *atomic commit problem*, stems from the traditional transaction model and its termination dependencies. From the informal description of transaction dependencies described in Subsection 3.2.1 four conditions are derived to define this problem. These conditions are classified as either safety or liveness conditions. Safety conditions describe what is allowed to happen, while liveness conditions describe what must happen to achieve progress of the agreement process. Articles describing the derivation of these conditions in more detail are for example [63, 13, 61]. The safety conditions of the atomic commit problem are:

AC1 (*Uniform-Agreement*): No two participants reach different decisions.

AC2 (*Uniform-Validity*): If a participant decides to commit, then all participants have voted for commit.

AC3 (*Stability*): A participant cannot reverse its decision after reaching agreement.

Note that *AC1–AC3* reflect the termination dependencies presented in Section 3.2.1. To ensure that a solution of the atomic commit problem makes progress, the liveness condition *AC4* is defined:

AC4 (*Non-Triviality*): If all participants can commit and there are no failures, then every correct participant decides to commit.

The main intention of condition *AC4* is to prevent the unexpected solution of unilateral abort of all participants, i.e. that all participants always decide on abort. The conditions *{AC1, AC2, AC3, AC4}* define the atomic commit (AC) problem.

The AC problem defined so far does not require all processes to decide. Hence, another liveness condition requiring progress of the commit process called the *non-blocking* property *AC5* is posed:

AC5 (*Non-Blocking*): All correct participants reach a decision.

Non-blocking here means, that no correct participant must wait for failed participants to recover in order to reach a decision. Conditions *{AC1, AC2, AC3, AC4, AC5}* define the non-blocking atomic commit (NB-AC) problem [61]. I will show later that the NB-AC problem is not solvable, due to the strict definition of *AC4*. To allow solvability in such an environment, the alleviated Non-Triviality property *AC4** is defined:

AC4* (*Non-Triviality):** If all participants can commit, and no participant is suspected to be failed, then every correct participant reaches a commit decision.

The difference between *AC4* and *AC4** is that *AC4** allows to decide on abort if all participants can commit. This is because commit must only be decided, if *"no participant is suspected to be failed"*. The unexpected solution to always decide on abort is still prohibited with *AC4**. For a more detailed discussion of *AC4** I refer the reader to [39]. The problem defined by *{AC1, AC2, AC3, AC4*, AC5}* is called the non-blocking weak atomic commit (NB-WAC) problem. And analogously, the weak atomic commit (WAC) problem is defined by *{AC1, AC2, AC3, AC4*}*.

Solvability of NB-AC and NB-WAC

Reasoning about the AC problems is commonly based on the standard asynchronous system model with crash failures [42, 48], where communication is assumed to be reliable but process speeds and message delays are generally unbounded. In this system model, AC and WAC are solvable, because blocking is allowed if precise knowledge about the state of other participants is missing due to a failure. Solvability is also indicated by the existence of protocols such as 2PC that are known to be correct. The asynchronous model assumes less about timeliness than the system model of this work which is assumed to be partially synchronous. Therefore, results of the asynchronous system model are portable to the MANET system model of this work and AC as well as WAC are solvable in a MANET. I refer to [97] for a proof and discussion of the relation between the asynchronous and partially synchronous model.

NB-AC was proven not to be solvable in the asynchronous system model as first shown in the fundamental result of [137]. This major theorem can be found in most textbooks on transactions like [153, 13, 14] and states that no commit protocol that solves NB-AC can exist in the asynchronous model if multiple site failures can occur. It is shown that the main requirement to solve NB-AC is to allow for independent recovery of transaction participants in the presence of multiple failures. The proof that independent recovery cannot be guaranteed in the asynchronous system model is based on the observation that for a protocol that provides independent recovery, there must always exist a time slot in which the simultaneous failure of two participants leads to inconsistency [137]. In a MANET, such a failure situation is generally given if a communication failure occurs. If the communication path between two nodes breaks, both nodes cannot distinguish whether the other node suffered a node failure and therefore abort is indicated, or if a communication failure occurred and messages from the other node are simply lost. Hence, a single communication failure leads to a situation which is indistinguishable from a real simultaneous node failure. Based on this reasoning, there exists no distributed commit protocol that can guarantee independent recovery if communication failures may occur, and therefore NB-AC is not solvable in a MANET.

While the minimal assumptions of the asynchronous system model are convenient for theoretical proof and algorithm development, distributed systems of the real world show periods of synchrony and hence show more timing guarantees than assumed in the asynchronous model. To examine agreement problems in a more realistic way two main concepts to consider synchronous periods within a distributed system have been defined: (i) partially synchronous system models have been defined and classified in [44, 46]; and (ii) the asynchronous system model has been enhanced by the theoretical concept of unreliable failure detectors in [36]. For the discussion of solvability of NB-AC and NB-WAC, I will follow the reasoning based on the concept of unreliable failure detectors.

An unreliable failure detector can be thought of as an oracle that gives hints on the failure state of processes, which are possibly false suspicions. It can be shown that in the partially synchronous model unreliable failure detectors with certain accuracy and completeness properties can be implemented [35] using time-outs. Several non-blocking agreement problems that are not solvable in the asynchronous system, e.g. consensus [48, 113], are found to be solvable in an asynchronous system with failure detectors. E.g. [36] and [35] showed that consensus is solvable in the asynchronous system model with unreliable failure detectors that meet *weak completeness* and *eventual weak accuracy* and a majority of non-faulty processes. Here, weak completeness means that eventually every faulty participant is permanently suspected by some correct participants. Eventual weak accuracy requires that there is eventually a correct participant that is never suspected.

However, the availability of unreliable failure detectors does not help to solve NB-AC. The strict definition of *AC4* demands precise knowledge about failures, which cannot be provided by unreliable failure detectors. Weakening *AC4* by requiring commit only if *"no participant is suspected to be failed"* allows to abort the global transaction also if all subtransactions could be committed. It is exactly this property that allows to solve NB-WAC in a partially synchronous model.

Solvability of NB-WAC is shown in [60] by reducing NB-WAC to consensus, which allows solvability results of consensus proven in [36] and [35], to be applied to the NB-WAC problem. [60] proved solvability of NB-WAC by constructing a protocol that uses multiple instances of a consensus algorithm and hence solves NB-WAC if a majority of participants does not crash, i.e. if a majority of participants can reach each other.

Based on the results of [60], it can be concluded that NB-WAC is solvable in a MANET environment if a majority of transaction participants does not suffer a node failure and can reach each other. Note that in MANETs, partitioning frequently causes a situation where a majority of participants is not available if they are residing in different partitions. However, in a partition with a majority of participants a decision is reached, while minorities in other partitions remain blocked.

Note that the theoretical result on NB-WAC solely states solvability, while applicability of solutions is not considered here. Solvability is based on the rationale that eventually a synchronous period occurs that allows the transaction to be terminated with at least a majority of participants. However, for a practical solution the time required to eventually reach agreement is crucial. Note that until agreement on the termination decision has been finally achieved, transaction participants remain uncertain about the global decision and are in a state similar to blocking.

While reasoning about solvability is interesting and attracted numerous scholars, its practical impact is questionable, since the most commonly used commit protocol (2PC) solves AC or WAC only and shows to be sufficient in practice.

3.3.2 Dictatorial Atomic Commit (DAC)

While in the AC problems presented above, the verification of ACID for local transaction branches is embedded into the AC problem, the basic idea of Dictatorial Atomic Commit (DAC) is to exclude the verification of the ACID properties from the commit problem. DAC presumes that the ACID properties are already guaranteed by participants at commit time. This means that, in contrast to the AC problem, participants are not required to explicitly announce that they will move into the prepared state. The main idea of DAC is to derive the vote of a participant implicitly from the way participants execute their transaction branch. If all operations have been successfully executed and acknowledged to the coordinator, the coordinator implicitly assumes that the remote RM can ensure ACID locally and simply dictates the global decision when all operations of all subtransactions are finally processed. Hence, a participant is implicitly assumed to move into prepared state every time it acknowledges an operation. The coordinator can then simply dictate the global decision without being required to collect any votes.

Protocols solving the DAC problem provide atomicity, but the problem solved is slightly different from the AC problem discussed above. While AC is defined by the uniform validity condition *AC2* stating: "*If a participant decides on commit, then all participant have voted for commit*", in DAC, the uniform validity condition is based on the initial values of participants that are known by the coordinator before the protocol is started. The initial values of participants are either commit if the last operation of a branch has been successfully executed and acknowledged or abort if a negative acknowledgment was received. The uniform validity condition in DAC is given by:

AC2_DAC (*Uniform Validity*): The global decision must be an initial value.

Hence, the DAC problem is defined by the conditions *{AC1, AC2_DAC, AC3}*. Note that the non-triviality property is not required here because *AC2_DAC* ensures that if all participants have the initial value commit, then the global decision must be commit. A weaker notion of the DAC problem is defined, by an alleviated variant of *AC2_DAC* that allows for an abort decision although all participants are correct.

AC2_DAC* (*Uniform Validity):** The global decision must be commit if no participant is suspected to be failed and the initial values of all participants is commit.

Requiring non-blocking leads to the non-blocking dictatorial commit (NB-DAC) and non-blocking weak dictatorial commit (NB-WDAC) problem, defined by *{AC1, AC2_DAC, AC3, AC5}* and *{AC1, AC2_DAC*, AC3, AC5}* respectively.

Solvability of NB-DAC and NB-WDAC

Reasoning about solvability of the dictatorial atomic commit problem is based on similar thoughts as those for the AC problem. In the synchronous model with crash failures, the lack of precise knowledge about the initial value of participants precludes a solution of NB-DAC similar to NB-AC. This is easy to see in the following situation: if a participant successfully executes and acknowledges its last operation and the acknowledgment message is lost due to a communication failure, then the coordinator cannot decide on the initial state of the participant and a correct decision cannot be guaranteed. NB-WDAC is not solvable in a purely asynchronous system models due to the impossibility for independent recovery in presence of multiple node failures as described in Section 3.3.1.

Similar to NB-WAC, NB-WDAC can be reduced to consensus as shown in [4], which allows for a solution in the partially synchronous system model with unreliable failure detectors implementing weak completeness and eventual weak accuracy if a majority of participants can reach each other. Hence, it can be concluded that DAC and NB-WDAC are generally solvable in a MANET if a majority of participants can reach each other.

3.3.3 Semantic Atomic Commit (SAC)

As described in Section 3.6.3, semantic atomicity allows RMs to decide on commit of their local transaction branch autonomously. No coordination with other RMs that execute subtransactions of the same global transaction or with the coordinator is required. This implies, that no prepared state is required since all transaction branches are directly transferred from executing to abort or commit state.

Semantic atomic commit (SAC) obviously violates the *AC1* property, as participants of a global transaction are allowed to decide differently on their local transaction branches. Reaching an unanimous decision a posteriori also violates the *AC3* (Stability) property of atomicity, which causes some authors to talk about non-atomic transactions here. Generally, there is no commonly accepted definition of the semantic atomic commit problem and most scholars contributing to this area tend to pose their own definitions, like semi-atomicity in [158] or relaxed-atomicity in [94]. I give a definition of SAC conditions here, that reflects the possibility of temporal inconsistency by using a vague "eventually"-formulation. I modify *AC1* and *AC3* as follows:

AC1_SAC (*Uniform Agreement*): Eventually all participants decide either on commit or to abort.

AC3_SAC (*Stability*): A participant cannot reverse its decision after eventually reaching agreement, while unilateral commit decisions can be revoked.

The condition *AC1_SAC* allows for temporarily inconsistent decisions of participants, because *AC1_SAC* requires only that *eventually* uniform agreement is reached. Similar *AC3_SAC* only requires that the eventual agreement is not reversed, while a unilateral commit decision is allowed to be revoked.
Validity of the global decision is defined similar to DAC, since the acknowledgment of the last operation is interpreted as an implicit vote by the coordinator. Hence, SAC is defined by conditions *{AC1_SAC, AC2_DAC, AC3_SAC}*.

While blocking as in AC and DAC cannot occur in SAC, another situation similar to blocking may occur, which I call *extended uncertainty.* An extended uncertainty situation is a situation where participants remain uncertain about the global decision while they have already reached a decision locally and are forced to wait until failures of other participants recover before they can learn about the global decision. To anticipate this situation I define condition *AC5_SAC*:

AC5_SAC (*Extended Uncertainty*): All participants eventually learn about the global decision and adjust their local decision accordingly.

In fact the semantic of condition *AC5_SAC* is similar to the non-blocking condition *AC5*, with the difference that *AC5_SAC* is concerned with the adjustment of a local decision already made according to the global decision, while *AC5* is concerned with a local decision not derived yet.

I call the commit problem defined by {*AC1_SAC, AC2_DAC, AC3_SAC, AC5_SAC*} eventually certain semantic atomic commit (EC-SAC). A weakened notion analogous to NB-WAC and NB-DAC is defined by {*AC1_SAC, AC2_DAC*, AC3_SAC, AC5_SAC*} and called eventually certain weak semantic atomic commit (EC-WSAC).

Solvability of SAC, EC-SAC and EC-WSAC

SAC is quite similar to the DAC problem: in both problems, blocking as well as extended uncertainty is caused by uncertainty about the global decision. Hence, argumentation about solvability of SAC and EC-SAC is similar to DAC.

SAC is solvable in the asynchronous and partially synchronous system model, since progress of participants is not required. The EC-SAC problem is not solvable in the asynchronous system model and in the partially synchronous model with unreliable failure detectors, because precise information about the failure state of participants is not available with unreliable failure detectors. EC-WSAC is solvable in the partially synchronous model with unreliable failure detectors and a majority of correct processes, as similar to NB-WDAC it is reducible to the consensus problem.

Hence, in a MANET the EC-WSAC problem is solvable if a majority of participants is reachable, i.e. if a majority of participants remains in one partition.

3.3.4 Summary - Atomic Commit Problems

This section defined atomic commit problems that so far have been analyzed by the distributed system and database research community. I presented some reasoning about the solvability of these problems in a MANET. For the sake of clarity the theoretical proofs have been cited only and have not been described in detail here. It was shown that the non-blocking variants of atomic commit are only solvable in a MANET if a majority of participants can reach each other, while participants residing in partitions with a minority of participants remain blocked or uncertain.

Hence, blocking cannot be completely avoided and applications deployed to a MANET have to cope with blocking situations in small partitions. The question to be answered now is how efficient protocols solve the non-blocking variants of atomic commit problems. To answer this question, I will present the important commit protocols proposed in the literature in the following.

3.4 Atomic Commit Protocols

In this section, atomic commit protocols (ACPs) solving the commit problems described in the previous section are described and evaluated according to their applicability in MANETs.

To evaluate the applicability of an ACP for MANETs its time and message complexity are especially important. Sending messages in MANETs is expensive in terms of required energy. With increased time complexity of a protocol, the probability of a communication or node failure to occur during protocol execution increases. Hence, the ideal ACPs for MANETs show small message and time complexities.

The log-complexity, i.e. the number of required forced-write log operations, is not considered here. Log-complexity is an important evaluation criterion in fixed environments where a write operation on concurrently accessed storage causes a considerable transaction delay. However, in a MANET environment where a high transaction load on a mobile node is not assumed, log-complexity is not considered to be a relevant factor and will therefore be omitted in the following descriptions of ACPs.

Especially important to estimate the blocking risks of an ACP is the so-called *window of uncertainty* which is inherent to all ACPs and closely related to the time complexity of protocols. By window of uncertainty I denote the period, where a participant is uncertain about the global decision and a communication failure with the coordinator or a node failure would cause blocking or extended uncertainty of the participant. The size of the uncertainty window is denoted by ΔU.

Hence, to evaluate the blocking risk of a protocol, the size of ΔU has to be considered. In most ACPs, ΔU increases when there are failures. I therefore define ΔU_{min} as the size of the uncertainty window in the failure-free case and ΔU_{max} for the situation where failures increase the uncertainty window.

In the following, the basic variants of protocols proposed to solve the WAC, WDAC, NB-WAC, and SAC problems, including basic recovery strategies to compensate for blocking situations, are presented.

3.4.1 Protocols solving AC and WAC

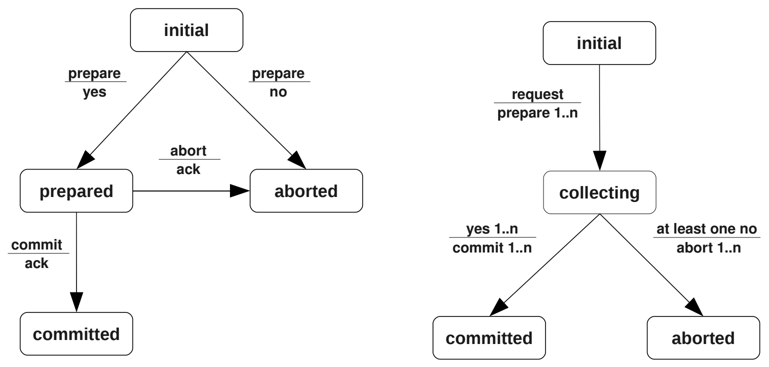
The most often used protocol in practice solving AC and WAC is the Two-Phase Commit (2PC) protocol [53] and its optimizations. 2PC is adapted by the major transaction standards DTP by X/Open [40] and OTS of OMG [110]. As the 2PC protocol is well known, I will only give a very brief description of the protocol in the following that focuses on termination and restart protocols as well as on blocking situations.

3.4.1.1 Two-Phase Commit Protocol (2PC)

2PC is initiated and controlled by a central transaction manager that models the global transaction to commit in one of the four states: *initial*, *collecting*, *aborted*, or *committed*. Transaction branches controlled by participants are either in the *initial*, *prepared*, *committed*, or *aborted* state as shown in Figure 3.1(a). As the name of the protocol indicates, it consists of two phases: (i) a voting phase where votes of participants are collected; and (ii) a decision phase where the decision is derived by the coordinator and distributed to participants. The voting phase is initiated by the coordinator by sending a *prepare* message to every participant. After issuing the *prepare* message, the coordinator immediately transits from the *initial* to the *collecting* state, as depicted in Figure 3.1(b).

A participant receiving a prepare message in initial state checks whether the ACID properties for its local transaction branch are assured and answers with an *OK* or *No* vote. If the participants vote is *No*, it aborts the local transaction and transits into the *abort* state. Otherwise the participant transits into the prepared state (see Figure 3.1(a)).

By evaluating the votes of all participants, the coordinator derives the global decision according to the following rules: if all participants have sent an *OK*



(a) 2PC state transitions of participants.

(b) 2PC state transitions of coordinator.

Figure 3.1: 2PC protocol state transitions.

message, the coordinator decides on commit and issues a *commit* message to every participant. If the coordinator receives at least one *No* message from a participant, it decides on global abort and sends an *abort* message to every participant. Participants proceed according to the coordinator's decision and either move their transaction from the prepared state to the committed state or from prepared to abort. This rule solves the AC problem. The WAC problem is solved, if the coordinator also decides on abort if a vote is missing, i.e. decided by a time-out.

The protocol description above describes the failure-free case of 2PC. In a MANET, node or communication failures can hit any node at any time during protocol execution. A failure during protocol execution leads to the execution of a *termination protocol* at all participants that did not suffer a node failure. A node that suffered from a node failure executes a *restart protocol* when recovering from its failure. During the restart protocol, a recovering participant examines its local log entries[3] to learn about the state of unfinished transactions at failure time. If a transaction in prepared state is found, the participant cannot decide on this transaction independently and is blocked.

Since time-outs are the only failure detectors available in MANETs, they are used to trigger termination protocols. Termination and restart protocols of participants are susceptible to blocking if they are initiated while the participants' transaction is in the prepared state, as in this situation the participant cannot decide on its transaction branch autonomously. Such blocking situations are defined in the following and used within the remainder of this thesis.

3.4.1.2 Blocking Situations in 2PC

A participant experiences a blocking situation when a failure occurs while it is in the prepared state. This stems from the fact that by entering the prepared state a participant gives up autonomy and control on its local transaction branch in favor of the coordinator. With handing over control to the coordinator, the

[3]The required log writes of 2PC have been omitted in the description of 2PC here. I refer to [14] for a detailed description of the forced-write log operations required in 2PC.

participant enters its window of uncertainty. Two blocking situations caused by failures defined in the system model of this thesis can occur:

Blocking caused by a participant failure Describes the situation, where a failure of a participant leads to blocking of this participant. This happens if a participant suffers from a failure while it is in its window of uncertainty. A communication failure with the coordinator during uncertainty prevents the global decision of the coordinator from being delivered. A node failure of the participant requires execution of the restart protocol at reconnection. While the participant is detached from $\mathcal{A}$ it is blocked; it remains blocked if it cannot reach the coordinator at the time it reconnects to $\mathcal{A}$[4]. In the remainder of this work, I denote this situation by **blocking (i)**.

Blocking caused by a coordinator failure Describes the situation, where a node failure of the coordinator leads to blocking of participants. This happens if the coordinator experiences a node failure, while participants are in their window of uncertainty (i.e. in the prepared state). All participants that are in the prepared state are then blocked, because they cannot receive the global decision from the coordinator. Depending on when exactly the node failure of the coordinator occurs, some participant may have received the global decision or all participants are uncertain. In the remainder of this work, I will call this situation **blocking (ii)**.

In the literature, blocking caused by a coordinator failure is considered to be the more severe blocking situation, because here possibly multiple participants are blocked and have to wait until the coordinator recovers from failure. It is one of the main contributions of this thesis to examine which blocking situation is actually relevant in MANETs. I will show in Chapter 4 that in MANETs, blocking (i) is more relevant than blocking (ii).

A common scheme to allow participants to leave the blocking state is *cooperative recovery* proposed with 2PC in [41]. In this scheme, a blocked participant queries all other participants in order to learn about the global decision within its termination or restart protocol. This is feasible because in the case of a blocking situation caused by a communication failure (blocking (i)), there is a chance that another participant has not suffered from a communication failure with the coordinator and thus is not uncertain about the global decision. Such a node is a potential cooperation partner for the blocked participant. In the case of a blocking situation caused by a coordinator failure, the chance to find another certain participant is also given, as there might be participants that did not receive the prepare message and hence did not move into uncertainty. I will show in Chapter 4 that cooperative recovery is a very efficient scheme to compensate for blocking in a MANET. The only requirement of cooperative recovery is that all participants know each other.

3.4.1.3 Evaluation of 2PC

With n participants, the 2PC protocol has a message complexity of $4n$ in the failure-free case if acknowledgments for commit messages are issued by partic-

[4]$\mathcal{A}$ is the MANET considered within the AGB mobility model as assumed in the system model of this work described in Section 2.3.1

ipants. Omitting acknowledgments leads to a message complexity of $3n$. If the coordinator and one participant are colocated on a single node, message complexity reduces to $3n-3$ or $4n-4$ in the failure-free case.

If f participants time-out during their uncertainty period and execute cooperative recovery within their termination protocol, fn requests are sent. At most $n-f+1$ certain participants (including the coordinator if the communication path recovered) will answer the request. As a result of an answer to a request, blocked participants move out of uncertainty and will also answer requests of uncertain participants resulting in a total number of $2nf - \frac{f^2}{2} + \frac{n}{2}$ messages as shown in [13]. However, cooperative recovery is not guaranteed to be successful within the first request round and is repeated until successful. Hence, the previous estimation is only a very rough approximation for the message complexity of cooperative recovery and can be interpreted as a lower bound.

2PC is susceptible to blocking because it only solves the AC and WAC problems. Hence, to evaluate the applicability of 2PC in a MANET, the likelihood for blocking should be examined, which depends on the size ΔU of the uncertainty window.

A participant enters its window of uncertainty after issuing its vote message. Considering only message delays, ΔU is of size $2\delta_m$ if no communication failures occur, i.e. one message delay δ_m is required for vote messages and one for the commit message. Hence, ΔU_{min} has size $2\delta_m$. If a failure occurs, the coordinator misses a vote and awaits a time-out Δ_{vo}, the uncertainty window of a correct participant is then extended to $\Delta U_{max} = 2\delta_m + \Delta_{vo}$.

The main advantage of 2PC is its small message complexity of $4n$; in fact, 2PC shows the smallest possible message complexity to solve AC and WAC. Additionally, the small size of ΔU compared to other protocols is an argument in favor of 2PC.

Variants of the 2PC protocol proposed in the literature reduce transaction delay by omitting log operations rather than changing the message scheme. The most important proposals here are the Presumed-Abort and Presumed-Commit protocol [90, 7, 101]. The Presumed-Abort protocol reduces required log writes by omitting a forced log write at the coordinator in cases where the transaction was aborted. If on restart the coordinator does not find a log on a transaction, it presumes that this transaction was aborted. The same idea is behind the presumed commit protocol, only that it is presumed that a transaction was committed if no log is found. However, such optimizations aim at increasing throughput in fixed environments, where forced log writes are expensive operations. This is not the case in MANETs.

3.4.2 Protocols solving DAC

The DAC problem assumes that the commit decision is derived from the execution phase and no explicit voting is required. This allows the voting phase of the 2PC protocol to be dropped. The most recognized protocol solving DAC is the one-phase commit protocol (1PC) proposed in [55]. The small message complexity of 1PC makes it especially interesting for infrastructure-based mobile environments as proposed in [5, 4]. In the following, the basic structure of the 1PC protocol is discussed.

3.4.2.1 One-Phase Commit Protocol (1PC)

In the 1PC protocol every operation of a local transaction branch executed by a participant is acknowledged to the coordinator. By sending an acknowledgment, the participant promises that the ACID properties are guaranteed for all operations executed so far and an immediate commit of the complete transaction branch is possible. A positive acknowledgment inherently posses the semantic of a commit vote, while a negative acknowledgment has the same consequences as an abort vote. The actual commit protocol consist only of a single message round, where the coordinator sends a *commit* or *abort* message to all participants and collects acknowledgments. The coordinator decides on global commit if positive acknowledgments from all participants for all operations are received. This is safe because the acknowledgments ensure that all participants receiving the commit decision can commit.

1PC is a blocking protocol and therefore solves DAC and WDAC. By issuing an acknowledgment message a participant transfers control of its transaction branch to the coordinator, i.e. it moves into prepared state. In fact, the only time a participant can unilaterally abort the transaction is after it has received an operation from the coordinator and before issuing the acknowledgment for this operation. During this period, 1PC is susceptible to the same blocking situation as 2PC.

3.4.2.2 Evaluation of 1PC

The low message complexity of $2n$ of the commit phase is the major advantage of 1PC for a MANET scenario. At first glance, the main drawback of the 1PC protocol is its tight integration of the execution and commit process, which requires that the commit coordinator receives acknowledgments for all operations. However, a closer look at the main standards DTP of X/Open and OTS by OMG reveals that the most important transaction standards assume a similar behavior with 2PC, with the difference that operation acknowledgments are sent to the execution coordinator and not to the commit coordinator.

In the case of failures, a cooperative recovery scheme similar to 2PC can be used to compensate for blocking of participants. The message complexity of cooperative recovery is the same as in 2PC. Similar to 2PC, 1PC is susceptible to blocking caused by a coordinator (blocking (ii)) and to blocking caused by a participant failure (blocking (i)). However, the size of ΔU is a severe drawback of the protocol as shown in the following.

Assume that $\delta_{op_{i,i+1}}$ is the size of the uncertainty period between operation i and $i+1$ of a participant called PA, m the total number of operations issued to PA, t_o the time PA acknowledges its last operation, and t_l the time the last participant acknowledges its last operation. Hence, at $t_l + \delta_m$ the global decision is derived by the coordinator. The size of ΔU of PA in the failure-free case is given by the sum of all intermediate uncertainty periodes and calculates as $\Delta U = \sum_{i=1}^{m}[\delta_{op_{i,i+1}}] + t_l + \delta_m - t_o$. During this time, PA is susceptible to blocking (i) and blocking (ii). Hence, the size of ΔU is different for every participant and depends on the number and distribution of operations assigned to PA. However, ΔU is considerably larger than in 2PC, which is the major disadvantage of this protocol concerning its applicability in MANETs. Due to the intermingling of processing and commit phase, a failure during the processing phase causing transaction abort reduces the size of ΔU only if the last participant fails with its last operation, then ΔU is increased by a time-out Δ_{vo}.

The different variants and optimizations of 1PC proposed aim at alleviating the disadvantage of extensive logging by proposing strategies reducing required log writes. For example, the Coordinator Log [138] and Implicit Yes-Vote protocol [6] transfer the responsibility for ensuring commit-resiliency from participants to the coordinator.

3.4.3 Protocols solving NB-WAC

The most prominent algorithms solving NB-WAC are the Paxos Commit protocol (PC) [56] and the Quorum Three-Phase Commit protocol (Q3PC) [13, 136].

At the heart of Paxos Commit lies the Paxos Consensus algorithm proposed in [91, 119, 88]. Paxos Consensus solves the consensus problem in partially synchronous systems if at least a majority of so called *acceptor* processes are available. For a detailed description of Paxos Consensus, see [89, 34].

The PC protocol uses multiple instances of Paxos Consensus to solve NB-WAC and reaches a decision if a majority of participants are available. Recall that this is exactly the behavior predicted by the discussion on solvability of NB-WAC in the previous section. PC does not provide special termination or restart protocols in case of failures like 2PC or 3PC do; in fact, it is always the same protocol that can be safely started and restarted multiple times in parallel to derive a decision. Ballot numbers chosen for every execution of the algorithm are used to order and separate multiple PC instances. The PC protocol is described in more detail later, while for Q3PC only a brief description is given in the following.

Q3PC is based on the Three-Phase Commit protocol (3PC) proposed in [135] that enhances the 2PC protocol with an additional protocol phase allowing a new coordinator to be elected that can can terminate the transaction on behalf of the failed coordinator. However, 3PC allows for non-blocking termination only under the single failure assumption [135], which is not given if communication failures can occur. To allow at least a majority of correct participants to terminate in presence of node and communication failures, a quorum-based decision on the global decision is proposed by Q3PC [13, 160]. However, deriving a quorum requires safe election of a new coordinator, and the newly elected coordinator has to assure that a majority of participants know its intention, i.e. a majority of participants must be reachable before it can decide. The main problem of Q3PC is that it is not resilient to faulty leader election. Reaching agreement on a new coordinator requires to solve the leader election problem (a problem as hard as consensus) in presence of node and communication failures. This process is disregarded in descriptions of Q3PC. However, it is treated separately in distributed system research, and solutions to this problem are assumed to be available elsewhere. It is the main advantage of PC over 3PC that PC is resilient to faulty leader election and the algorithm is specified completely. Multiple instances of PC can be safely processed in parallel with two nodes consider themselves as coordinator. Such a situation would lead to inconsistency in Q3PC. In the following, I will briefly describe the PC protocol, while for the sake of clarity, Q3PC is not regarded further.

3.4.3.1 Paxos Commit (PC)

Within an instance of the Paxos Consensus algorithm, proposed values of participants are collected and one proposed value is chosen and remembered forever, i.e. consensus among participants is reached on this value.

The basic idea of Paxos Commit is to derive consensus on each participants' state (prepared or abort), while the coordinator simply combines the consensus values, i.e. if every participant proposed prepare and prepare was the chosen value for every Paxos instance, the global decision is commit. Therefore, the PC protocol involves the three roles of Paxos Consensus: *acceptor*, *proposer*, and *leader*. The leader role is initially taken by the transaction coordinator, while transaction participants act in the role of proposers and as acceptors. However, there can be more acceptors integrated than participants are involved in the transaction. The basic idea is that every participant tries to get his vote (prepared or abort) accepted within an instance of the Paxos Consensus protocol by a majority of acceptors. Hence, there is one instance of the Paxos Consensus protocol started for each transaction participant.

The protocol is initiated by the leader sending a prepare message containing ballot number 0 to all participants. The coordinator is stateless and simply learns about the outcome of PC later. A participant receiving this message tries to get its initial value (prepared or abort) accepted by sending a message containing its vote and the ballot number received with the according prepare message to every acceptor. Every acceptor maintains a vector with the votes of participants. These vectors are sent to the leader and are evaluated according to the following rule: if for every participant involved a majority of acceptors received prepared as proposal, the global decision is commit. If the leader learns that one instance with ballot 0 has proposed abort, the global decision is abort and the leader can shortcut the protocol by broadcasting an abort message to all participants.

If no decision with some participants is derived in the first round, the leader can issue a new prepare message with an increased ballot number starting a new instance of Paxos Commit. For every ballot number different from 0, an additional phase is prepended to the protocol. The new ballot number is sent in a propose message to all acceptors. If a majority of acceptors reply and indicate that they have not seen a higher sequence number, then the leader is the current leader and acceptors will reject messages with lower ballot numbers. The new leader then issues a new prepare message with the new ballot number.

3.4.3.2 Evaluation of PC

Paxos Commit is resilient against communication and site failures and not susceptible to blocking if a majority of acceptors remain non-faulty. In the failure-free case, five message rounds are required to derive a decision. As an instance of Paxos Consensus is executed for every participant, PC shows a high message-complexity of $n(2n + 3) - 1$, if only participants act as acceptors. The high message complexity is a severe drawback of the protocol and its application in MANETs. For an exact analysis of message complexity, see [56]. If only the coordinator node is used as acceptor, PC reduces to 2PC.

Although participants in PC do not enter an uncertainty window as in 2PC and 1PC where a failure causes execution of a termination or restart protocol that is susceptible to blocking, failures while participants are in prepared state cause initiation of a new instance of PC. This period has a similar semantic as the uncertainty window ΔU defined above and is given by: (i) the message delay of the propose message of a participant; (ii) the message delay for the accepted votes sent to the coordinator by acceptors; and (iii) the message delay to learn the final decision derived by the coordinator. Hence, in the failure-free case the period where a failure causes restart of PC is given by $\Delta U_{min} = 3\delta_m$, while if there are failures this period can extend to $\Delta U_{max} = 3(\delta_m + \delta_{to})$. If the time acceptors and the coordinator wait for proposals and accepted votes is similar to the time a coordinator would wait for a vote in 2PC, i.e. $\Delta_{vo} = \delta_{to}$, then ΔU_{max} is three times larger in PC than in 2PC.

In cases of network partitioning, isolated participants remain blocked in the sense that they cannot get a quorum as leader of PC and will restart an instance of PC over and over again until a participant can be reached that knows the global decision or a majority of acceptors is finally reached. However, in a partition containing a majority of participants, NB-WAC is solved. Hence, PC is susceptible to the same blocking situations as 2PC, i.e. blocking (i) and blocking (ii), while these are limited to participants residing in partitions with a minority of participants.

If transactions with only two participants are processed while the coordinator is colocated with one participant, additional nodes to act as acceptors have to be defined. This is required because a communication failure between the coordinator and the participant will immediately cause a situation where neither the coordinator nor the other participant can reach a majority of acceptors and both will remain blocked. Hence, at least three acceptors are required to allow for a majority quorum.

3.4.4 Protocols solving SAC

In semantic atomic commit, participants are allowed to terminate their local transaction branches unilaterally without waiting for the other participants to decide. However, the global commit decision can be derived first, when all participants have completed their branches. This induces a new class of blocking situations that I denote as *extended uncertainty* and that are defined later in this subsection.

Protocols proposed so far differ in when participants are notified that their local transaction branch is finished and no new operations will be sent for a transaction. The most cited protocol addressing SAC is the Optimistic-2PC (O2PC) protocol [93], which is briefly described in the following, while more important to this work is a variant that I call Early Commit (EC) proposed in ATMs like Sagas [50] or Multi-Level transactions [152].

O2PC is a slightly modified version of the 2PC protocol using the same message flow as 2PC. The only difference is that in O2PC, participants do not move into a prepare state but directly commit their local branch after sending a commit vote. All participants commit their transaction branch at the same time on receipt of the prepare message. Thus, participants must wait until all other participants have finished their transaction branches before local commit is possible. If the local commit decision opposes the global decision, a compensating transaction must be executed to semantically undo the effects of the committed transaction branch as described in Section 3.2.2. The message and time complexity of O2PC is similar to 2PC as the protocols show the same message flow.

3.4.4.1 Early Commit (EC)

An alternative scheme allowing for fast local commit is to let participants know immediately when their transaction branch is finished and no further operations will follow. In this scheme, a participant can terminate its local transaction branch as soon as it has executed the last operation of its branch at time t_o, without being required to wait until all other participants have finished their branches. Hence, similar to 1PC the commit phase and the execution phase in EC are not separated as in 2PC, but they overlap. The EC approach is proposed with most advanced transaction models.

The general semantic transaction model used in the remainder of this work to investigate semantic atomicity in MANETs will follow the EC approach.

3.4.4.2 Evaluation of EC

While with O2PC the uncertainty window of participants is similar as in 2PC, the EC approach induces larger uncertainty windows, as a participant moves into uncertainty with the acknowledgment of its last operation. While it is beneficial for participants to terminate their transaction branches early, the increased size of uncertainty periods might have a negative effect. Assume that t_o is the time a participant acknowledges its last operation, then its uncertainty window begins at t_o, because from this point in time it cannot influence the global decision any more. If the last participant acknowledges its last operation at time t_l, then the coordinator derives the global decision at time $t_l + \delta_m$. The uncertainty window ΔU of the participant is of size $[t_l + 2\delta_m - t_o]$ in the failure-free case, while the uncertainty window of the last participant is of size $2\delta_m$. Hence, in the EC scheme all participants have individual uncertainty windows. In the presence of failures the uncertainty window of participants varies, as the coordinator may decide on abort before t_l and after t_o. Hence, ΔU of a correct participant is reduced to $t_f - t_o$ where t_f is the time, when the coordinator detects a failure that leads to abort.

Similar to 1PC, the commit process itself requires only one message round with message complexity $2n$. The suggested approach to compensate for extended uncertainty situations that are defined in the following is cooperative recovery showing a message complexity of $[2nf - \frac{f^2}{2} + \frac{n}{2}]$, while f is the number of failed participants.

3.4.4.3 Blocking Situations in Early Commit

Blocking, in the sense that a participant is forced to remain in prepared state, cannot occur in semantic atomicity since no prepared state exists. However, for reasons already described in Section 3.2.2, it is undesirable for participants to remain uncertain about the global decision indefinitely. The same failure situations that cause blocking (i) and (ii) in 2PC, cause a situation where a participant is uncertain about the global decision for an indefinite period in the

EC protocol. I call these situations *extended uncertainty* and distinguish two cases:

Extended uncertainty caused by a participant failure Describes the situation where a node failure of a participant PA or a communication failure with the coordinator force PA to remain uncertain about the global decision, although this decision is available in $\mathcal{A}$. A communication failure with the coordinator leads to such a situation if it occurs after PA has acknowledged its last operation at t_o and the global decision cannot be delivered to PA due to a communication failure. Similarly, a node failure of PA causes extended uncertainty if the coordinator is unreachable at reconnection of PA to $\mathcal{A}$. In the remainder of this work, I will call this situation **extended uncertainty (i)**.

Extended uncertainty caused by node failure of the coordinator Defines the situation situation where a node failure of the coordinator leads to extended uncertainty of participant PA if the node failure happens after PA has acknowledged its last operation at t_o and before the global decision is derived by the coordinator. In this situation, PA will not receive the global decision and cannot decide independently on the global decision. In the remainder of my thesis, I will call this situation **extended uncertainty (ii)**.

A main difference of the extended uncertainty situation defined above to blocking is that extended uncertainty (i) and (ii) can already occur during the processing phase, while blocking (i) and (ii) can only happen during the commit phase, causing uncertainty windows to be wider than in the strict case. It will be a major contribution of Chapter 4 to predict the risk of extended uncertainty induced by the EC scheme.

3.4.5 Summary - Atomic Commit Protocols

In this section, I summarized the results of over two decades of research on ACPs, ranging from 2PC proposed by Jim Gray in 1978 to Paxos Commit published by Lamport and Gray in 2003. The protocols presented are theoretically well understood and proven to be correct and therefore present the state-of-the-art in commit protocols for general system models like the asynchronous model and models assuming partial synchrony. Table 3.1 summarizes the important characteristics of the presented protocols that influence their applicability in a MANET.

For applicability in a MANET, commit protocols should show a small uncertainty window to keep the probability for blocking in strict atomicity and extended uncertainty situations in semantic atomicity small, because these situations generally cause the initiation of expensive recovery schemes such as cooperative recovery, election of a new leader, or restart of a PC instance. Additionally, message complexity should be low, as message transfer in MANETs is expensive in terms of required energy and is susceptible to communication failures.

Of all schemes enforcing strict atomicity, the basic 2PC protocol has the smallest uncertainty window. It is therefore the candidate to be considered first

Commit Protocol	Problem solved	Message Complexity *(no failures)*	Message Complexity *(f failures)*	Uncertainty Window ΔU
Two-Pase Commit	AC/WAC	$4n$	$2nf - \frac{f^2}{2} + \frac{n}{2}$	$\Delta U_{min} = 2\delta_m$ $\Delta U_{max} = 2\delta_m + \Delta_{vo}$
One-Phase Commit	DAC/WDAC	$2n$	$2n_pf - \frac{f^2}{2} + \frac{n_p}{2}$	$\sum_{i=1}^{m}[\delta_{op_{i,i+1}}] + t_l - t_o$
Paxos Commit	NB-WAC♣	$[n(2a+3)-1]^{\star}$	$[2a + n(2a+3) - 1]^{\star}$	$\Delta U_{max} = 3(\delta_m + \delta_{to})$ $\Delta U_{min} = 3\delta_m$
Quorum Three-Phase Commit ♦	NB-WAC♠	$5n$	$[(n-f)^2 + 2(n-f)]$	$\Delta U_{max} = 4\delta_m + \delta_{to}$ $\Delta U_{min} = 4\delta_m$
Optimistic Two-Phase Commit	SAC/WSAC	$4n$	$2nf - \frac{f^2}{2} + \frac{n}{2}$	$\Delta U_{min} = 2\delta_m$ $\Delta U_{max} = 2\delta_m + \Delta_{vo}$
Early Commit	SAC/WSAC	$2n$	$2nf - \frac{f^2}{2} + \frac{n}{2}$	$t_l + 2\delta_m - t_o$

* a is the number of acceptors in each instance of Paxos Consensus.
♣ NB-WAC is solved if a majority of acceptors can reach each other.
♦ For the derivation of message complexities for O3PC, see [13].
♠NB-WAC is solved if a majority of participants can reach each other.

Table 3.1: Comparison of atomic commit protocols.

for a MANET environment. However, 2PC is susceptible to blocking as it solves AC and WAC only. Hence, the question to be answered to judge the applicability of 2PC for MANETs is how often a blocking situation is actually experienced, and how well cooperative recovery can compensate for these situations. Only if the resulting blocking rate is unacceptable, more reliable protocols solving NB-WAC, such as PC or Q3PC, must be considered.

The price to pay by the non-blocking protocols PC and Q3PC is a higher message complexity and larger uncertainty windows than in 2PC. While the non-blocking behavior in a partition with a majority of participants is desirable in MANETs, the latter is not. The question to be answered here is how much better the compensation for blocking in a MANET environment is compared to 2PC with cooperative recovery. Note that with increased size of uncertainty windows and a higher message complexity, also the probability of failures and therefore for blocking increases, because more messages are transferred that can be lost and larger uncertainty windows increase the risk of failures to occur within these windows. Hence, only if the negative effect of larger uncertainty windows does not overcompensate the non-blocking behavior in a large partition, application of PC or Q3PC is feasible. Hence, whether a protocol is beneficial depends on the distribution of node and communication failures in a certain MANET scenario.

Of the protocols solving semantic atomicity, O2PC has a slightly higher message complexity of $4n$, but a much smaller uncertainty window compared to the EC scheme. O2PC shows basically the same characteristics as 2PC, except that it is not susceptible to blocking but to extended uncertainty. This is not astonishing since it has exactly the same structure and message flow as 2PC. Hence, results of the blocking rate derived for 2PC are portable to extended uncertainty that would occur with O2PC. I will therefore not consider the O2PC protocol directly any further since 2PC will be examined in detail in the following chapters.

In contrast, the EC scheme shows a unique structure and message flow not found in any other protocol. The size of uncertainty windows in the EC scheme depends heavily on the temporal distribution of operations to participants, but can generally be assumed to be larger than in 2PC schemes. An exception is the case where only one remote participant exists; then the size of the uncertainty window reduces to $2\delta_m$. The message complexity of the EC scheme is rather low at $2n$, making it a promising candidate for MANETs. I will examine the susceptibility for extended uncertainty of the EC scheme in more detail in the following chapters, as this scheme is mostly used in ATMs and is also less susceptible to abort than 2PC, as I will show in Chapter 4.

The protocols presented above solve the atomic commit problems with a minimum of assumptions about the system model, i.e. they consider an asynchronous or partially synchronous system model. I consider the development of fundamentally new protocols solving these problems in the asynchronous and partially synchronous system models to be not feasible since the minimal protocols for solving atomic commit in these models are found yet.

The development of new commit protocols is more application-driven. Special characteristics of the environment that the transactional system is deployed to can be exploited to design more reliable commit protocols. Hence, new protocols can be found for more special system models rather than the basic asynchronous or partially synchronous models. Therefore, the basic question in the context of this work is whether a MANET shows some special characteristics that can be exploited to increase reliability of commit protocols, i.e. to design novel protocols.

If minimum assumptions are made for a MANET environment as in the system model of this work described in Section 2.3, the system model equals the asynchronous system model and the basic protocols presented in this section are indicated. In cases where further assumptions can be made, such as stable nodes that are assumed not to disconnect from $\mathcal{A}$ or reliable communication channels with at least some nodes, enhancements of the basic protocols presented here are imaginable. This general pattern can be observed in the area of cell-based wireless environments. Here, several commit protocols have been proposed that are tailored to the more special system model of cell-based wireless networks and make use of special characteristics such as mobile-support stations, that are assumed to be stable and always reachable within a cell. For the sake of completeness, I present some of these protocols in the following section since they present the latest developments in the area of atomic commit protocols.

3.5 Commit Protocols for Mobile Environments

Several commit protocols that are optimized for mobile environments have recently been proposed. These protocols are mostly not designed for MANET environments but for cell-based mobile networks where a distributed transaction is processed between mobile hosts and fixed servers that reside in a fixed network. I call this environment *infrastructure-based*.

To the best of my knowledge, there has only been one protocol proposed so far in [29] that is especially developed for MANETs.

3.5.1 Infrastructure-Based Mobile Environments

In an infrastructure-based environment, so-called *mobile-support stations* (also called base-stations) connected through a fixed network serve mobile nodes roaming in a defined area (cell). Within the back-end, stationary servers are placed. While fixed servers and mobile-support stations and their communication network is assumed to be stable, mobile nodes are assumed to frequently disconnect from their mobile-support station.

The most-cited transaction models proposed in this environment are Reporting Transactions [38], Kangaroo Transactions [45], Pro-Motion Compacts [151], and Toggle Transactions [43]. For an overview of transaction concepts for the mobile-host fixed-server environment, see [133]. These models propose advanced schemes addressing the migration of mobile hosts between cells during transactions processing and mostly assume semantic atomicity.

However, there exist some commit protocols specially developed for the infrastructure-based environment, such as the Transaction Commit on Timeout protocol (TCOT) [83], the Unilateral Commit for Mobile (UMC) [20] protocol, and the Mobile 2PC (M2PC) [106] protocol. The basic building block for optimization here is the assumption that the fixed-wired back-end servers and mobile-support stations are reliable. Mobile support-stations and back-end server are used as log entities to persist the global transaction decision. At reconnection time, an uncertain mobile host is then guaranteed to leave blocking. In the following, TCOT, UMC, and M2PC are described briefly.

3.5.1.1 Transaction Commit on Timeout (TCOT)

The Transaction Commit on Timeout (TCOT) protocol [83] uses time-outs to allow for unilateral commit and compensating transactions to undo opposing local decisions. Hence, TCOT solves semantic atomicity. The basic idea of TCOT is that participants assume commit as the global decision if the transaction is not aborted within a certain time-out period. The transaction model assumed by TCOT differs from the standard X/Open model by presuming that every transaction participant knows all operations of its branch in advance. In TCOT a transaction branch is called *execution-fragment* and the global transaction is decomposed into such fragments that are shipped to participants at transaction start. TCOT assumes exactly one mobile node among transaction participants that holds a data replica originated from a fixed server. All participants calculate an *execution time-out* and a *shipping time-out* at transaction start, that is sent to the coordinator before they begin execution of their fragments. The execution time-out is an upper bound for the time a node requires to execute its fragment, while the shipping time-out describes an upper bound required by the mobile host to transfer updates to the fixed replica master server in the back-end.

The coordinator decides on abort if an abort message from a participant is received or if a participant does not send a commit message until its execution time-out is exceeded. Global commit is decided if the coordinator receives commit messages from all participants within their execution time-outs. The main advantage of the protocol is that only a global abort decision is propagated to participants, while a global commit decision is not. Hence, the message-complexity is reduced. A participant can unilaterally decide on commit if it commits within its execution time-out. Furthermore, participants may complete the transaction independently based on time-out values.

The main drawback of TCOT is the presumption that accurate time-outs for execution and message transfer can be derived. Note that this poses strong timing guarantees not given by the basic asynchronous and partially synchronous system models.

3.5.1.2 Unilateral Commit for Mobile (UCM)

The Unilateral Commit for Mobile (UCM) protocol [20] is basically a 1PC protocol providing strict atomicity, i.e. the DAC problem is solved. In contrast to TCOT, multiple mobile clients can participate in a transaction. The main failure cases considered are transient failures of mobile clients and disconnection of clients from their mobile-support station. Coordinator failures are not considered.

UCM mainly proposes an architecture with the functional entities *Participant*, *PAgent*, *LogAgent*, and *Coordinator*. Operations sent to a mobile host are continually logged on a fixed server to allow later retransmission if the global decision was commit and a mobile host suffered a failure before being able to commit its local transaction branch. To preserve the commit information, the PAgent is located at a mobile-support station. To ensure that a recovering mobile participant can learn if it must redo a branch which was lost due to a failure, the PAgent executes a small transaction on the mobile host that force-writes the global commit decision. First, if this operation is acknowledged successfully by the mobile participant, the PAgent forwards the final commit decision. On receipt of the final commit message, the mobile host commits its local transaction branch and makes it durable. Hence, UCM assumes that the work of transactions in prepared state is completely lost when the mobile host fails. In cases where recovering mobile hosts contact the PAgent and the PAgent discovers that an uncommitted transaction branch was lost and has to be re-executed, the LogAgent is contacted and the according operations are transferred to the mobile host. This concept is also known as logical logging.

In fact, UCM does not pose any fundamentally new solutions to solve DAC. The main contribution of this protocol is to propose an architecture that integrates the mobile-support stations into 1PC processing and describes how to distribute transaction logs efficiently among fixed servers, mobile-support stations, and mobile hosts.

3.5.1.3 Mobile-2PC (M-2PC)

The Mobile-2PC (M-2PC) protocol [106] revisits the standard 2PC protocol in the mobile-client fixed-server environment. M-2PC does not consider coordinator failures or permanent failures of mobile hosts.

The main idea is to distinguish between commit among fixed participants and with mobile participants. Commit among participants residing in the fixed back-end is done using the standard 2PC protocol, while mobile participants can delegate their commit duties to the coordinator. The idea here is that a mobile participant moves directly into prepared state after successfully executing its branch by sending a commit vote to the coordinator without awaiting a prepare message. Afterwards, the mobile participant can disconnect from the support station, while the coordinator commits the transaction among the fixed participants considering the mobile host's vote. On reconnection, the mobile host can learn about the global decision.

The advantage of this scheme is that a disconnection of the mobile host after it has finished its transaction branch does not delay other participants or aborts the complete transaction. Note that this is basically the same idea underlying the EC scheme described in Section 3.4.4.1. Hence, SAC is solved for mobile transaction participants and AC is solved for participants of the fixed network.

3.5.2 Infrastructure-less Mobile Environments

As MANETs are infrastructure-less by definition, commit protocols assuming reliable parts such as mobile-support stations are not applicable. One of the few commit protocol tailored specially to MANETs is the integrated commit protocol for mobile network databases proposed by Böse et. al. [29] and described in the following.

3.5.2.1 An Integrated Commit Protocol for Mobile Networks

The integrated commit protocol for mobile network databases uses multiple coordinators to increase reliability of commit processing. Blocking risks are reduced because participants that can contact multiple coordinators in case of failures to learn about the global decision and the failure of a coordinator can be compensated by another coordinator that derive the global decision on behalf of the failed coordinator.

While the use of multiple coordinators decreases the risk that a participant cannot reach any coordinator, a new problem has to be solved, namely agreement among coordinators on the global commit decision must be established, i.e. NB-WAC has to be solved among the coordinator group. Blocking risks within the group of coordinators is reduced by assuming that all coordinator nodes will remain in close vicinity, e.g. 1–2 hop distance, to each other for the time of transaction processing. To reduce the blocking risk of coordinators further, 3PC in combination with Paxos Consensus is used to solve NB-WAC for at least a majority of coordinators. These considerations resulted in the system architecture depicted in Figure 3.2.

Each participant is assigned to one node of the coordinator group and processes a 2PC protocol with this node. One node of the coordinator group acts as main coordinator within the 3PC protocol processed by the coordinator group. Since communication among coordinators is transparent for participants, from the outside perspective the coordinator group acts like a single commit coordinator, i.e. the group presents a consistent state of the global decision.

In the failure-free case coordinators collect votes derived within 2PC from participants and forward the bundled votes to the main coordinator. This presents the first phase of the 3PC protocol processed within the coordinator cloud. If all participants voted for commit, the main coordinator issues a *prepare_to_commit* message that is acknowledged by all coordinators. The final commit message of the main coordinator is then forwarded to the participants of each coordinator.

In the case of a blocking situation caused by a participant failure, i.e. blocking (i) as defined in Section 3.4.1.2, a participant that cannot reach its assigned

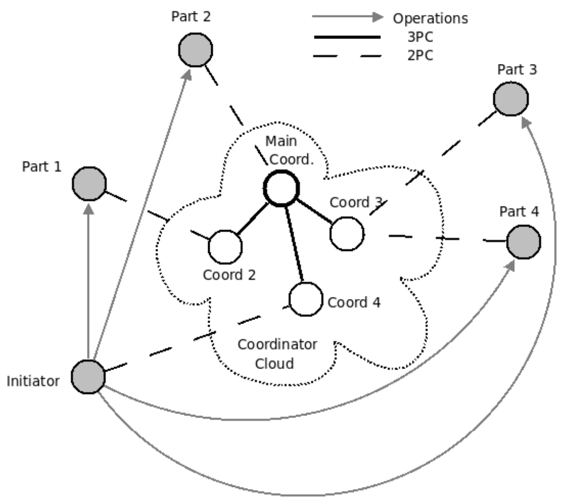


Figure 3.2: Integrated Commit Protocol for Mobile Network Databases.

coordinator can contact another node of the coordinator group to learn about the global decision. This implies that all nodes of the coordinator group must be known to all participants before the transaction is started.

Blocking situations caused by a node failure of the coordinator, i.e. blocking (ii), is prevented by allowing another coordinator to terminate the transaction. However, this requires to solve safe leader election and blocking among coordinators. Blocking within the coordinator group is avoided for at least a majority of coordinators by the use of 3PC combined with a quorum approach. To preclude faulty election of a new coordinator, ballot numbers as known from Paxos Consensus are used. In the case of network partitioning, non-blocking behavior can only be reached for a majority of coordinators that can reach each other. The probability of this blocking situation is minimized by assuming that coordinators are in close distance to each other and that therefore partitioning is unlikely. For a more detailed description of failure handling, see [29].

In the failure-free case, the protocol needs $3n + 4c + 1$ messages to reach a decision, where c denotes the number of coordinators. While $3n + 1$ messages are required by the 2PC protocol, $4c$ messages are the cost of 3PC within the coordinator group. The message costs of 3PC are less expensive since coordinators are assumed to reside in direct vicinity and hence no relaying of messages is required, i.e. a single broadcast message of the main coordinator is received by the complete coordination cloud. Therefore, a major advantage of the protocol is that the message load in the failure-free case is only slightly increased compared to 2PC, while the protocol is considerably more reliable since NB-WAC is solved if a majority of coordinators is available.

A drawback of the approach is its complexity in the failure case, which anticipates application of this protocol if there are frequent failures. However, message complexity is then not higher than in Paxos Commit or Q3PC. The main disadvantage of the protocol is the strong assumption that a coordinator cloud can be found. To make sense, the protocol needs at least a group of three coordinators, otherwise more efficient backup commit protocols like [82, 124] can be used, which I will present in more detail in Chapter 6. However, discovering at least three stable nodes in single or 1–2 hop distance may be a difficult task in a highly dynamic MANET. Furthermore, in a dense MANET with low node mobility where such a group is likely to be given, multi-hop routing schemes will most likely be able to provide long path durations; hence, blocking due to partitioning or communication failures is a rare event and even more lightweight protocols such as 2PC are possibly sufficient. It is one objective of this work to answer this question.

3.5.3 Summary - ACPs for Mobile Environments

In this section, I have presented some commit protocols designed for wireless environments. The common rationale behind all protocols presented was to exploit special characteristics of the system environment such as mobile-support stations or a reliable group of coordinators that is assumed not to partition to allow for more efficient recovery schemes to compensate blocking and extended uncertainty situations. Note that increasing the efficiency of recovery or reducing the probability that a protocol leads to blocking are the natural options to approach the blocking problem, since completely anticipating the problem is impossible, as shown in Section 3.3.

The common idea behind the protocols presented above is to use more reliable entities to preserve transaction logs or to coordinate commit processing of a transaction. Hence, as blocking is at its core a matter of reliability and availability of commit information, the basic rationale all optimizations of commit protocols in wireless environments must follow is to increase the availability of this information. In fact, the SLS architecture presented in Chapter 5 follows exactly this idea.

Another observation of this section is that except for the integrated commit protocol of Section 3.5.2.1, no commit protocols specially tailored to MANETs exist. The protocols proposed for infrastructure-based environments are obviously not applicable in a MANET.

Design of new commit protocols for MANETs requires characteristics of the system model to be identified, that can be exploited to increase the availability of commit information. For example, the integrated commit protocol assumes a group of nodes with temporarily reliable communication. While the infrastructure-based environment has some obvious properties such as reliable fixed mobile-support stations and servers that can be used for this purpose, the MANET system model of this work is more closely related to the basic partially synchronous system model that assumes no such properties. Hence, possible optimizations in a MANET can only be found if the current network topology and therefore reliability of communication and availability of nodes for a certain time are considered in one way or another. However, this must happen in an ad-hoc manner, i.e. adaptive approaches considering the current situation and adapting commit processing accordingly are required.

Protocols developed for MANETs show an optimization in theory, while a quantitative statement about the blocking reduction achieved is at most given by simulation for some example MANET scenarios. Estimation of the benefit for other MANET environments is impossible without new simulation studies. Generally it is not clear whether optimizations are actually required or if lightweight protocols such as 2PC show a blocking rate in MANETs one can live with.

In the following, I will define the transaction models that are used to examine the abort and blocking risks of the basic commit protocols described in Section 3.4 in MANETs.

3.6 Transaction Models of this Work

Investigation of abort and blocking probability within the remainder of this thesis will be based on two transaction models defined in this section. One represents traditional transactions as described in Section 3.2.1 and enforces strict atomicity using 2PC, the other transaction model shall represent advanced transactions as introduced in Section 3.2.2 and provides semantic atomicity using the EC scheme. While in the beginning of this chapter, these models were described in a rather informal way, a more formal description is given here. Both models are based on a general model of distributed transactions described first and differ in commit processing only.

3.6.1 General Distributed Transaction Model

The basic transaction model I consider is the flat ACID transaction model. Following the X/Open DTP model, a transaction consists of a set of operations that are issued by an *application*. All operations received by a *participant* constitute a local transaction branch of the global transaction. To avoid the need for initially choosing a *coordinator*, I assume that the application process and the transaction coordinator are colocated.

Each execution of an operation is acknowledged by the participant. These acknowledgments are used to detect failures of participants, while a differentiation between a node failure or a communication failure is not possible. If an acknowledgment for an operation is not received during timeout Δ_{op}, the application requests the coordinator to globally abort the transaction. The coordinator will then issue abort messages to all participants. Advanced transaction models can be reflected here by tolerating detected failures as described in Section 3.2.2. However, rich nested structures of subtransactions as proposed in some advanced transaction models are not assumed.

Generally, I distinguish between the *processing phase* and the *decision phase* of a distributed transaction. The processing phase begins at time t_s when the application initiates the transaction and ends at time t_p when the acknowledgment of the last operation of the global transaction is received by the application. The participant executing the last operation of the global transaction is denoted as PA_{last}.

I assume that a participant i receives the last operation of its transaction branch at some random time t_o. For each participant, the random variable t_o is distributed within the interval $[t_s, t_p]$ according to a pdf $o(t_o)$. The pdf $o(t_o)$ depends on application semantics, i.e. the role of participant i in the transaction. For the sake of simplicity, I assume the same pdf for all participants of a transaction. The distribution of operations in $[t_s, t_o]$ for a participant is not considered. The basic idea of the model is that a failure in the interval $[t_s, t_o]$ is detected at the latest at time t_o, independently of the distribution of operations

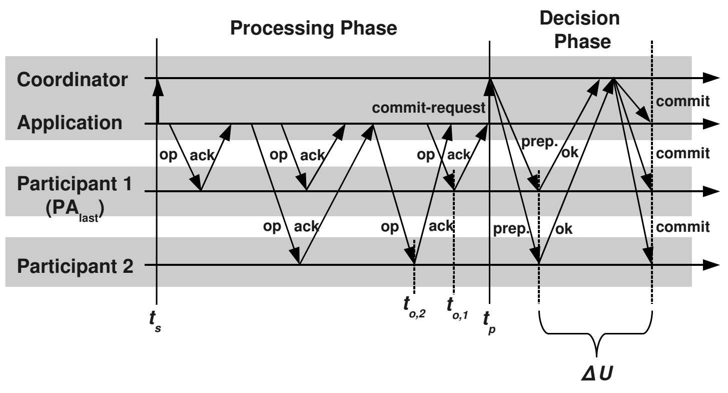


Figure 3.3: Strict transaction model with 2PC.

within this interval, while a failure in $[t_o, t_p]$ will first be detected in the commit phase.

If no failure is detected during $[t_s, t_p]$, the decision phase is initiated by starting an ACP at time t_p in the strict model and at time t'_p in the semantic transaction model.

As already mentioned, the application and the coordinator are assumed to be colocated on the same node, allowing message delays of message exchange between the coordinator and the application to be neglected. Execution delays during the processing phase are also neglected. Based on this general model, I differentiate between the *strict* and *semantic transaction model*.

3.6.2 Strict Transaction Model

The strict model shall represent the traditional transaction model described in Section 3.2.1. In the strict model, termination of a local transaction branch is conditionally bound to the termination of the other branches of the same global transaction. A local transaction branch that already completed all its operations is not allowed to commit until all other remote branches are known to terminate successfully. To resolve these termination dependencies, a local transaction manager must be able to announce that it is *prepared* to commit its local branch. In the strict model, the 2PC protocol is used to terminate the global transaction atomically. 2PC is applied to solve the WAC problem, hence if the coordinator misses a vote it will decide on abort during the commit phase.

Since the uncertainty periods resulting from the strict model are of special interest within the following chapters, they are recalled briefly. In the strict model, 2PC is started at time t_p. I assume that all *prepare* messages are sent at the same time (t_p). Then the length of the critical window ΔU, where participants are vulnerable to a coordinator's node failure or a communication failure, is at minimum $\Delta U_{min} = 2\delta_m$ and at maximum $\Delta U_{max} = 2\delta_m + \Delta_{vo}$. In cases where no failure has occurred, the window has length ΔU_{min}, because all vote and final commit messages are delivered within δ_m. Otherwise, the coordinator must await timeout Δ_{vo}, so that ΔU increases to ΔU_{max}.

Figure 3.3 depicts ΔU where participants are vulnerable to blocking situa-

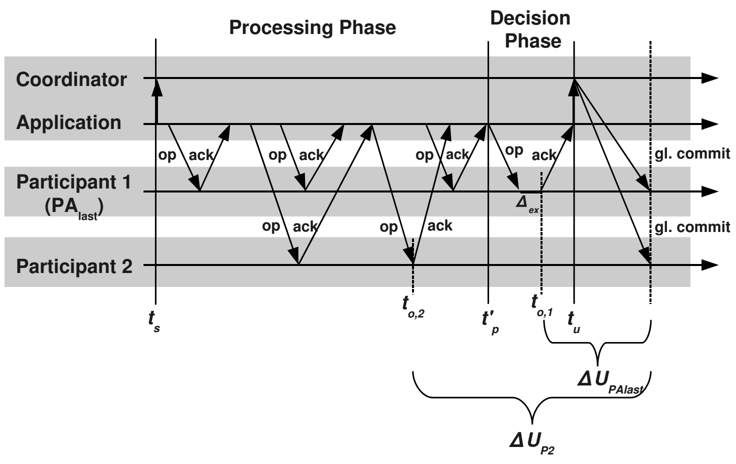


Figure 3.4: Semantic transaction model with EC.

tions. A node of failure of a participant PA or a communication failure of PA with the coordinator during ΔU causes a blocking (i) situation. ΔU is given by the interval $[t_p + \delta_m, t_p + 3\delta_m]$ if all other participants are correct or by $[t_p + \delta_m, t_p + 3\delta_m + \Delta_{vo}]$ if any other participant suffered a failure that causes the coordinator to await time-out Δ_{vo}. The intervals where a node failure of the coordinator causes a blocking (ii) situation is $[t_p, t_p + 2\delta_m]$ or $[t_p, t_p + 2\delta_m + \Delta_{vo}]$ in case a participant suffers a failure that prevents its vote message to be delivered within δ_m. If blocking (i) or blocking (ii) occurs, I consider the execution of a cooperative recovery scheme by blocked participants.

3.6.3 Semantic Transaction Model

The semantic model represents the class of advanced transaction models ensuring semantic atomicity as described in Section 3.6.3. In the semantic model, a local transaction branch terminates as soon its last operation is processed. Hence, *semantic atomicity* is maintained by means of compensating transactions, which semantically undo the effects of a committed branch. I assume that a participant knows if an operation is the last of a branch. Commit is processed according to the EC scheme, presented in Section 3.4.4.1.

While in strict atomicity, ΔU is the same for all participants, in semantic atomicity, ΔU is individual for each participant. This is due to the fact that a participant i commits its local transaction branch right after successfully executing its last operation at time t_o, moving into uncertainty afterwards. The acknowledgment of this operation is an implicit *yes* vote to the coordinator. The coordinator later derives the global decision, without requiring an additional vote from this participant. In fact, the last operation of the last participant PA_{last} decides the global transaction. Hence, in semantic atomicity, I define that the processing phase ends at t'_p, which is the time the coordinator sends the last operation to PA_{last}. Hence, execution and acknowledgment of this operation can be considered as the decision phase. The coordinator derives the global decision at $t_u = t'_p + 2\delta_m + \Delta_{ex}$, where Δ_{ex} is a constant time assumed for the execution of the last operation. This scheme is depicted in Figure 3.4.

It is easy to see that the size of the uncertainty window is generally wider with semantic atomicity. If no failure with any participant is detected by the coordinator, the individual uncertainty period of a participant i starts already at time $t_{o,i}$ and ends with $t_u + \delta_m$. If the coordinator detects a failure during the processing phase at time t_f, the global decision (abort) is decided before t'_p and the uncertainty window of i reduces to $[t_{o,i}, t_f + \delta_m]$. If $t_{o,i} > t_f$, i never enters uncertainty, since i receives the coordinator's decision before moving into uncertainty. A node failure of i or a communication with the coordinator while i is in its uncertainty period results in extended uncertainty (i), while a node failure of the coordinator while i is in its uncertainty window causes extended uncertainty (ii) as defined in Section 3.4.4.3.

If a participant suffers an extended uncertainty (i) or extended uncertainty (ii) situation it will initiate a cooperative recovery scheme, by contacting all other participants to derive the global decision.

3.7 Summary and Conclusion

This chapter has introduced principles of atomic transaction processing and discussed several Atomic Commit Protocols (ACPs) proposed within the past 30 years. I have presented the status quo on atomic commit problems and shown that blocking cannot be completely avoided in presence of node and communication failures in MANETs.

The blocking problem has been described for strict and semantic atomicity, and the relevant blocking situations to be regarded within the remainder of this work (i) caused by a participant failure and (ii) caused by a node failure of the coordinator have been defined.

I have presented some related work in the area of infrastructure-based wireless networks and discussed the basic rationales behind optimization of atomic commit protocols. Finally, the transaction models to be used in the remainder of this thesis have been defined.

The objective of this chapter was to present the theoretical foundation of atomic commit and related work in this area. It was made clear that at most the NB-WAC problem is solvable in a MANET if a majority of transaction participants remains connected to each other while a minority of participants remains blocked.

Fundamental enhancements of existing ACPs or the design of completely new protocols, e.g. solving NB-WAC are not expected for minimal system models. The direction followed by most scholars in this area is application driven. More efficient recovery protocols are developed by exploiting special characteristics of the system environment or application as observed in infrastructure-based networks. However, exploiting reliable parts of the system model like mobile-support stations is not an option in MANETs.

While solvability results of commit problems and characteristics of ACPs like message and time complexity have been intensively studied, the implication of these results for concrete scenarios have not been addressed yet. However, from a practical point of view, it is of interest how many blocking situations have to be expected for a given application and MANET scenario if a certain ACP is used. The currently available results, e.g. that in the case of network partitioning only participants residing in a small partition are blocked while in a partition with a majority of participants non-blocking is achieved, are of little use for a practical decision on which protocol to use in a given MANET scenario. Here, it must be decided for a given application whether an acceptable number of blocking situations can be expected or whether a high blocking probability hinders progress of an application and therefore deployment of the application to a MANET is not feasible. Additionally it is of interest how much the blocking risk is reduced in case a more reliable ACP like PC or Q3PC is used. To answer such questions, quantitative statements about blocking probabilities are required. In fact, the most fundamental question of whether blocking is actually a relevant problem in MANETs has not been answered yet.

Given the observation that in practice blocking situations in fixed wired networks are extremely rare (2PC is mostly used in practice) and the few blocking situations that occur can easily be tolerated without large delays in transaction processing, it is of interest whether the situation is fundamentally different in MANETs.

To answer the more general question whether atomic transaction processing is feasible at all in a given MANET scenario or whether high abort rates render transaction processing impossible, abort risks must also be analyzed for a MANET. The next step of my work is to answer precisely these questions. I will use the MANET system model defined in Chapter 2 to derive an analytical model that predicts the abort and blocking probabilities in MANETs for the strict and semantic transaction model defined in this chapter.

Chapter 4

Atomic Transactions in MANETs

The previous two chapters provided background information on MANETs and a theoretical view of the atomic commit problem and atomic commit protocols (ACPs). The aim of this chapter is to relate both areas and to examine the real-world behavior of atomic transactions in MANETs to investigate the relevance of blocking problems. This is motivated by the observation that in practice theoretically unreliable protocols such as 2PC are used in partially synchronous system, e.g. the internet, and the theoretical impossibility results on non-blocking atomic commit and consensus seem to have no relevant impact, since blocking risks are negligible and tolerable. The major objective of this chapter is to investigate whether the situation is similar for MANETs, i.e. I want to show the expected dimension of abort and blocking probability in MANETs. Until now, most research concerned with coordination problems in MANETs has simply presumed that, due to node mobility and resource constraints more reliable ACPs are required and hence that blocking is a relevant problem in MANETs, while quantitative statements about the expected impact of failures on coordination guarantees have not been provided.

However, general statements about abort and uncertainty rates in MANETs cannot be given, since an infinitive number of transaction and MANET scenarios exist and each combination shows individual failure characteristics. Therefore, I am presenting a probabilistic model that allows to predict the expected abort and blocking rate of transactions for any MANET scenario that is based on the MANET model introduced in Section 2.3.

The purpose of the probabilistic model developed here is twofold; on the one hand, it can be used to decide on the applicability of a transactional application in a MANET setting a priori by predicting expected abort rates. Without such a model it requires vast simulation studies to find out if a transactional system is applicable at all in a MANET environment, or if a more failure tolerant transaction model is indicated. For example, if for a transactional application 50 % of all transactions must be expected to abort, then transaction processing is not feasible. However, this decision depends on the individual application. On the other hand, the presented model can be used to optimize transaction processing at runtime by integrating more reliable ACPs only when actually required.

Based on the number of participants and duration of the processing phase, a coordinator can calculate the probability of the transaction to be aborted due to a communication or node failure as well as the risk of a participant to suffer a blocking situation. If these probabilities are unacceptable, the transaction can be rejected, or additional schemes like the Shared Log Space (SLS) as proposed in Chapter 5 or a backup coordinator as proposed in Chapter 6 can be embedded in commit processing to compensate for blocking. In short, the questions answered by the presented calculation model are:

- What is the probability that a transaction of the strict or semantic transaction model will abort due to node or communication failure in a given MANET scenario?

- What is the probability that a participant of a strict or semantic transaction will encounter a blocking (i) or extended uncertainty (i) situation that cannot be compensated for by cooperative recovery?

- What is the probability that a participant of a strict or semantic transaction will encounter a blocking (ii) or extended uncertainty (ii) situation caused by a node failure of the transaction coordinator?

To calculate the abort and blocking probability of a transaction, the transaction model and several transaction and MANET related parameters have to be extracted from the application scenario.

Transaction related parameters considered describe the transactions processed, i.e. (i) the size of the processing phase given by $t_p - t_s$, which is mainly determined by the number of operations issued within the distributed transaction; (ii) the number of participants denoted by n; (iii) the distribution of last operations described by the pdf $o(t_o)$; and (iv) time-out parameters defined by the transaction model, e.g. Δ_{ex} or Δ_{vo}.

Parameters from the MANET scenario are required to derive the probabilities of node, communication and general failures. These parameters are the pdfs and constants $f_E(t)$, $f_{ER}(t)$, $f_T(t)$, $f_L(t)$, $f_J(t)$, $f_C(t)$, $f_{CR}(t)$, δ_m, and δ_{to} defined by the MANET system model of this work.

I will present how these parameter are derived before the probabilistic model is developed. Therefore, this chapter is structured as follows: first, MANET related parameters are extracted from an example scenario. This example scenario is then used within the remaining chapters. The derivation of $f_C(t)$ and $f_{CR}(t)$ requires special attention, since an analytical evaluation is not possible, but simulations and statistical analysis are required. Afterwards, Section 4.4 calculates the probability of transaction abort in the strict and semantic transaction model. Calculations to predict blocking caused by participant failures are presented in Section 4.5, while Section 4.6 presents calculations to predict blocking caused by node failure of the coordinator. The results of Sections 4.4, 4.5, and 4.6 are then applied to a more concrete application scenario in Section 4.7. In the end, a detailed problem and research statement is derived, as well as a conclusion on the relevance of the blocking problem in MANETs is given.

Parts of the calculation models presented in this chapter have been published in [26, 25, 28].

4.1 MANET Parameters

In this section, I will present how the cdfs $F_N(t)$ and $F_C(t)$ describing the probability of node and communication failures can be derived for a concrete MANET scenario. As an example MANET scenario used within the remainder of this thesis, I assume the following setting based on a disaster recovery situation:

> 15 mobile recovery units move on a square of 500 m * 500 m according to the Random Way-point (RWP) mobility model at 2.0–5.0 mps, relaying messages for each other using AODV. Batteries of nodes are assumed to deliver 2 h of service, while the mean time to failure due to a technical failure is 500 h. The rescue units form a MANET $\mathcal{A}$ connected to other areas according to the AGB mobility model; each node has an expected sojourn time of 30 min before moving out of $\mathcal{A}$. For example, rescue units salvage injured persons from collapsed buildings in $\mathcal{A}$, and transport them to a rendezvous site outside of $\mathcal{A}$ for medical treatment. Mobile units are assumed to carry PDAs with IEEE 802.11–compliant radio adapters with approx. 120 m radio range. The time rescue units remain disconnected from $\mathcal{A}$ before reentering $\mathcal{A}$ is distributed exponentially with an expectation of 1 h.

To obtain the probability $F(t)$, that communication between two nodes is possible within t, the cdf for node failures $F_N(t)$ and for communication failures $F_C(t)$ has to be determined.

4.1.1 Probability of Node Failures $F_N(t)$

According to the failure model of this thesis a node failure causes the complete disconnection of a node from $\mathcal{A}$. The probability of this event is derived from the following probabilities: (i) disconnection caused by exhausted energy resources $f_B(t)$; (ii) disconnection due to a technical problem $f_T(t)$; or (iii) because the node moves out of the area of $\mathcal{A}$, given by the pdf $f_L(t)$.

$f_L(t)$ can be directly obtained from the AGB mobility model as defined in Section 2.2.3.1, since it takes the probability distribution of how long a node remains within one area as input. In this work, I am assuming exponentially distributed sojourn times, while any other distribution could be used without changing the calculation model presented here. In the real world, distribution of sojourn times may differ among nodes; e.g., a supply team dispatching supplies to rescue workers is expected to spend less time in $\mathcal{A}$ than a rescue team removing construction waste with heavy machinery. However, for the sake of simplicity, I assume $F_L(t)$ to be an exponential distribution with parameter $\lambda_L = 1/1800$ in the example scenario for all nodes of $\mathcal{A}$.

The probability that a node disconnects from $\mathcal{A}$ due to exhausted energy resources by t is denoted by the cdf $F_B(t)$. For a randomly chosen node from $\mathcal{A}$, it is unknown how long it has been operational, and hence how long its energy resources will last. If mobile nodes enter $\mathcal{A}$ with fully charged batteries and have a constant energy consumption, a randomly chosen node from $\mathcal{A}$ can be assumed to have remaining energy resources uniformly distributed in the interval $[0, b]$. b denotes the maximum service time of 7200 s as described above. If nodes are not assumed to enter $\mathcal{A}$ with fully charged batteries, then an exponential distribution with parameter $\lambda_E = 1/b$ is a feasible assumption for $f_E(t)$. It is then modeled that the probability of exhausted energy resources within an infinitesimally small time step is always the same, while the expected service time is b. However, both the uniform distribution over the interval $[0, b]$ as well as an exponential distribution are significant simplifications, since in reality, the remaining energy is influenced by fluctuating power consumption, which is subject to numerous influences and therefore hard to capture analytically. However, it will be shown later that the probability of transaction abort or blocking due to exhausted energy resources is small and negligible. Additionally, energy related failures can be easily anticipating by simply excluding participants with low energy resources from a transaction at runtime. Therefore, a raw estimate is sufficient and favored over accurate modeling here. In the following, I will mostly use the uniform distribution over $[0, b]$ for calculations if not stated differently.

The disconnection from $\mathcal{A}$ caused by a technical failure is also a rare event, if PDAs or laptops are assumed as mobile devices. The scenario description states that the mean time to failure is given by 500 h, hence an exponential distribution with $\lambda_T = 1/(18 * 10^4)$ is a meaningful assumption for $F_T(t)$, which results in a negligible probability of node failures due to technical defects. However, in other scenarios with much cheaper hardware, like sensor nodes, $F_T(t)$ may become more relevant.

If for $f_E(t)$ a uniform distribution over $[0, b]$ is assumed, the probability that a node failure happens within time t is given by:

$$F_N(t) = 1 - \left((1 - \frac{t}{b}) \cdot \left[1 - (1 - e^{\frac{t}{\lambda_L}})\right] \cdot \left[1 - (1 - e^{\frac{t}{\lambda_T}})\right]\right) \tag{4.1}$$

Note that in Formula (4.1), the case $t > b$ is neglected, as I am not concerned with long-lived but OLTP transactions in this work and therefore with small values of t_p. In Section 4.4, I will show that transactions with a processing phase larger than 100 s are not feasible in the example scenario.

4.1.2 Communication Failures $F_c(t)$

The failure model of this work defines a communication failure as all events that lead to an outage of the communication between two nodes in $\mathcal{A}$, while both communication partners are connected to $\mathcal{A}$. A communication path that was functional before breaks, if a direct link between two nodes on the path suddenly becomes unavailable. If no multi-hop routing is used, every link break immediately causes a communication failure, while with multi-hop routing, the underlying routing scheme possibly provides an alternative route.

The probability that a communication path is available until time t is described by the cdf $F_C(t)$, which is primarily influenced by the node density, radio range of nodes, node mobility, and the routing scheme in $\mathcal{A}$. I also show in the following that $F_C(t)$ also depends on the hop count of the path when communication is initiated. As it is complicated to model the numerous dependent events that cause the break of a communication path, most scholars propose statistical analysis of path duration, i.e. of $F_C(t)$, based on simulation studies. For example, in [10, 128] the distribution of path durations for different mobility models is derived by simulation. In [67, 87] an analytical approach is also proposed to approximate the distributions of path durations. However, [67, 87, 10, 128] show that the underlying mobility model impacts path and link durations, but for the most common mobility models, such as RWP, Manhattan Mobility [96], and Freeway Mobility [129], an exponential distribution of path durations for routes with more than 2 hops is a reasonable approximation. The work cited above solely considers paths with more than two hops (in the following denoted by 2+ hop count) and derives exponentially distributed path durations. In contrast, I am especially interested in the probability distribution of paths with 1–2 hops. This is due to the fact that the abort rate for transactions initiated in 1–2 hop distances is considerably smaller than for transactions initiated with participants in arbitrary hop distances, as shown in Section 4.4. In fact, transaction processing with participants in 2+ hops distances mostly shows such a high abort probability in the example scenario that the feasibility of transaction processing must be questioned.

To derive $F_C(t)$ for 1–2 hop paths, as well as for 2+ hop paths in the example scenario, I present a simulation study using the *ns2* network simulator. The simulation considers movement in $\mathcal{A}$ only, where 15 nodes move according to the RWP mobility model within an area of 500 m * 500 m, as assumed in the example scenario, with speeds between 2.0 and 5.0 mps and a pause time of 1 s before choosing a new way-point.

The following behavior of nodes was simulated in *ns2*: two nodes in 1–2 or 2+ hop range were randomly chosen and a probe message was exchanged every second between these nodes. A node receiving a probe answered with an acknowledgment message. The time until a communication path breaks, i.e. the time when no acknowledgment for a probe message was received anymore, was measured as well as message delays of all messages exchanged. A more detailed description of the simulation settings, e.g. antenna characteristics etc., are given in Appendix B.1.1.1.

The resulting histograms showing the frequencies of measured path durations are given by Figure 4.1. Analysis of the durations of paths initiated in 2+ hop range shown in Figure 4.1(b) confirms the results of [67, 87, 10, 128]. For these paths, an exponential distribution of path durations can be presumed. Figure 4.1(b) shows an exponential distribution with parameter λ=0.0514 fitting the measured distribution, where λ is derived using the Maximum Likelihood method. An important observation to be made here is the high probability of very short path durations in the example scenario, i.e. 22% of the paths do not survive 5 seconds. I show later that the abort rate in such a setting is not acceptable, even for very short transactions.

A different picture is obtained for paths that are initiated in 1–2 hop distances. The resulting histogram of path durations in Figure 4.1(a) shows a very different shape compared to 2+ hop paths. What is important here is the high probability that a path will survive the first seconds after initiation, e.g. in the example scenario, 99 % of all links survive the first 5 seconds. After a period with a small risk for a path break right after path initiation, the risk increases quickly, as shown in Figure 4.1(a). After 40 s about 60 % of all links have to be expected to fail. The path characteristics of 1–2 hops paths is accurately approximated by a log-normal distribution, as shown by the red curve in Figure 4.1(a) with parameters μ=3.5343 and σ=0.6770 for the example scenario. Again, standard techniques such as the Maximum Likelihood method have been used to estimate these parameters.

To demonstrate the influence of node speed $F_C(t)$, Figure 4.1(c) depicts the results for a simulation assuming the parameters of the example scenario, while node speeds are reduced from 2.0–5.0 mps to 1.0–2.0 mps. With reduced node speeds path durations are obviously higher, while the distribution of path durations is still accurately modeled by a log-normal distribution as shown by the red curve in Figure 4.1(c).

Based on the results presented above and related work [67, 87, 10, 128] it can be concluded that log-normal and exponential distributions can be assumed for $f_C(t)$ for the common mobility scenarios at moderate and high node mobility. For the purposes of this work I derived $f_C(t)$ by simulation, while in future approaches to derive $f_C(t)$ analytically might become available. However, for the development of the probabilistic model presented in this chapter it is irrelevant how $f_C(t)$ is derived.

While an exponential distribution of communication failures eases calculations significantly due to the memoryless property of exponential distribution functions, the probabilistic models presented in this chapter can take arbitrary distributions as input, as the model does not make any assumptions about the type of the input pdfs.

I will mostly consider a log-normal distribution for $f_C(t)$ as this would be a realistic choice for the example scenario as shown later.

4.1.3 General Failure $F(t)$

The event that a node connected to $\mathcal{A}$ suddenly cannot communicate with another node anymore, that was reachable before is defined as the general failure. The probability of a general failure happening within time t is given by $F(t)$ and describes the probability that a communication failure between the two nodes will occur or that the communication partner will disconnect from $\mathcal{A}$, i.e. the communication partner suffered a node failure.

Given the cdfs derived above, $F(t)$ of the example scenario if 1–2 hop paths are assumed is given by Formula (4.2):

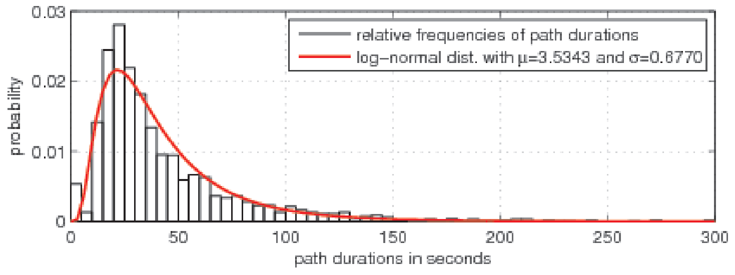
$$F(t) = 1 - \Big((1 - \frac{1}{\sqrt{2\pi} \cdot \sigma} \int\limits_0^t \frac{1}{x} e^{-(lnx-\mu)^2/2\sigma^2} dx) \cdot \big[(1 - \frac{t}{b}) \cdot (e^{-t \cdot (1/(\lambda_L + \lambda_T))}) \big] \Big) \tag{4.2}$$

with values for the parameters σ, μ, λ_T, λ_L, and b as derived in Section 4.1.1.

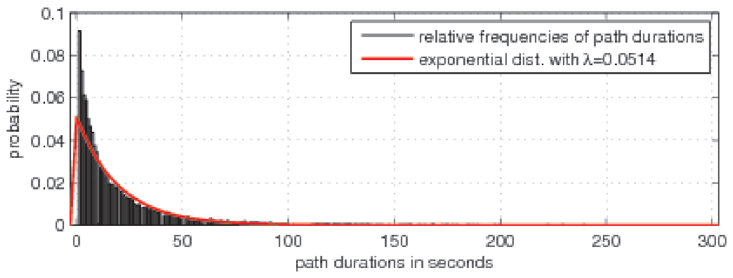
Figure 4.2 shows a plot of $F(t)$ for the example scenario to demonstrate the major influence of $F_C(t)$ on the general failure.

Figure 4.2 shows that, for the example scenario, node failures are almost negligible. Even if nodes move slowly at 1.0–2.0 mps (red curves), the influence of node failures on the general failure rate is small compared to communication failures in this setting. For example, at t=50 s consideration of node failures raises the overall failure probability from 18 % to 22 %, as shown by the red dashed and the red solid curve in Figure 4.2. At at an increased node speed of 2.0–5.0 mps, the effect of node failures on the general failure is marginal as shown by the green curves in Figure 4.2.

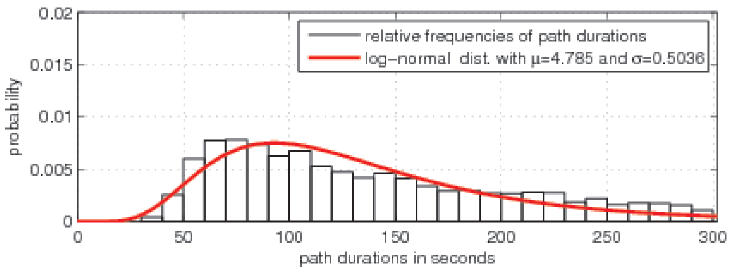
The influence of node failures might have a greater influence in other scenarios, e.g. where transition rates between MANET areas are significantly higher, or more failure-prone hardware like sensors nodes are employed. Therefore, node



(a) Relative frequencies of path durations for paths initiated in 1–2 hop distance, measured in the example scenario with AODV multi-hop routing.



(b) Relative frequencies of path durations for paths initiated in 2+ hop distance, measured in the example scenario with AODV multi-hop routing.



(c) Relative frequencies of path duration for paths initiated in 1–2 hop distance, measured in the example scenario at reduced speeds of 1.0–2.0 mps with AODV multi-hop routing.

Figure 4.1: Histograms of measured path durations based on 10000 tests.

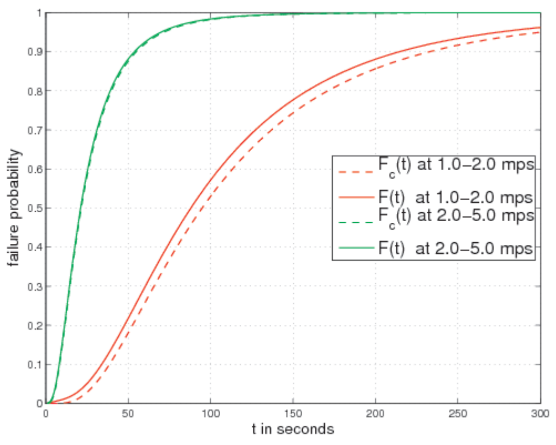


Figure 4.2: Influence of node failures on $F(t)$.

failures are considered within all calculations in the following, although their impact is small in the example scenario.

One important property of communication failures is that they are assumed to eventually recover if both communication partners remain in $\mathcal{A}$. To understand to what extent failures are transparent to transaction processing, $f_{CR}(t)$ has to be derived.

4.1.4 Probability of Path Recovery $F_{CR}(t)$

As described in the system model of this work, communication failures are subject to recovery. The time that messages between two nodes cannot be delivered, is distributed by $f_{CR}(t)$ (see system model Section 2.3.1).

The time that communication between two nodes is unavailable is influenced by multiple factors. The network density and network size $n_{\mathcal{A}}$ influence the probability that an alternative path can be found, while node mobility influences the probability that new paths are formed. Additionally, the routing scheme plays an important role, as the time required to detect an invalid route and to initiate discovery of an alternative route differs for multi-hop routing schemes.

Proactive routing like DSDV recognizes broken routes more quickly, since topology changes are constantly propagated through the network. DSR maintains multiple paths for one destination, while AODV maintains only one route per destination and therefore has to perform a route discovery whenever a path breaks. As all these factors are hard to grasp analytically, I propose a simulation study using the same simulation as in Section 4.1.2, with the difference that exchange of the probe message is continued after the path breaks and the time is measured until the probe message is received again. For the example scenario with AODV routing, Figure 4.3(a) shows the resulting histogram of measured path outage times. It can be observed that new paths are found with a probability of 14 % after 10 s. This is explained by the fact that AODV awaits a time-out before a stale route is considered to be broken and discovery of a new route is initiated. A log-normal distribution here fits the distribution of the path outage periods, if the delay δ_{PB} is considered that describes the time

AODV requires to detect the path break, as shown by the dotted line in Figure 4.3(a). $f_{CR}(t)$ is then given by:

$$f_{CR}(t) = \begin{cases} \frac{1}{(t-\delta_{PB})\sigma_r\sqrt{2\pi}} \cdot e^{\left(-\frac{(ln(t-\delta_{PB})-\mu_r)^2}{2\sigma_r^2}\right)} & for\ t > \delta_{PB} \\ 0 & for\ t \leq \delta_{PB} \end{cases} \tag{4.3}$$

The influence of the routing scheme and node speeds on path recovery is demonstrated by Figure 4.3(b) and 4.3(c). Figure 4.3(b) shows the distribution of path outages of the example scenario and node speeds of 2.0–5.0 mps if no multi-hop routing is used, i.e. communication is only possible between nodes in direct radio range. Here, long outage periods are more likely than in scenarios with AODV. However, a log-normal distribution with parameters μ=4.78 and σ=1.34 provides a good approximation to these frequencies.

Figure 4.3(c) shows the effect of slow moving nodes (1.0–2.0 mps) if no multi-hop routing scheme is considered. Here, the duration of path outages is more widely distributed, and very long outage periods of up to 500 s are possible.

I will show later that $f_{CR}(t)$ has a strong influence on abort and blocking probabilities if a multi-hop routing scheme is used.

4.1.5 Summary - MANET Parameters

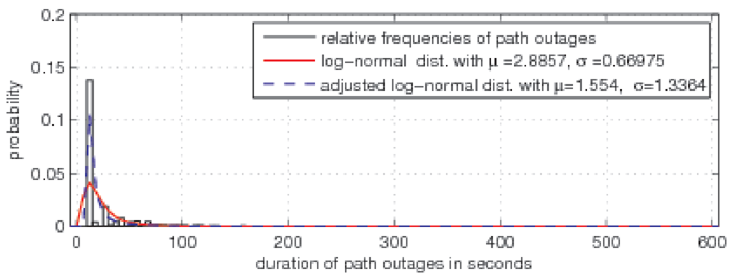
In this section, I demonstrated how the parameters required by the MANET system model can be obtained for a given MANET scenario. Since convenient analytical approaches to derive cdfs $F_C(t)$ and $F_{CR}(t)$ and message delay δ_m are yet not available yet, I proposed a simple simulation study to estimate these parameters statistically. However, the MANET research community is actively working on approaches to derive $F_C(t)$ and $F_{CR}(t)$ analytically, e.g. [67, 87]. Such models can be possibly used in future, since it is irrelevant for this work how these distributions are derived. However, it is out of the scope of this thesis to develop such analytical approaches. For the purpose of this work, simulation provides a convenient way to learn about the characteristics of a MANET scenario and to derive the desired cdfs.

4.2 Transaction Parameters

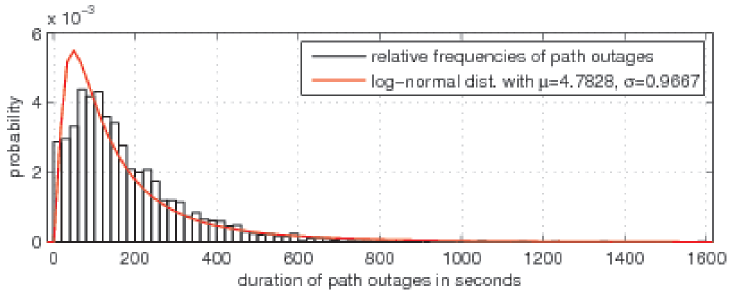
As described in Chapter 3, the parameters of a transaction are: (i) the transaction model; (ii) the size of its processing phase $[t_s, t_p]$; (iii) the number of participants n and (iv) the distributions of the last operations assigned to participants given by $o(t_o)$.

I do not propose an example transaction scenario to derive a fixed set of parameters like in the previous section, but the four transaction parameters will be varied systematically to demonstrate their influence on abort and blocking risks.

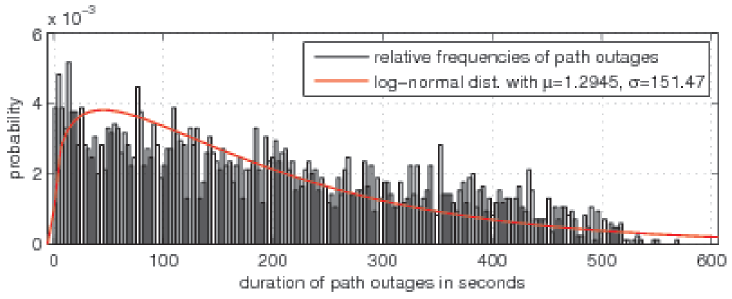
Whether the strict or the semantic transaction model is assumed depends on application semantic, i.e. if a distributed database application or a non-traditional application is considered. In the following, both models are considered by the calculation models proposed.



(a) Relative frequencies of first outage durations with AODV for paths initiated in 1–2 hops distance.



(b) Relative frequencies of first outage durations without multi-hop routing, for link initiated in 1–2 hops distance.



(c) Relative frequencies of first outage durations without multi-hop routing and slow node movement of 1.0–2.0 mps for hops initiated in 1–2 hops distance.

Figure 4.3: Distribution of path outages measured in 10000 test for the example MANET scenario.

The size of the processing phase $[t_s, t_p]$ mainly depends on the number of operations issued to participants and the grade of parallelism of operation allocation. Given an execution delay Δ_{ex}, the number of operations issued, and δ_m, it is obviously possible to approximate the size of the processing phase for a concrete transaction. I present an example of how the size of the processing phase is derived for an example scenario in the end of this chapter.

The number of nodes n participating in a distributed transaction is assumed to be small, i.e. 2–6 participants. The exact value of n depends on the concrete application on hand. In the following, I will mostly assume 3 participants, but I will also vary n to examine its influence on abort and blocking rates.

The time t_o the last operation of a participant's transaction branch is executed has great influences on abort and blocking rates in the strict and the semantic transaction model. A general failure with a participant during the processing phase is detected at latest at t_o. Therefore, t_o defines the interval $[t_o, t_p]$ where a communication failure with a participant or a node failure of the participant is not detected by the coordinator, because no messages are exchanged. A failure in $[t_o, t_p]$ will first be detected within the decision phase of the transaction. In the semantic transaction model, t_o is also the time where a node enters its uncertainty window.

For the calculation model presented in the next subsections, I assume that t_o follows the same probability distribution $o(t_o)$ for all participants. In reality, the distribution of operations during the processing phase depends on the application and the role of each participant. Arbitrary distributions for $o(t_o)$ can be imagined. For sake of simplicity, I assume a uniform distribution of t_o over t_p, i.e.

$$o(t_o) = 1/t_p \tag{4.4}$$

where t_p is the size of the interval $[t_s, t_p]$ with $t_s = 0$.

In the following, the MANET and transaction parameters are used as input for a calculation model that predicts abort and blocking risks of transactions in MANETs. The example MANET scenario defined above is used as the main scenario while transaction parameters are varied.

4.3 Preliminary Considerations

In this section, I present some preliminary calculations, which are frequently used. In many cases, I calculate the probability that an event happens during an interval $[t_1, t_2]$. Given a cdf F, it is computed by the difference $F(t_2) - F(t_1)$. In the following I use the notation $F(t_1..t_2)$ for this probability. For most calculations, I present two variants, one considering a single recovery cycle of communication paths and another neglecting recovery of communication paths. If a cdf F describes the variant that does not consider recovery, then F' denotes the expression considering a single recovery cycle.

Important preliminary results are the probabilities that a transaction actually enters the decision phase of a transaction. The decision phase is not entered if the transaction is aborted before, because the coordinator detects a participant's failure. These probabilities are derived in the following.

4.3.1 Recognized Failures in the Processing Phase

A participant's node or communication failure is only detected in the processing phase if it happens within the interval $[t_s, t_o]$, since then the coordinator would observe that an operation has not been acknowledged. This event occurs if a participant suffers from a failure at time t_f before the last operation is processed at time t_o, hence if $t_o > t_f$. $P_{o>f}(t_p)$ denotes the probability that a participant's failure happens in the interval $[t_s, t_o]$ and thus is detected by the coordinator. $P_{o>f}(t_p)$ is given by

$$P_{o>f}(t_p) = \int\limits_0^{t_p} \int\limits_0^{t_o} o(t_o) \cdot f(t_f)\, dt_f dt_o \tag{4.5}$$

if $t_s = 0$. The bounds of the integrals in $P_{o>f}(t_p)$ are chosen by the following consideration: if $t_o \in [0, t_p]$, then t_f must occur in the interval $[0, t_o]$.

I use the subscript of $o > f$ to indicate the failure type considered; i.e. $P_{o>f_N}(t_p)$ denotes the probability that a node failure occurs and is recognized by the coordinator, while $P_{o<f_C}(t_P)$ describes the same for communication failures.

If (i) a log-normal distribution with parameters μ and σ for communication failures, (ii) node failure probabilities as derived above, and (iii) $o(t_p) = 1/t_p$ are applied to $P_{o>f}(t_p)$, then $P_{o>f}(t_p)$ is given by

$$\begin{aligned} P_{o>f}(t_p) &= \int\limits_0^{t_p} \frac{1}{2t_p} \cdot \left[3 - 2e^{-\lambda_L t_o} - 2e^{-\lambda_T} + \sqrt{\frac{1}{\sigma^2}}\sigma \right. \\ &\qquad \left. + \frac{2t_o}{b} + \text{Erfc}\left[\frac{\mu - ln(t_o)}{\sqrt{2}\sigma}\right]\right] dt_o \end{aligned} \tag{4.6}$$

while Erfc denotes the complementary error function. The complementary probability $1 - P_{o>f}(t_p)$ is the probability that a failure occurs and is *not* detected *or* that no failure happens during $[t_s, t_p]$.

4.3.2 Unrecognized Failures in the Processing Phase

A failure of a participant during $[t_s, t_p]$ is not recognized if the failure happens after the last operation was acknowledged. Hence, if $t_o \in [t_s, t_p]$, then t_f has to be from the interval $[t_o, t_p]$ for this event to happen. The probability that a failure occurs in $[t_s, t_p]$ *and* is *not* recognized by the coordinator is denoted by $P_{o<f}(t_p)$ and is given by

$$P_{o<f}(t_p) = \int\limits_0^{t_p} \int\limits_{t_o}^{t_p} o(t_o) \cdot f(t_f)\, dt_f dt_o \tag{4.7}$$

if t_s=0. Using a log-normal distribution for $f_C(t)$ and $f_N(t)$ as derived for the example MANET scenario of this work, $P_{o<f}(t)$ given by

$$\begin{aligned} P_{o<f}(t_p) &= \int\limits_0^{t_p} \frac{\left(e^{-\lambda_L t_o} + e^{-\lambda_T t_o} - e^{-\lambda_L t_p} - e^{-\lambda_T t_p} + \frac{t_p - t_o}{b}\right)}{t_p} \\ &\qquad + \frac{\text{Erf}\left[\frac{\mu - ln(t_o)}{\sqrt{2}\sigma}\right] - \text{Erf}\left[\frac{\mu - ln(t_p)}{\sqrt{2}\sigma}\right]}{2t_p}\, dt_o \end{aligned} \tag{4.8}$$

The complementary probability $1 - P_{o<f}(t_p)$ describes the probability of the events that either *no* failure occurs during $[t_s, t_p]$ or that a failure occurs *and* is recognized.

4.3.3 Unrecognized Failure and Recovery

As described in the system model and in Section 4.1.4, communication failures are subject to recovery, and the first random outage time of a communication path is described by the pdf $f_{CR}(t)$. A result frequently used is the probability of an agnostic failure in $[t_s, t_p]$. The coordinator reacts agnostically to a failure of a participant if the failure happens after t_o and recovers by t_p. Given the pdf of path outage $f_{CR}(t_r)$, last operation $o(t_o)$, and communication failure $f_C(t_f)$, the probability of an agnostic communication failure is given by $P_{o<f_C,r}(t_p)$:

$$P_{o<f_C,r}(t_p) = \int_0^{t_p}\int_0^{t_f}\int_0^{t_p-t_f} f_{CR}(t_r) \cdot o(t_o) \cdot f_C(t_f)\, dt_r dt_o dt_f \tag{4.9}$$

if t_s=0. The bounds of the integrals in $P_{o<f_C,r}(t_p)$ are chosen according to the following consideration: if $t_f \in [0, t_p]$ then the last operation must have been acknowledged before the failure occurs and hence, $t_o \in [0, t_f]$. The outage period of communication must be smaller than the time until the end of the decision phase is reached and therefore $t_r \in [0, t_p - t_f]$.

Note that $P_{o<f_C,r}(t_p)$ assumes only a single failure and recovery cycle, while in reality multiple failure and recovery cycles may occur over time. However, the probability of multiple failure and recovery cycles is negligible for the short transactions considered in this work. Simulation results presented later show that accurate predictions are derived by considering only one recovery cycle. If for $f_C(t)$ and $F_{CR}(t)$ exponential distributions are assumed, consideration of multiple failure and recovery cycles is possible, because the memoryless property of exponential distributions can be exploited to model a stochastic process describing the states of a path. Such calculations are omitted here, but their integration in the calculation model presented here is straightforward.

The complementary probability $1 - P_{o<f_C,r}(t_p)$ describes the probability that (i) no failure happens during $[t_s, t_p]$, *or* (ii) that a failure is experienced and recognized, *or* (iii) that a failure is not recognized *and* does not recover until t_p.

Recovery from a node failure was not considered here since the time horizon of recovery from a node failure is in the dimension of several minutes. Hence, the probability that a node failure is not recognized and recovers within the processing phase is negligible. In contrast, for calculations of the probability that a failure is not recognized by the coordinator and does not recover before t_p, node failures have to be considered as shown in the following.

4.3.4 Unrecognized Failure and no Recovery

The event that a participant suffers from an unrecognized failure in the interval $[t_s, t_p]$ and the failure does not recover until t_p may occur in two situations: (i) the participant suffers from a communication failure that does not recover until t_p, or (ii) the participant suffers from a node failure.

The probability of the first event is calculated by $P_{o<f_C,nr}(t_p)$:

$$P_{o<f_C,nr}(t_p) = \int_0^{t_p}\int_0^{t_f}\int_{t_p-t_f}^{\infty} f_{CR}(t_r)\cdot o(t_o)\cdot f_C(t_f)\,dt_r dt_o dt_f \tag{4.10}$$

if t_s=0, where the bounds of the integrals are chosen as follows: if $t_f \in [0, t_p]$, then $t_o \in [0, t_f]$ and the outage time of the communication path must exceed the remaining processing phase, hence $t_r \in [t_p - t_f, \infty]$. If situation (ii) is considered, the probability that a general failure happens in $[t_s, t_p]$, which is *not* recognized and does *not* recover by t_p, is given by $P_{o<f,nr}(t)$:

$$P_{o<f,nr}(t_p) = 1 - \left[1 - P_{o<f_N}(t)\right]\cdot\left[1 - P_{o<f_C,nr}(t)\right] \tag{4.11}$$

Given the preliminary considerations above, the remainder of this chapter is concerned with the prediction of abort and blocking probabilities in the strict and semantic transaction models.

4.4 Abort Probability

The probability that a transaction is aborted due to a node failure or a communication failure is useful at runtime to decide on whether abort risks of a given transaction can be tolerated or if the transaction is better rejected. Additionally, being able to predict abort probabilities also allows to decide whether transaction processing is feasible at all in a certain application and MANET scenario at design time.

While the abort rate that an application can live with depends on the individual application semantics, I assume that an abort rate larger than 20 % is not tolerable for most applications in the disaster scenario assumed here.

Calculation of the abort probability is also crucial for the derivation of blocking probabilities, since blocking situations can only occur if the transaction has not been aborted before. Hence, a high abort rate reduces the probability of blocking. This effect is especially strong in the strict model, where blocking cannot occur during the processing phase and abort during this phase renders blocking impossible.

In the following, I present a calculation model to derive the abort probabilities for the strict and semantic transaction models. The model is applied to the example MANET scenario of this chapter.

4.4.1 Abort Probability in the Strict Model

In the strict transaction model, a transaction can be aborted by the coordinator during the processing phase $[t_s, t_p]$ or during the decision phase $[t_p, t_p + \Delta U]$. Both events are mutually exclusive and considered separately in the following.

Abort is decided in $[t_s, t_p]$ if the coordinator misses an acknowledgment for an operation and within $[t_p, t_p + \Delta U]$ if a vote of a participant does not arrive within time-out Δ_{vo}. First, I will consider the probability of abort in the interval $[t_s, t_p]$ and afterwards for $[t_p, t_p + \Delta U]$.

4.4.1.1 Abort Probability in the Processing Phase

The probability that in a transaction with n participants all participants either do not suffer from a failure within $[t_s, t_p]$ or the failure is not recognized, is calculated by $\left[1-P_{o>f}(t_p)\right]^n$. The complement of $\left[1-P_{o>f}(t_p)\right]^n$ describes the probability that at least one participant suffers a recognized failure in $[t_s, t_p]$. This is the probability of a transaction to abort in $[t_s, t_p]$ due to a participant failure.

If the coordinator suffers from a node failure within $[t_s, t_p]$, participants will abort unilaterally at $t_p + \Delta U_{max} + \delta_m$. This is safe, as no participant will move into prepared state, since no prepare messages can arrive. The probability that a transaction is aborted can now be calculated by the probability that either a recognized participant failure *or* a node failure of the coordinator happens within $[t_s, t_p]$ denoted by $P_{a_p}(t_p)$.

$$P_{a_p}(t_p) = 1 - \left[1 - P_{o>f}(t_p)\right]^n \cdot \left[1 - F_N(t_p)\right] \tag{4.12}$$

4.4.1.2 Abort Probability in the Decision Phase

In interval $[t_p, t_p + \Delta U]$, a transaction is aborted if the coordinator misses the vote of a participant after awaiting a timeout Δ_{vo}. This can happen either because a participant has not received a prepare message or its vote message cannot be transmitted due to a communication failure. The prepare message is not received in three situations: (A) if a participant suffers an unrecognized failure that does not recover until t_p; (B) if the prepare message is lost due to a general failure of a participant in $[t_p, t_p + \delta_m]$; and (C) if a participant receives the prepare message, but its vote message is lost due to a communication failure within the interval $[t_p + \delta_m, t_p + 2\delta_m]$.

If recovery of communication failures is not considered, the probability that at least one of n participants experiences situation A while the coordinator does not suffer from a node failure is given by $PA(t_p)$.

$$PA(t_p) = \Big(\left[1 - P_{o>f}(t_p)\right]^n - \left[1 - F(t_p)\right]^n\Big) \cdot \left[1 - F_N(t_p)\right] \tag{4.13}$$

Node failures of the coordinator within $[t_p, t_p + \Delta U]$ are not considered, as this situation causes blocking, which is considered in Section 4.6.

The probability of situation B is given by $F(t_p..t_p + \delta_m)$, that of C by $F_C(t_p + \delta_m..t_p + 2\delta_m)$. Since the events B and C are not independent, the probability for the event that B or C occurs is calculated by $PBC(t_p)$.

$$\begin{aligned} PBC(t_p) &= F(t_p..t_p + \delta_m) + F_c(t_p + \delta_m..t_p + 2\delta_m) \\ &\quad - F(t_p..t_p + \delta_m) \cdot F_c(t_p + \delta_m..t_p + 2\delta_m) \end{aligned} \tag{4.14}$$

The probability that abort is decided in the decision phase if recovery of communication paths is not considered is now given by

$$P_{a_d}(t_p) = PA(t_p) + \left[1 - F(t_p)\right]^n \cdot PBC(t_p) \tag{4.15}$$

In the case where recovery of communication paths is considered, an additional event must be regarded for situation A. This is the situation that at least one of n participants suffers from an unrecognized failure in $[t_s, t_p]$ that does not recover by t_p, while the other participants do not suffer from a failure in $[t_s, t_p]$ or an unrecognized failure occurs which recovers by t_p. This probability is calculated by

$$\begin{aligned} PA'(t_p) &= \sum_{i=1}^{n} \begin{pmatrix} n \\ i \end{pmatrix} P_{o<f,nr}(t_p)^i \\ &\cdot \sum_{j=0}^{n-i} \begin{pmatrix} n-i \\ j \end{pmatrix} [1-F(t_p)]^j \cdot P_{o<f,r}(t_p)^{n-i-j} \end{aligned} \tag{4.16}$$

If recovery of communication paths is considered for situation B and C, the probability that at least one participant suffers from B or C has to consider that participants may suffer a failure that recovers by t_p. Therefore, the probability that, for at least one of n participants, event B or C occurs is given by $PBC'(t_p)$.

$$PBC'(t_p) = \sum_{i=0}^{n} [1-F(t_p)]^i \cdot P_{o<f_C,r}(t_p)^{n-i} \cdot \left(1 - [1 - PBC(t_p)]^i\right) \tag{4.17}$$

The probability of abort in the decision phase, considering one recovery cycle of communication failures, is now given by $P'_{a_d}(t_p)$.

$$P'_{a_d}(t_p) = PA'(t_p) + PBC'(t_p) \tag{4.18}$$

The overall risk of a transaction to abort is now simply calculated by considering the probability that abort is decided in interval $[t_s, t_p]$ or $[t_p, t_p + \Delta U]$. If no recovery of communication is considered, $P_a(t_p)$ gives the overall abort probability.

$$P_a(t_p) = P_{a_p}(t_p) + P_{a_d}(t_p) \tag{4.19}$$

$P'_a(t_p)$ analogously gives the overall abort probability if a single recovery cycle of communication failures is assumed.

$$P'_a(t_p) = P_{a_p}(t_p) + P'_{a_d}(t_p) \tag{4.20}$$

4.4.1.3 Predictions and Simulation Results

In the following, the calculation model derived above is applied to the example MANET scenario and theoretical results are compared to measurements of a simulation study. Results are presented for strict transactions with n=3 and processing phases of 1–200 s, while additionally node speeds, initiation distance, and routing schemes are varied to demonstrate their influence on transaction abort.

Experiments have been done using the *ns2* network simulator with the mobility and radio settings of the example MANET scenario. For a detailed description of the settings see Appendix B.1.1.1. The following was implemented in *ns2* to simulate a transaction of the strict model: a transaction is initiated by choosing a random node to act as coordinator. n participants are randomly chosen from all nodes in 1–2 hop distance to the coordinator. The exact process of how coordinator and participants are chosen and simulation scripts are created is described in Appendix B.1. For every participant, time t_o is calculated using $o(t_p)$ and beginning with transaction start, the coordinator issues an operation message every second to all participants that have not reached their t_o. A participant receiving such a message immediately replies with an acknowledgment. If the coordinator does not receive an acknowledgment for an operation message within time-out δ_{to}=1 s, the transaction is aborted. In case acknowledgments for all issued operations are received by t_p, 2PC is initiated. Here, abort is decided if a vote times-out after Δ_{vo}=1 s.

Figure 4.4 compares the abort rates predicted by the proposed calculations, with measurements obtained from the simulation study. Figures 4.4(a), 4.4(b), and 4.4(c) compare predicted and measured abort rates in the example MANET scenario (a) with AODV routing, (b) without routing, and (c) with slow moving nodes (1.0–2.0 mps). In the following, I will first analyze abort rates in the processing and decision phase, before the overall abort rate and the influence of hop distances at initiation time is discussed.

Abort in Processing and Decision Phase

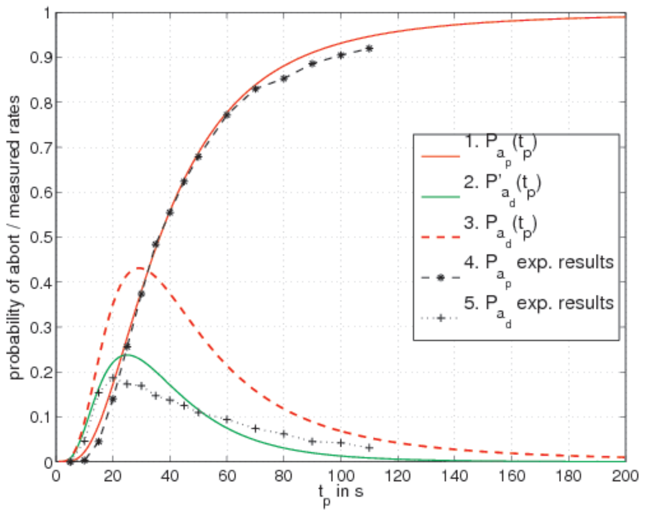
Generally, it can be observed that the predicted abort rates in the processing phase approximate the experimental rates accurately. For example, Curve 1 and 4 of Figure 4.4(a) show that at a processing phase of 40 s, $P_{a_p}(t_p)$ predicts an abort rate of 55.7 %, while the measured rate of transactions aborted in the processing phase is 55.4 %. A similar accuracy of predictions is achieved in the scenario where no multi-hop routing is used as shown in Figure 4.4(b) by Curve 1 and 4, and if node speeds are decreased to 1.0–2.0 mps as depicted in Figure 4.4(c) by Curve 2 and 5.

The probability of an abort decision in the processing phase is monotonically increasing over t_p and converges to 1. $P_{a_p}(t_p)$ has a log-normal like shape, as its major influence is the probability for communication failures during the processing phase. For the example scenario, it is shown that the abort risks within the processing phase increases fast, e.g. with node speeds of 2.0–5.0 mps and multi-hop routing the abort probability during the processing phase is greater than 90 % for t_p>100 s as shown in Figure 4.4(a).

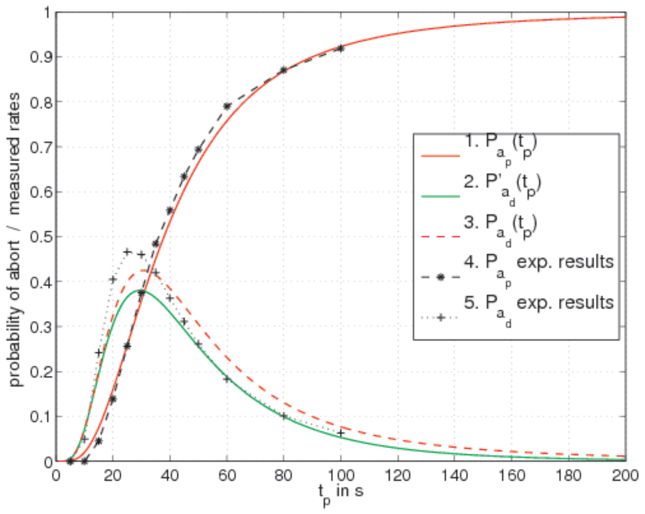
In contrast to $P_{a_p}(t_p)$, the probability of transaction abort in the decision phase $P_{a_d}(t_p)$ converges to 0 for increasing t_p as shown in the three Figures 4.4(a), 4.4(b), and 4.4(c), while for small t_p abort in the decision phase is more likely than in the processing phase.

Figure 4.4(a) shows that neglecting the effect of path recovery leads to predictions higher than the actually observed rates, e.g. for t_p=40 s, $P_{a_d}(t_p)$ predicts a 37.4 % abort probability, while the measured rate is only 13.7 %. Here, $P'_{a_d}(t_p)$ considering path recovery has to be used to derive more realistic approximations as shown by Curves 2 and 5 in Figure 4.4(a). It can be further observed that for small t_p, the measured values are slightly smaller than predicted by $P'_{a_d}(t_p)$, while for large t_p, the measured abort rate in the decision phase is slightly underestimated.

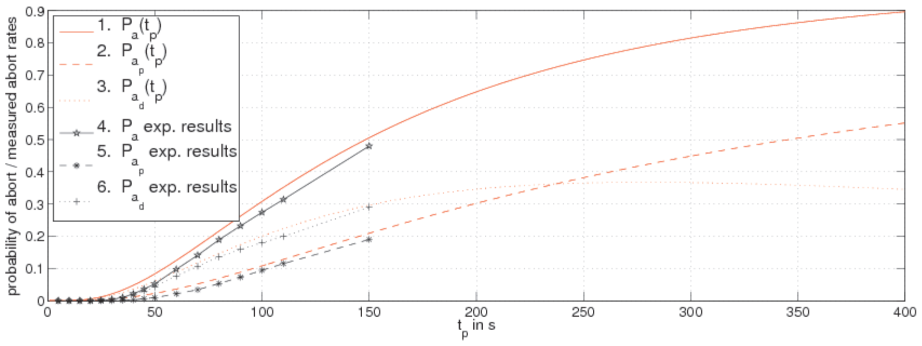
Higher predictions for small t_p are explained by the effect that, for small t_p, the message delay is smaller than for large t_p, because for small t_p participants and the coordinator are most likely still in close vicinity during the decision phase, since transactions are initiated in 1–2 hop distance. The predicted value calculated by $P'_{a_d}(t_p)$ uses δ_m, which is an average value greater than the real message delay for small t_p.



(a) Abort rates in the example MANET scenario with AODV multi-hop routing. Predictions are based on parameters n=3 and δ_m=180 ms.



(b) Abort rates for the example MANET scenario without multi-hop routing. Predictions are based parameters n=3 and δ_m=180 ms.



(c) Abort rates for n=1 and reduced speeds of 1.0–2.0 mps in the example MANET scenario and strict transaction model.

Figure 4.4: Abort probabilities of strict transactions in the example scenario.

Smaller predictions for large t_p are explained by the fact that $P'_{a_d}(t_p)$ considers only a single recovery cycle of communication paths, while in reality multiple failure and recovery cycles can occur. Especially for large t_p, the probability of a communication paths that recovered to fail again increases.

If no multi-hop routing is used neglecting the effect of recovery of failed communication paths is small as shown in Figure 4.4(b) by Curves 2, 3, and 5. Here $P'_{a_d}(t_p)$ as well as $P_{a_d}(t_p)$ provide good approximations of the measured abort rates.

Overall Abort Rate

Figure 4.5(b) depicts the overall abort probability as calculated by Formula (4.20) and experimental values for the example scenario with and without multi-hop routing. Generally, it can be observed that the overall abort rate is high. For example, at t_p=20 s an overall abort rate of 32.7 % is observed with multi-hop routing and of 54.4 % without. A feasible abort rate smaller than 20 % with n=3 is given only for processing phases smaller than 15 s with multi-hop routing. If no multi-hop routing is used, feasible transaction processing in the example scenario is only possible for t_p<13 s.

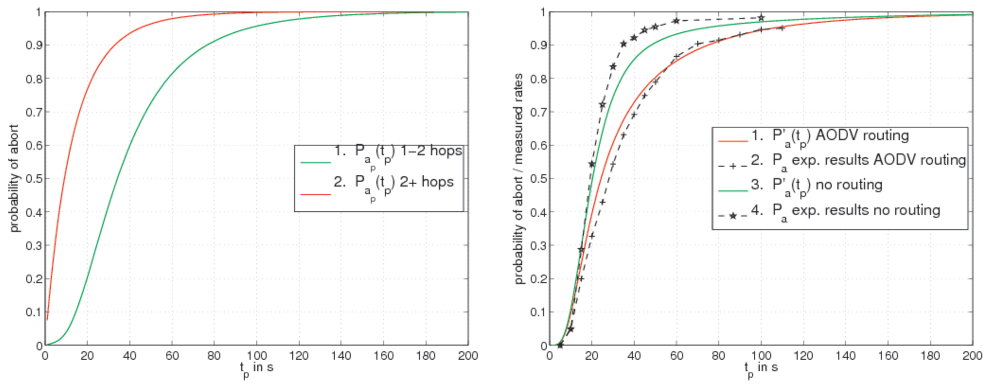
As already stated in Section 4.1.2, the hop count of communication paths at transaction initiation is critical to the abort rates as shown in Figure 4.5(a). While for the results in Figure 4.4(a), 4.4(b), 4.4(c), and 4.5(b) transactions initiated in 1–2 hop distance and therefore log-normal distributions have been assumed for $F_C(t_p)$, Figure 4.5(a) compares the abort rate for transactions initiated in 1–2 (Curve 1) and 2+ (Curve 2) hop distances. For 2+ hop distances the abort probability in the processing phase is already larger than 20 % for transactions with t_p>3 s. Hence, only very short transactions are feasible at all with three participants in this scenario. Given this observation, I will assume 1–2 hop distances for transaction initiation for the example scenario within the remainder of this chapter.

How sensitive the abort probability is to node speeds is shown in Figure 4.4(c). Here, the example MANET scenario is simulated with lower node speeds of 1.0–2.0 mps. At these speeds, the predicted and measured abort rates are significantly smaller than with speeds of 2.0–5.0 mps. Here, even transactions with a duration of up to 80 s show an abort risk smaller than 20 %.

Based on the results presented above, I conclude that calculations of Section 4.4.1 predicting the abort risks of strict transactions in a MANET accurately characterize the dimensions of abort rates to be expected in a MANET scenario. While $P_d(t_p)$ gives very accurate predictions, the major abstraction of the system model to assume an average δ_m and considering a single failure and recovery cycle of communication paths leads to less accuracy of predictions of abort risks in the decision phase. However, for transaction sizes where moderate abort rates are observed, predictions of the overall abort rate are sufficiently accurate.

4.4.2 Abort Probability in the Semantic Model

The semantic transaction model defined in Section 3.2.2 allows for temporarily diverse commit decisions of participants. A participant derives a local preliminary decision on abort or commit that is verified later when the final decision



(a) Comparison of abort rates in the processing phase for 1–2 and 2+ hop initiation distances.

(b) Abort rates with and without multi-hop routing.

Figure 4.5: Overall abort probabilities in the example scenario.

is made by the coordinator. The processing phase of a transaction ends at t'_p, when the coordinator issues the last operation to the last participant. Successful acknowledgment of this operation decides the global transaction. Therefore, the global decision is derived at time $t'_P + 2\delta_m + \Delta_{ex}$, where Δ_{ex} is the time required by the last participant (PA_{last}) to execute its last operation; I denote this point in time as t_u.

In the semantic scheme, the decisive factor for transaction abort is the probability of abort during the processing phase, because the effect that an unrecognized failure that does not recover in time causes an abort decision in the decision phase does not exist. Thus, the abort probability is expected to be lower than in the strict model. Node failures of the coordinator during the processing and decision phase are not considered in the following calculations, because in the semantic model these failures cause an extended uncertainty (ii) situation, which is extensively examined in Section 4.6.

In the semantic model, I denote the probability of abort during the processing phase as $P^*_{a_p}(t'_p)$, which is computed by the complementary probability that neither all nodes in PA_{other} do not cause an abort nor does PA_{last} during $[t_s, t_p]$.

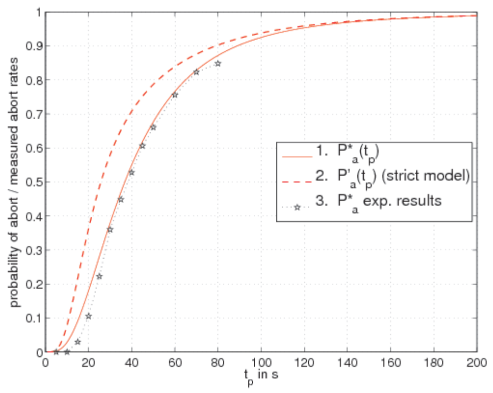
$$P^*_{a_p}(t'_p) = 1 - \left[1 - P_{o>f}(t'_p)\right]^{n-1} \cdot \left[1 - F(t'_p)\right] \tag{4.21}$$

In the interval $[t'_p, t_u]$, only a failure of PA_{last} can cause an abort decision denoted by $P^*_{a_d}(t'_p)$. For this event to happen, the transaction should not be aborted during the processing phase and PA_{last} has to suffer a node or communication failure in the interval $[t'_p, t_u]$.

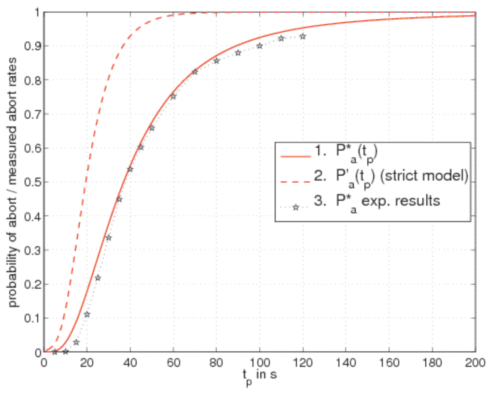
$$P^*{}_{a_d}(t_p) = \left[1 - P_{o>f}(t'_p)\right]^{n-1} \cdot F(t'_p..t_u) \tag{4.22}$$

The overall probability of a transaction being aborted in the semantic model is then given by $P^*_a(t'_p)$.

$$P^*_a(t'_p) = P^*_{a_p}(t'_p) + P^*_{a_d}(t'_p) \tag{4.23}$$



(a) Abort probability of semantic transactions in the example scenario with AODV routing.



(b) Abort probability of semantic transactions in the example scenario without routing.

Figure 4.6: Abort probabilities of semantic transactions in the example MANET scenario.

4.4.2.1 Predictions and Simulation Results

In Figure 4.6, the abort rate predicted by $P^*_a(t_p)$ is compared with measurements obtained from a *ns2* simulation study. In the simulation study, the message exchange of the semantic model was implemented. Similar to the simulation study of the previous section, coordinators and three participants in 1–2 hop distance are randomly chosen. For every participant, t_o was derived by $o(t_p)$ and operation messages are issued to a participant by the coordinator every second until participants reach t_o. If acknowledgments for operations are not received within δ_{to}=1 s, abort is executed. PA_{last} waits for Δ_{ex}=1 s before answering its last operation.

Abort rates are measured for the example scenario with and without multi-hop routing. The simulation results validate the predictions of $P^*_a(t_p)$ in both cases. It can be observed that $P^*_a(t_p)$ predicts slightly higher abort probabilities than observed in experiments for small t_p. Similar to strict transactions, this is explained with smaller δ_m in reality for short transactions, while $P^*_a(t_p)$ uses an average estimate of δ_m for long and short transactions.

In contrast to the strict transaction model, multi-hop routing has little influence on the abort rate, as recovery of communication links does not influence the abort probability in the semantic model.

Another important result is the validation of the presumption that the abort probability in the semantic model is smaller than in the strict model. It showed that this is especially true in the case where no multi-hop routing is used. In this case, transactions with t_p=20 s show an abort risk of 10.5 % in the semantic model, while in the strict model abort has a probability of 54.4 %. If multi-hop routing is used, then the decrease in abort probability is smaller, e.g. 10.4 % in the semantic model compared to 32.7 % in the strict model at t_p=20 s.

4.4.3 Summary - Abort Probabilities

In this section, I presented formulae to approximate the abort rate of transactions in the strict transaction model and in the semantic transaction model. The general observation is that the abort rate is high in the example MANET scenario. Abort rates of similar dimensions should be expected in other MANET scenarios that are of the same class as the example scenario, i.e. scenarios that show similar node movement and network densities. It was shown that only short transactions with values of t_p<15 s in the strict model and t'_p<23 s in the semantic model are feasible if abort rates smaller than 20 % are required.

Generally, the semantic transaction model shows a lower susceptibility for abort than the strict model, e.g. 10.4 % abort risks at t'_p=20 s, while in the strict model 32.7 % is observed.

The abort probability is mainly influenced by node mobility and node speeds as well as by hop distances at transaction initiation. Decreased node speeds drastically reduce the expected abort rate, e.g. node speeds of 1.0–2.0 mps allow for transaction sizes of up to 80 s as shown in Section 4.4.1.3. Choosing participants in 2+ hop distance at transaction initiation leads to exponentially distributed communication failures causing intolerable high abort risks.

Multi-hop routing slightly reduces the number of aborts in the strict model, since recovery of communication paths is more likely with multi-hop routing and more transient failures occur that do not cause abort. Abort during the processing phase is not reduced, since it is assumed that until t_o operations are issued every second to a participant and a missing acknowledgment immediately causes abort.

Although abort rates have been only presented for the example MANET scenario, I argue that this section shows that a primary problem of transaction processing in MANETs are high abort rates. It is important to keep in mind that other failure situations, such as blocking (i) and extended uncertainty (i) considered in the following section, are subsequent problems to abort, since blocking can only occur if a transaction is not aborted before.

4.5 Blocking caused by Participant Failures

In the strict and the semantic transaction models, a participant encounters a blocking or extended uncertainty situation if it suffers a communication failure with the coordinator or disconnects from $\mathcal{A}$ while it is uncertain about the global decision. The probability of this event is strongly influenced by the probability that participants enter their uncertainty window and by the extent of this uncertainty period.

In the following, the probability of blocking (i) and extended uncertainty (i) as defined earlier is analyzed and a probabilistic model is presented and applied to the example MANET scenario.

4.5.1 Probability of Blocking (i)

In 2PC, a participant failure causes a blocking (i) situation if a communication failure with the coordinator or a disconnection of the participant from $\mathcal{A}$ happens while the participant is in the prepared state (see Section 3.6.2). The objective of this section is to develop calculations that predict the probability of a participant to experience such a situation.

The formulae I present in the following have to be interpreted from a single participant's perspective, i.e. they describe the probability of an individual participant to suffer blocking. In the following, I denote this participant by PA, the set of the other $n-1$ participants is called PA_{other}.

The probability of blocking is calculated by considering the probability that PA enters its uncertainty window at time $t_p+\delta_m$ with sending its vote and that a failure occurs before PA leaves the prepared state at time $t_p+\delta_m+\Delta U$. For clarity's sake I define $\tilde{t_p}=t_p+\delta_m$, where $\tilde{t_p}$ is the time PA enters its uncertainty period.

As described in Section 3.6.2, the uncertainty window ΔU in 2PC can be of size $\Delta U_{min}=2\delta_m$ or of size $\Delta U_{max}=2\delta_m+\Delta_{vo}$ when the coordinator awaits a time-out Δ_{vo} for a missing vote.

The most decisive factors in the computation of the blocking risk of PA are the probabilities for entering the uncertainty window and that the uncertainty window is extended to ΔU_{max}. Thus, the probabilities for entering an uncertainty window of size ΔU_{min} or of size ΔU_{max} are required.

If at least one participant of PA_{other} does not reply with a vote message, the uncertainty window enlarges to size ΔU_{max}. In case recovery of communication failures is assumed, the probability of this event is denoted by $P'_{U_{max}}(t_p)$ and given by the probability that at least one node of PA_{other} suffers from a failure that does not recover until t_p, while the other nodes in PA_{other} either do not suffer from a failure or the failure recovers by t_p.

$$\begin{aligned} P'_{U_{max}}(t_p) &= \sum_{i=1}^{n-1}\left[\binom{n-1}{i} P_{o<f_C,nr}(t_p)^i \right. \\ & \left. \cdot \sum_{j=0}^{n-1-i}\binom{n-1-i}{j}\left[1-F(t_p)\right]^j P_{o<f_C,r}(t_p)^{n-1-i-j}\right] \end{aligned} \tag{4.24}$$

In fact a time-out may also happen if no failure of a node in PA_{other} happened until t_p, but in $[t_p, t_p+2\delta_m]$. Here a time-out is caused by a participant, if a general failure happens within $[t_p,\tilde{t_p}]$ or a communication failure occurs within $[\tilde{t_p}, t_p+2\delta_m]$. I do not consider these cases, as the probability of such an event is negligible small. This is due to the fact that the intervals are of size δ_m only. Generally, I will neglect events that occur in intervals of size $\leq \delta_m$ in the following.

$P_{U_{max}}(t_p)$ describes the probability that the uncertainty window of PA is of size ΔU_{max} if recovery of communication failures is not considered. $P_{U_{max}}(t_p)$ solely requires that at least one node in PA_{other} suffers an unrecognized failure given by

$$P_{U_{max}}(t_p)=\left[1-P_{o>f}(t_p)\right]^{n-1}-\left[1-F(t_p)\right]^{n-1} \tag{4.25}$$

The probability no time-out occurs, i.e. that ΔU is of size ΔU_{min}, is given by the probability that all PA_{other} do not suffer a failure or that all failures recover by t_p. This probability is given by $P'_{U_{min}}(t_p)$ if a single recovery cycle of communication failures is assumed.

$$P'_{U_{min}}(t_p) = \sum_{i=0}^{n-1}\left[\binom{n-1}{i} \cdot \left[1 - F(t_p)\right]^i \cdot P_{o<f_C,r}(t_p)^{n-1-i}\right] \tag{4.26}$$

$P'_{U_{min}}(t_p)$ considers all possible situations where i nodes do not suffer from any failure while $n_p - 1 - i$ suffer a communication failure that recovers until t_p. If no recovery of communication failures is considered, $P'_{U_{min}}(t_p)$ simplifies to $P_{U_{min}}(t_p)$.

$$P_{U_{min}}(t_p) = \left[1 - F(t_p)\right]^{n-1} \tag{4.27}$$

The probability that PA suffers from a failure within its window of uncertainty is given by $F(\tilde{t_p}..\tilde{t_p} + \Delta U)$. I denote the probability $F(\tilde{t_p}..\tilde{t_p} + \Delta U_{max})$ by $UF_{max}(t_p)$ and define $UF_{min}(t_p)$ analogously.

Since blocking (i) considers only failures of participants, I am only interested in situations where the coordinator does not suffer from a node failure. This probability is given by $CN_{min}(t_p) = 1 - F_N(t_p + \Delta U_{min})$ and $CN_{max}(t_p) = 1 - F_N(t_p + \Delta U_{max})$ respectively.

The risk of PA of suffering from a blocking situation caused by a participant failure during uncertainty can now be derived as the probability that PA enters an uncertainty window of size ΔU_{min} or ΔU_{max} and that a failure occurs during this period. This probability is computed by $P'_u(t)$ in case recovery of communication is considered.

$$\begin{aligned} P'_u(t_p) &= CN_{max}(t_p) \cdot P'_{U_{max}}(t_p) \cdot UF_{max}(t_p) \\ &\quad + CN_{min}(t_p) \cdot P'_{U_{min}}(t_p) \cdot UF_{min}(t_p) \end{aligned} \tag{4.28}$$

If recovery of paths is not regarded, the risk of PA of suffering blocking is given by $P_u(t_p)$.

$$\begin{aligned} P_u(t_p) &= CN_{max}(t_p) \cdot P_{U_{max}}(t_p) \cdot UF_{max}(t_p) \\ &\quad + CN_{min}(t_p) \cdot P_{U_{min}}(t_p) \cdot UF_{min}(t_p) \end{aligned} \tag{4.29}$$

4.5.1.1 Predictions and Simulation Results

To verify the developed formulae predicting the probability of blocking (i), I did a simulation study for the example MANET scenario using *ns2*. In this study, the message flow of strict transactions with n=3 was simulated.

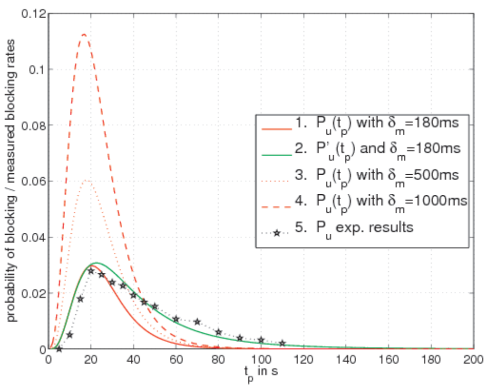
Transaction initiation and processing is similar to the simulations of Section 4.4.1, while here node failures of the coordinator are anticipated to reduce measured blocking situations to cases caused by participant failures. Participants are informed about the global decision of the coordinator at time $\tilde{t_p} + \Delta U$. To derive the rate of blocking (i) situations, the number of participants which entered uncertainty but did not received the global decision from the coordinator is counted.

Figure 4.7 depicts the results of simulations and predictions of $P_u(t_p)$ and $P'_u(t_p)$. While results in Figure 4.7(a) consider multi-hop routing, Figure 4.7(b) depicts results for the example MANET scenario without routing.

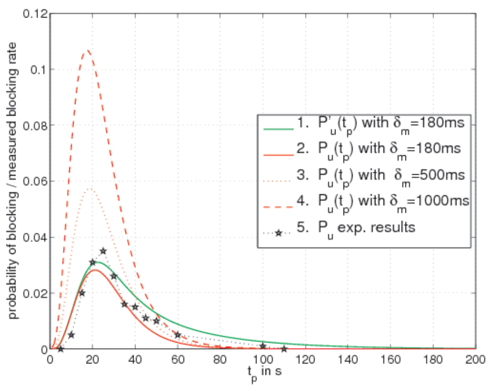
The important observation concerning transaction processing in the exemplary MANET scenario is that the probability of blocking (i) situations are low compared to abort rates. For example, for $t_p < 15$ s, the probability of blocking (i) situations is at maximum of 2 % with and without multi-hop routing.

Additionally it can be observed that the predictions of $P'_u(t_p)$ meet the measured data better for large t_p than $P_u(t_p)$ in case multi-hop routing is used (see Figure 4.7(a) Curve 1, 2, and 5). However, with increasing t_p the probability of multiple path outages and recovery cycles increases and leads to slightly higher blocking rates than predicted by $P'_u(t_p)$.

If no multi-hop routing is used, $P_u(t_p)$ as well as $P'_u(t_p)$ provide accurate approximations of the measured blocking rate (see Figure 4.7(b) Curves 1, 2, and 3). Generally it is shown that multi-hop routing has little influence on the probability of blocking (i), since the blocking rate is nearly the same for the scenario with and without multi-hop routing.



(a) Probability of blocking caused by participant failures in the example MANET scenario with multi-hop routing.



(b) Probability of blocking caused by participant failures in the example MANET scenario without multi-hop routing.

Figure 4.7: Probability of a blocking situations caused by a participant failure in the strict model. Transaction parameters are Δ_{vo}=1 s and n=3. The measured blocking rates are based on 10000 initiated transactions.

Figures 4.7(a) and 4.7(b) also show that the message delay δ_m has a major influence on blocking probability. In case an average message delay of δ_m=1 s is given, the probability of blocking (i) rises to 10.5 %. Such message delays are imaginable if the message load is very high in a MANET or if epidemic message transport is used.

However, the blocking risk examined here can be further reduced if a cooperative recovery scheme is executed. Another possibility would be to integrate an availability check of participants, e.g. an additional message round, before the prepare message is issued to reduce the risk of the uncertainty window to enlarge to ΔU_{max}. In the following, I will calculate the probability of a node to suffer a blocking (i) situation and unsuccessful cooperative recovery.

4.5.1.2 Blocking (i) with Cooperative Recovery

As described in Chapter 3, a blocked participant PA executes a cooperative recovery scheme at time $t_{cr} = t_p + \Delta U_{max} + \delta_m$, since this is the latest time that the global decision can arrive from the coordinator. The success of cooperative recovery depends on the probability that PA can reach one of PA_{other} that is not blocked.

For communication paths between PA and nodes in PA_{other}, I assume the same distribution of path durations as for communication paths between the coordinator and participants. If recovery of paths is disregarded, the probability that PA can reach a node in PA_{other} is given by $F_N(t_{cr})$. If recovery from communication failures between PA and one of PA_{other} is considered to derive more realistic predictions in case a multi-hop routing scheme is available, $P_{c,nr}(t)$ is used. I define $P_{c,nr}(t)$ as the probability of a communication path between PA and one of PA_{other} to break and not to recover until t. $P_{c,nr}(t)$, is calculated as

$$P_{c,nr}(t) = \int\limits_{0}^{t} \int\limits_{t-t_{f_c}}^{\infty} f_C(t_{f_c}) \cdot f_{RC}(t_r)\, dt_r dt_{f_c} \tag{4.30}$$

if t_s=0. Since, $P_{c,nr}(t_{cr})$ does not consider a recovered link to fail again, it converges towards 1.0 for increasing t_{cr}, while $F_N(t_{cr})$ converges to 0. In the real world, the reachability of another participant is obviously not certain or impossible for large t_{cr}. A communication path will most likely experience numerous failure and recovery cycles over time. As already mentioned in Section 4.1.2, multiple failure and recovery cycles can be modeled if failures and path outages are exponentially distributed. In case of log-normal cdf $F_C(t)$, this is not possible. An alternative option is to use P_{path} as defined in Chapter 2 as an approximation for the probability that PA can reach a node of PA_{other}. However, for small t_{cr}, P_{path} underestimates this probability and is better approximated by $F_N(t_{cr})$ and $P_{c,nr}(t_{cr})$, while for larger t_{cr} P_{path} provides a better approximation than $F(t_{cr})$ and $P_{c,nr}(t_{cr})$. In the following, I will present calculations using $F(t_{cr})$, $P_{c,nr}(t_{cr})$, and P_{path} to calculate the probability of cooperative recovery to be unsuccessful.

To derive the probability that PA experiences a blocking situation and cooperative recovery is not successful, the states of nodes in PA_{other} have to be considered. If PA is blocked, a node in PA_{other} can experience one of the three situations: (i) the node never received a prepare message and therefore never entered uncertainty; (ii) the prepare message is received, the participant voted and also received the global decision; or (iii) the participant is blocked like PA.

Participants in PA_{other} that experienced situation (i) and (ii) are potential cooperation partners for PA. Cooperative recovery is not successful if PA cannot reach at least one of these nodes.

To calculate the probability that cooperative recovery is not successful, I distinguish the two cases that ΔU is either of size ΔU_{min} or of size ΔU_{max}. The uncertainty window is of size ΔU_{max} if at least one unrecognized failure occurs with one of PA_{other} that does not recover until t_p. I first consider the case ΔU_{max}. Formula $CR1(t_p)$ therefore enumerates the probabilities for all combinations of events that lead to a situation where j nodes of PA_{other} encounter situation (i) and cannot be reached by PA, while k of PA_{other} experience situation (ii) and are also unreachable for PA. In $CR1(t_p)$, I use three nested sums to enumerate the combined events. The outer sum selects subsets X of PA_{other} that do not encounter a node failure. The second sum selects subsets Y with

j nodes from X that have suffered an unrecognized communication failure that does not recover until t_p. These nodes have experienced situation (i). Hence, for unsuccessful cooperative recovery, PA should not reach any of the j nodes. The innermost sum considers participants that encounter situation (ii). Here, subsets Z with k nodes from Y are selected that have received the global decision but are not reachable by the participant due to a communication failure. $CR1(t_p)$ is then given by

$$\begin{aligned} CR1(t_p) \quad = \quad & \sum_{i=0}^{n-1} \binom{n-1}{i} P_{o<f_n}(t_p)^i \left[1 - F_n(t_p)\right]^{n-1-i} \\ & \cdot \sum_{j=0}^{n-1-i} \binom{n-1-i}{j} P_{o<f_c}(t_p)^j \left[1 - P_{o<f_c}(t_p)\right]^{n-1-i-j} F_C(t_{cr})^j \\ & \cdot \sum_{k=0}^{n-1-i-j} \binom{n-1-i-j}{k} \left[1 - F(t_p..t_p + \Delta U_{max})\right]^k F_C(t_{cr})^k \\ & \cdot F(t_p..t_p + \Delta U_{max})^{n-1-i-j-k} \end{aligned} \tag{4.31}$$

In $CR1(t_p)$, recovery of communication failures is not considered, because $F_C(t_{cr})$ is used as probability of successful communication between PA and a node in PA_{other} for cooperative recovery. Consideration of path recovery leads to $CR1'(t_p)$ by substituting all occurrences of $F_C(t_{cr})$ in Formula (4.31) with $P_{c,nr}(t_{cr})$. Using P_{path} results in $CR1^{pp}(t_p)$ and is derived by replacing all occurrences of $F_C(t_{cr})$ in Formula (4.31) with P_{path}.

If i=j=0, Formulae $CR1'(t_p)$, $CR1^{pp}(t_p)$, and $CR1(t_p)$ consider a case where the uncertainty window is of size ΔU_{min}. The probability of this event has to be subtracted from $CR1(t_p)$, $CR1^{pp}(t_p)$ and $CR1'(t_p)$ respectively, and is given by $CR2(t_p)$, $CR2^{pp}(t_p)$, and $CR2'(t_p)$. $CR2(t_p)$ is given by

$$\begin{aligned} CR2(t_p) \quad = \quad & CR1(t_p) - \Bigg[\left[1 - F_n(t_p)\right]^{n-1} \left[1 - P_{o<f_c}(t_p)\right]^{n-1} \\ & \cdot \sum_{i=0}^{n-1} \Bigg[\binom{n-1}{i} \left[1 - F(t_p..t_p + \Delta U_{max})\right]^i F_C(t_{cr})^i \\ & \cdot F(t_p..t_p + \Delta U_{max})^{n-1-i} \Bigg]\Bigg] \end{aligned} \tag{4.32}$$

while again $CR2'(t_p)$ is derived by replacing all occurrences of $F_C(t_{cr})$ with $P_{c,nr}(t_{cr})$ and $CR2^{pp}$ is given by substituting $F_C(t_{cr})$ in Formula (4.32) with P_{path}.

If all nodes vote, ΔU_{min} is entered. PA then blocks if suffering failure during $[\tilde{t_p}, \tilde{t_p} + \Delta U_{min}]$, described by probability UF_{min} as defined in Section 4.5.1. Cooperative recovery is not successful if all nodes of PA_{other} that received the global decision are not reachable for PA. This probability is denoted by $CR3(t_p)$ in case recovery of communication paths is not considered, by $CR3^{pp}(t_p)$ if P_{path} is used, and by $CR3'(t_p)$ if a single recovery cycle for communication failures is assumed and therefore $P_{c,nr}(t_{cr})$ is used. $CR3(t_p)$, $CR3'(t_p)$, and $CR3^{pp}(t_p)$ are given by

$$\begin{aligned} CR3(t_p) &= [1-F(t_p)]^{n-1} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} [1-F(t_p..t_p+\Delta U_{min})]^i \\ &\cdot F_C(t_{cr})^i F(t_p..t_p+\Delta U_{min})^{n-1-i} \end{aligned} \tag{4.33}$$

$$\begin{aligned} CR3'(t_p) &= [1-P_{o<f,nr}(t_p)]^{n-1} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} [1-F(t_p..t_p+\Delta U_{min})]^i \\ &\cdot P_{c,nr}(t_{cr})^i F(t_p..t_p+\Delta U_{min})^{n-1-i} \end{aligned} \tag{4.34}$$

$$\begin{aligned} CR3^{pp}(t_p) &= [1-F(t_p)]^{n-1} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} [1-F(t_p..t_p+\Delta U_{min})]^i \\ &\cdot P_{path}^i F(t_p..t_p+\Delta U_{min})^{n-1-i} \end{aligned} \tag{4.35}$$

The probability that PA suffers a blocking situation that cannot be recovered immediately is now given by

$$\begin{aligned} P'_{u,cr}(t_p) &= CN_{max}(t_p) \cdot UF_{max}(t_p) \cdot CR2'(t_p) \\ &+ CN_{min}(t_p) \cdot UF_{min}(t_p) \cdot CR3'(t_p) \end{aligned} \tag{4.36}$$

if a single recovery cycle of communication failures is assumed. If no recovery from path breaks is considered, $P_{u,cr}(t_p)$ describes the risk of PA to suffer blocking (i) and unsuccessful cooperative recovery.

$$\begin{aligned} P_{u,cr}(t_p) &= CN_{max}(t_p) \cdot UF_{max}(t_p) \cdot CR2(t_p) \\ &+ CN_{max}(t_p) \cdot UF_{min}(t_p) \cdot CR3(t_p) \end{aligned} \tag{4.37}$$

If P_{path} is used to this probability is given by $P^{pp}_{u,cr}(t_p)$

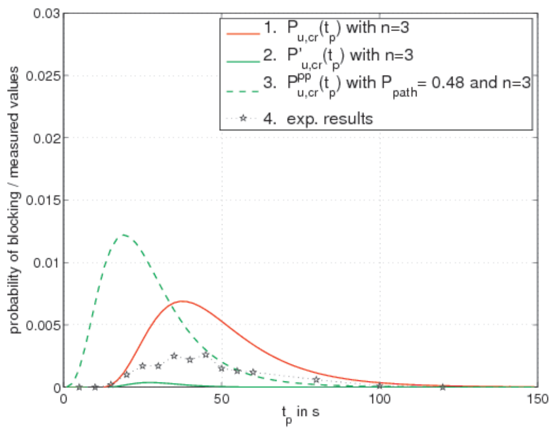
$$\begin{aligned} P^{pp}_{u,cr}(t_p) &= CN_{max}(t_p) \cdot UF_{max}(t_p) \cdot CR2^{pp}(t_p) \\ &+ CN_{max}(t_p) \cdot UF_{min}(t_p) \cdot CR3^{pp}(t_p) \end{aligned} \tag{4.38}$$

While $P_{u,cr}(t_p)$, $P^{pp}_{u,cr}(t_p)$, and $P'_{u,cr}(t_p)$ calculate the probability that the first request round of cooperative recovery is not successful, consecutively recovery rounds are possibly successful. $P_{u,cr}(t_p)$, $P^{pp}_{u,cr}(t_p)$ and $P'_{u,cr}(t_p)$ are the relevant probabilities here, because only a participant that experiences blocking and cannot recover immediately must retry recovery for an indefinite period which is the semantic of a blocking situation.

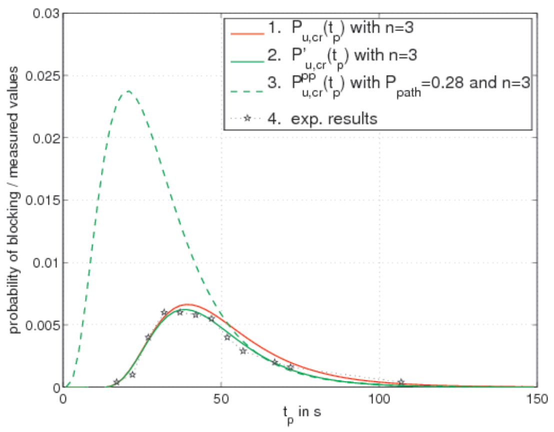
4.5.1.3 Predictions and Simulation Results

Figure 4.8 shows probabilities derived by $P'_{u,cr}(t_p)$, $P^{pp}_{u,cr}(t_p)$, and $P_{u,cr}(t_p)$ for the example MANET scenario. Figure 4.8(a) presents results if multi-hop routing is used, while Figure 4.8(b) considers single-hop routing only. In Figures 4.8(c) and 4.8(d) results for transactions with two participants with and without multi-hop routing are presented.

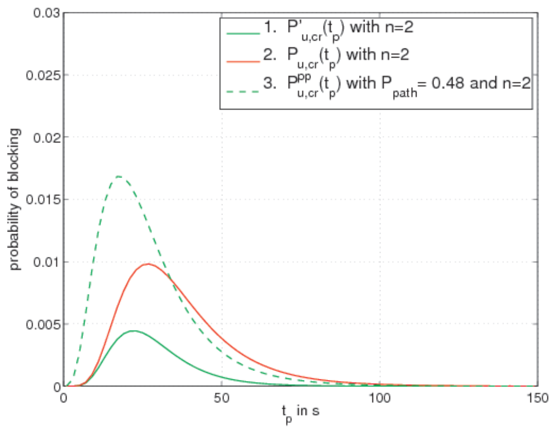
The important observation is that the risk of PA suffering blocking (i) is significantly reduced by cooperative recovery. For example, without cooperative recovery and with AODV routing, the probability of blocking is 2 % at



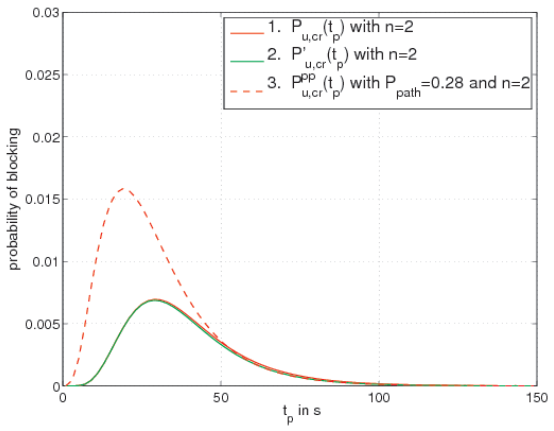
(a) Probability of blocking (i) with cooperative recovery and multi-hop routing.



(b) Probability of blocking (i) with cooperative recovery without multi-hop routing.



(c) Probability of blocking (i) with cooperative recovery, multi-hop routing, and n=2.



(d) Probability of blocking (i) with cooperative recovery, no routing, and n=2.

Figure 4.8: Risk of PA of suffering from a blocking (i) situation and unsuccessful cooperative recovery in the example MANET scenario with transaction parameters Δ_{vo}=1 s, and δ_m=180 ms.

t_p=15 s with n=3 (see Section 4.5.1.1). With cooperative recovery this probability decreases to 0.22 %. In the example scenario, this probability reaches a considerable value of 0.7 % only for transactions with t_p>30 s. Note that for the example MANET scenario, I assumed that only transactions with t_p<15 s are feasible, here blocking (i) risks are negligible in the example MANET scenario with cooperative recovery.

The majority of blocking (i) situations compensated for by cooperative recovery benefit from nodes in PA_{other} that did not receive the prepare message and hence suffered from an undetected communication failure with the coordinator, while the situation where a partner for cooperative recovery is found that received the global decision is rare.

Simulation results of a *ns2* simulation study for the example MANET scenario show that the proposed calculation model predicts the real world blocking rates accurately, especially if no multi-hop routing is used, i.e. if recovery of communication failures takes long, as shown in Figure 4.8(b). If multi-hop routing is used, $P_{u,cr}(t_p)$ provides an upper bound, while $P'_{u,cr}(t_p)$ gives a lower bound of blocking (i) risks. $P^{pp}_{u,cr}(t_p)$ calculates accurate approximations for large t_p as shown in Figure 4.8(a). This is explained by the fact that $P_{u,cr}(t_P)$ does not consider any recovery of failed communication paths, while $P^{'}_{u,cr}(t_p)$ considers exactly one recovery cycle and assumes no subsequent path failures. In contrast, $P^{pp}_{u,cr}(t_p)$ considers the constant P_{path} of $\mathcal{A}$ and therefore meets the real values exactly for large t_p (here, for $t_p < 50s$). For the transaction sizes where transaction processing is feasible in the example scenario, predictions of $P_{u,cr}(t_P)$ and $P^{'}_{u,cr}(t_p)$ are close together and therefore also meet the simulation results accurately. However, in contrast to strict transaction processing without cooperative recovery, using a multi-hop routing scheme significantly reduces blocking risks compared to the situation where no multi-hop routing is used. For example, with multi-hop routing the maximum blocking (i) probability is 0.26 %, without a maximum of 0.58 % is measured.

Experimental values are derived from a simulation similar to the simulation presented in Section 4.5.1.1, while a cooperative recovery scheme is initiated by blocked participants at time t_{cr}. Presented measurements are derived by including all recovery attempts successfully within the first message round of cooperative recovery.

Success of cooperative recovery is highly dependent on the number of participants, as shown in Figures 4.8(c) and 4.8(d). Figure 4.8(c) shows that the probability of blocking, i.e. of unsuccessful cooperative recovery increases significantly for n=2 compared to n=3. E.g. with n=2 the probability of blocking is 0.48 %, compared to 0.22 % with n=3 if no multi-hop routing is used (see Figure 4.8(c)). The reduction of blocking (i) risks with n=3 is also supported by the effect that with n=3 a higher abort rate in the processing phase is observed, compared to the case with n=2, which leads to fewer transactions actually entering the commit phase and less blocking can happen.

Given the observations above, it can be concluded that cooperative recovery is a highly efficient scheme to compensate for blocking (i) situations. Since, cooperative recovery does not increase the message or time complexity of 2PC in the failure-free case, it is the number one choice to decrease blocking risks in the strict scheme.

4.5.2 Probability of Extended Uncertainty (i)

In the semantic model, the processing phase of a transaction is given by the interval $[t_s, t'_p]$, while the decision phase is defined by $[t'_p, t_u + \delta_m]$ with $t_u = t'_p + \Delta_{ex} + \delta_m$. t'_p is defined as the time the coordinator sends the last operation to the last participant denoted by PA_{last}. A participant PA enters its uncertainty window at time t_o and leaves uncertainty at $t_u + \delta_m$.

Analogous to blocking (i) in the strict model, an extended uncertainty (i) situation is defined in the semantic model as any situation where the global decision is made by the coordinator but cannot be transferred to PA due to a communication failure or because PA has disconnected from $\mathcal{A}$. In contrast to the strict model, where blocking can only occur in the decision phase, extended uncertainty (i) can already occur in the processing phase. In the following, I will analyze the probability of extended uncertainty (i) in the processing and decision phase of a semantic transaction. I first consider the processing phase.

In the interval $[t_s, t'_p]$, the coordinator decides on abort if detecting a failure. PA experiences an extended uncertainty (i) situation if the coordinator recognizes a failure with one of PA_{other} or with PA_{last}, while PA has already entered its uncertainty window and cannot receive the global abort decision from the coordinator. The abort decision is not received if PA experiences a communication failure with the coordinator that does not recover in time, or if PA suffers a node failure.

While in the strict model all participants enter uncertainty at time $\tilde{t_p}$, the time a participant enters uncertainty in the semantic model varies for every participant and is given by t_o. Hence, for every point in time t_f in $[t_s, t'_p]$, the situation must be considered that a recognized failure occurs causing abort of the transaction (I call this situation A) and that PA is in its uncertainty window and cannot receive the global abort decision (I call this event B). The probability of event A is given by $P_{o<f_c,nr}(t_f)$ and $P_{o<f_c}(t_f)$ respectively, while the probability of situation B is computed by $PB(t_f, t'_p)$.

$$PB(t_f, t'_p) = 1 - \left[1 - f(t_f) \cdot O(t_f..t'_p)\right]^{n-2} \cdot \left[1 - f(t_f)\right] \tag{4.39}$$

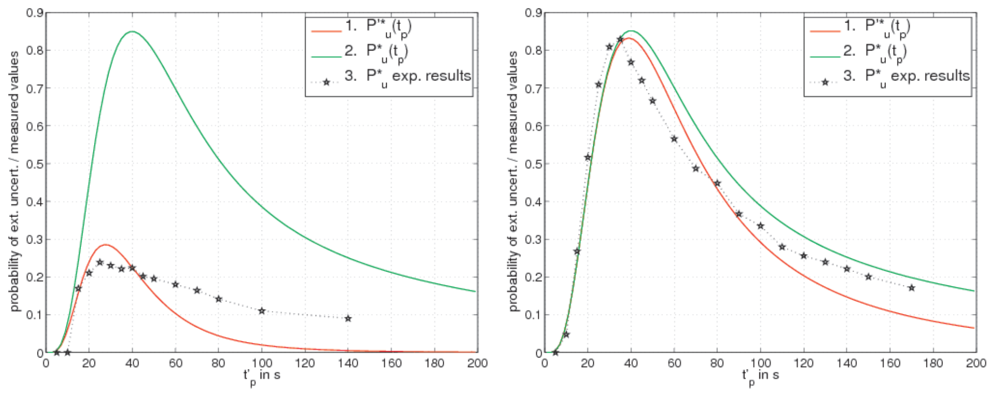
The probability that situation A and B happens in the interval $[t_s, t'_p]$ is the probability of PA to experience extended uncertainty (i) in this interval. I call this probability $Pu_1(t'_p)$ and it is given by

$$Pu_1(t'_p) = \int_0^{t'_p} PB(t_f, t_p) \cdot P_{o<f_c}(t_f)\, dt_f \tag{4.40}$$

if recovery of communication is not considered. If the reachability of the coordinator is calculated by $P_{o<f_c,nr}(t)$, $P'u_1(t'_p)$ is derived.

$$P^{'}u_1(t'_p) = \int_0^{t'_p} PB(t_f, t_p) \cdot P_{o<f_c,nr}(t_f)\, dt_f \tag{4.41}$$

An extended uncertainty situation of PA is caused in the interval $[t'_p, t_u + \delta_m]$ if the global decision is made by the coordinator at t_u, but PA cannot receive this decision because of a communication failure or disconnection from $\mathcal{A}$. The probability for PA to experience an extended uncertainty situation if the trans-



(a) Probability of extended uncertainty in the example MANET scenario with multi-hop routing.

(b) Probability of extended uncertainty caused by a participant failure without routing.

Figure 4.9: Probability of extended uncertainty (i) for semantic transactions, with n=3, δ_m=180 ms and Δ_{ex}=1 s.

action enters the decision phase is given by $Pu_2(t'_p)$ if recovery of path breaks is not considered.

$$Pu_2(t'_p) = P_{o<f}(t_u) \cdot \left(1 - P_{o>f}(t'_p)^{n-2} \cdot [1 - F(t'_p)]\right) \tag{4.42}$$

and by $P'u_2(t'_p)$ if a single recovery cycle of communication failures is considered.

$$P'u_2(t'_p) = P_{o<f_c,nr}(t_u) \cdot \left(1 - P_{o>f}(t'_p)^{n-2} \cdot [1 - F(t'_p)]\right) \tag{4.43}$$

The probability that PA experiences an extended uncertainty (i) situation in the processing phase or in the decision phase is now given by $P'^*_u(t'_p)$ if recovery of path breaks is considered:

$$P'^*_u(t'_p) = P'u_1(t'_p) + P'u_2(t'_p) \tag{4.44}$$

and by $P^*_u(t'_p)$ if recovery of communication paths is neglected.

$$P^*_u(t'_p) = Pu_1(t'_p) + Pu_2(t'_p) \tag{4.45}$$

4.5.2.1 Predictions and Simulation Results

Figure 4.9 depicts the probability of extended uncertainty (i) for the example MANET scenario calculated by $P'^*_u(t'_p)$ and $P^*_u(t'_p)$. Predictions are compared to measurements obtained from a *ns2* simulation study.

ns2 was used in order to simulate the message flow of transactions in the semantic transaction model. The rate of extended uncertainty (i) situations was measured by counting all the participants that have entered uncertainty and have not subsequently received the global decision until $t_u + \delta_m + \delta_{to}$.

The hypothesis that the semantic model shows a higher susceptibility to extended uncertainty (i) situations than the strict model does to blocking (i) is clearly confirmed by analytical predictions as well as by simulation results as shown by Figure 4.9(a) and 4.9(b).

Figure 4.9(a) shows that the probability of extended uncertainty (i) in the example MANET scenario with multi-hop routing is considerably higher, with 16.94 % for t'_p=15 s compared to 2 % for blocking (i) situations in the strict model.

If no multi-hop routing is used, the probability for extended uncertainty (i) increases to 26.77 % at t'_p=15 s. Hence, the effect of short path outages with multi-hop routing reduces the probability of extended uncertainty (i) drastically. This also shows the importance of considering recovery of paths if multi-hop routing is used. Neglecting path recovery leads to predictions that drastically overestimates the real risks as shown in Figure 4.9(a) by Curve 2 and 3.

The mistake in $P_u^{'*}(t'_p)$ (shown by Curve 1 in Figure 4.9(a)) of considering only one recovery cycle is reflected by the effect that, for large t_p, the probability of extended uncertainty (i) is underestimated by $P_u^{'*}(t'_p)$. However, this has only a small impact, because at large t_p where the simplified assumption of the calculation model becomes relevant, transaction processing is not feasible because of high abort rates, as described in Section 4.4.2. Recall that only transactions with t_p<20 s are considered feasible in the example MANET scenario and semantic transaction model.

4.5.2.2 Extended Uncertainty (i) with Cooperative Recovery

In the semantic model, cooperative recovery is started at $t^*_{cr} = t_u + \delta_m$, as this is the latest point in time a participant can expect the global decision. The probability that PA suffers an extended uncertainty (i) situation that cannot be compensated for immediately by cooperative recovery at t^*_{cr} depends on the probability that PA suffers extended uncertainty and that neither a node of PA_{other} nor PA_{last} is certain and reachable for PA at t^*_{cr}.

I first consider recovery with PA_{last} separately. A node failure of PA_{last} within $[t_s, t'_p]$ always causes an abort of the global transaction and also induces PA_{last} unavailability for cooperative recovery at t^*_{cr}. A communication failure of PA_{last} with the coordinator within the processing phase $[t_s, t'_p]$ also causes an abort of the global transaction, but PA_{last} is a potential cooperation partner for PA in this case, as PA_{last} is always certain within $[t_s, t'_p]$. Now, I assume that such a communication failure occurs at time t_f during $[t_s, t'_p]$. PA_{last} is only available for cooperative recovery with PA if it does not suffer node failure within $[t_f, t^*_{cr}]$.

The probability that at time t_f the transaction is aborted by a communication failure between the coordinator and PA_{last} (given by $f_C(t_f)$), while PA_{last} *does not* suffer a node failure until recovery of PA is started (given by $\left[1 - F_N(t_f..t^*_{cr})\right]$) *and* PA is uncertain at time t_f (given by $O(0..t_f)$) *and* PA cannot reach PA_{last} at t^*_{cr} is calculated by $CR1^{*'}(t_f)$ if recovery of path breaks is neglected.

$$\begin{aligned} CR1^{*'}(t_f) \quad = \quad & O(0..t_f) \cdot P_{o<f_c,nr}(t_f) \\ & \cdot \Big[f_N(t_f) + f_c(t_f) \cdot \left[1 - F_N(t_f..t^*_{cr})\right] \cdot P_{c,nr}(t^*_{cr}) \Big] \end{aligned} \tag{4.46}$$

If recovery of communication paths is not considered, i.e. the reachability of a recovery partner is given by $F_C(t_{cr})$, $CR1^*(t_f)$ is derived.

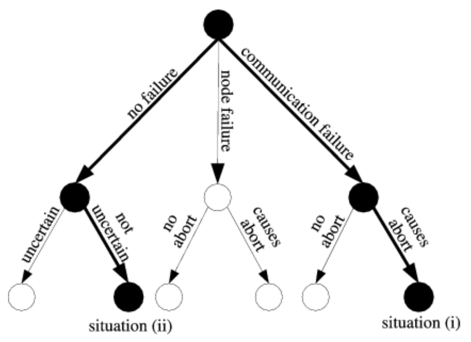


Figure 4.10: Decision tree of nodes in PA_{other}.

$$\begin{aligned} CR1^*(t_f) &= O(0..t_f) \cdot P_{o<f_c}(t_f) \\ &\quad \cdot \Big[f_N(t_f) + f_c(t_f) \cdot [1 - F_N(t_f..t^*_{cr})] \cdot F_C(t^*_{cr}) \Big] \end{aligned} \tag{4.47}$$

If the reachability of a participant of PA_{other} is described by P_{path}, $CR1^*$ results in $CR1^{*pp}(t_f)$.

$$\begin{aligned} CR1^{*pp}(t_f) &= O(0..t_f) \cdot P_{o<f_c}(t_f) \\ &\quad \cdot \Big[f_n(t_f) + f_c(t_f) \cdot [1 - F_N(t_f..t^*_{cr})] \cdot P_{path} \Big] \end{aligned} \tag{4.48}$$

Line (2) of $CR1^{*'}(t_f)$, $CR1^*(t_f)$, and $CR^{*pp}(t_f)$ considers the probability that PA encounters a communication failure with PA_{last} that prevents communication at time t^*_{cr} (given by $P_{c,nr}(t^*_{cr})$, $F_C(t^*_{cr})$, and P_{path} respectively). The factors $P_{o<f_c}(t_f)$ in Formulae (4.47) and (4.48) and $P_{o<f_c,nr}(t_f)$ in Formula (4.46), describe the probability that the coordinator cannot reach PA at time t_f, when the global transaction is aborted.

To estimate the probability that a node of PA_{other} is a potential partner for cooperative recovery, for every point in time within the processing phase $[t_s, t'_p]$, the state of each participant has to be be considered. A participant can remove uncertainty from PA if it is in one of the two following situations: (i) if it has not suffered failure and has not received its last operation; or (ii) if it encounters a communication failure with the coordinator that leads to abort of the transaction, i.e. the communication failure with the coordinator happens before t_o. Participants that experienced a node failure or are uncertain cannot remove uncertainty from PA.

Figure 4.10 shows the decision tree with paths leading to situations (i) and (ii). The idea of the following calculation is to sum up the probabilities for a subset of PA_{other} to encounter situation (i) *or* situation (ii) *and not* to be reachable for PA at t^*_{cr} due to a communication failure.

To select the relevant probabilities for nodes in PA_{other} to encounter situation (i) or (ii), I use nested sums according to the following considerations: for a point in time t_f, a set A from PA_{other} with i nodes is selected, which encounter a node failure that causes abort, while the other $n-2-i$ nodes are divided in the set B of j nodes, which experience a node failure that does not lead to the abort of the global transaction, and a set C of $n-2-i-j$ nodes, which does not experience a node failure. Set C is further decomposed into sets D with k nodes, which encounter a communication failure with the coordinator that causes abort, and E with $n-2-i-j-k$ nodes that either encounter a communication failure that does not lead to transaction abort or do not suffer from a communication failure with the coordinator at t_f. All nodes in D experience situation (i). Set E is further decomposed into sets F and G, where F contains l nodes that experienced a communication failure that does not cause abort, while G contains $n-2-i-j-k-l$ nodes, which do not experience a communication failure at t_f. Set G now contains m nodes, which have not experienced any failure and are not uncertain as they have not received their last operation at t_f, while $n-2-i-j-k-l-m$ nodes of G are uncertain. Hence, the m nodes of G have experienced situation (ii). Cooperative recovery is not successful if PA cannot reach nodes in D and m nodes of G. From these considerations, I derive Formula $CR2^{*'}(t'_p)$:

$$\begin{aligned}
CR2^{*'}(t'_p) \quad = \quad & \int_0^{t'_p} pcr_1(t_f) \cdot \sum_{i=0}^{n-2} \binom{n-2}{i} P'_{o>f_n}(t_f)^i \\
& \cdot \sum_{j=0}^{n-2-i} \binom{n-2-i}{j} P'_{o<f_n}(t_f)^j \cdot [1 - f_N(t_f)]^a \\
& \cdot \sum_{k=0}^{a} \binom{a}{k} \left[P'_{o>f_c}(t'_f) \cdot [1 - F_N(t_f..t_u)] \cdot P_{c,nr}(t^*_{cr})\right]^k \\
& \cdot \sum_{l=0}^{b} \binom{b}{l} P'_{o<f_c}(t_f)^l \cdot [1 - f_c(t_f)]^c \\
& \cdot \sum_{m=0}^{c} \binom{c}{m} \left[O(t_f..t'_p) \cdot [1 - F_N(t_f..t^*_{cr})] \cdot P_{c,nr}(t^*_{cr})\right]^m \\
& \cdot O(0..t_f)^d \, dt_f \qquad (4.49)
\end{aligned}$$

with the upper bounds of summations:

$$\begin{aligned}
a &= n-2-i-j \\
b &= n-2-i-j-k \\
c &= n-2-i-j-k-l \\
d &= n-2-i-j-k-l-m
\end{aligned}$$

The variant of $CR2^{*'}(t'_p)$ that does not consider recovery of communication paths is called $CR2^{*}(t'_p)$ and is derived by substituting all occurrences of $P_{f_c,nr}(t^*_{cr})$ with $F_C(t^*_{cr})$. If reachability of a recovery partner is approximated by P_{path}, $CR2^{*pp}(t'_p)$ is derived by substituting all occurrences of $P_{f_c,nr}(t^*_{cr})$ with P_{path}. Formula $CR2^{*'}(t'_p)$ includes a path where the transaction is not aborted in $[t_s, t'_p]$, which happens for i=k=0. I denote this case as $CR3^{*'}(t'_p)$.

$$\begin{aligned} CR3^{*'}(t'_p) \quad = \quad & \int_0^{t'_p} pcr_1(t_f) \cdot \sum_{i=0}^{n-2} \binom{n-2}{i} P'_{o<f_n}(t_f)^i \cdot [1 - f_N(t_f)]^{n-2-i} \\ & \cdot \sum_{j=0}^{n-2-i} \binom{n-2-i}{j} P'_{o<f_c}(t_f)^j \cdot [1 - f_c(t_f)]^a \\ & \cdot \sum_{k=0}^{a} \binom{a}{k} (O(t_f..t'_p) \cdot [1 - F_N(t_f..t_u)] \cdot P_{f_c,nr}(t^*_{cr})]^k \\ & \cdot O(0..t_f)^d \, dt_f \end{aligned} \tag{4.50}$$

Again, variant $CR3^*(t'_p)$ is derived by substituting $P_{c,nr}(t^*_{cr})$ with $F_C(t^*_{cr})$. $CR3^{*pp}$ is derived similarly by using P_{path} instead of $P_{c,nr}(t^*_{cr})$.

Now, the probability that PA suffers from an extended uncertainty situation that cannot be recovered immediately at time t^*_{cr} is given by $P^{*'}_{u,cr}(t'_p)$ if a single recovery cycle is assumed.

$$P^{*'}_{u,cr}(t'_p) = CR2^{*'}(t'_p) - CR3^{*'}(t'_p) \tag{4.51}$$

By $P^*_{u,cr}(t'_p)$ if recovery of path breaks is neglected.

$$P^*_{u,cr}(t'_p) = CR2^*(t'_p) - CR3^*(t'_p) \tag{4.52}$$

And by $P^{*pp}_{u,cr}(t'_p)$ if P_{path} is used to approximate successful communication at t^*_{cr}.

$$P^{*pp}_{u,cr}(t'_p) = CR2^{*pp}(t'_p) - CR3^{*pp}(t'_p) \tag{4.53}$$

4.5.2.3 Predictions Extended Uncertainty (i)

Figure 4.11 depicts probabilities calculated by $P^*_{u,cr}(t'_p)$, $P^{*'}_{u,cr}(t'_p)$, and $P^{*pp}_{u,cr}(t'_p)$ for the example MANET scenario and transactions with 2–3 participants.

The major result is that, similar to the strict case, cooperative recovery significantly compensates for extended uncertainty (i) situations. For example, Figure 4.9(a) shows a probability of extended uncertainty (i) to occur of 21 % for a transaction with t'_p=20 s, n=3, and without cooperative recovery, which reduces to maximum 1.7 % if cooperative recovery is used as shown in Figure 4.11(a).

If no multi-hop routing is used, the probability for blocking reduces from 51 % to 1.19 % at t'_p=20 s with three participants as shown in Figures 4.9(b) and 4.11(b).

If multi-hop routing is used, recovery of communication paths has a major influence as shown in Figure 4.11(a). In this case, $P^{*'}_{u,cr}(t'_p)$ underestimates the real uncertainty rate, because only one failure and recovery cycle of communication paths is considered by $P^{*'}_{u,cr}(t'_p)$. Hence, once recovered, a link is assumed to remain operational forever. In reality, this is obviously not true, and therefore the observed rate of uncertainty probability is upper bounded by $P^*_{u,cr}(t'_p)$ and lower bounded by $P^{*'}_{u,cr}(t'_p)$. $P^{*pp}_{u,cr}(t'_p)$ predicts uncertainty rates considering the constant path probability for communication paths and derives values lying between $P^*_{u,cr}(t'_p)$ and $P^{*'}_{u,cr}(t'_p)$ for large t'_p. However, for t'_p<20 s, the values of all three predictions are close together and should provide a good approximation of extended uncertainty (i) rates in the example scenario.

Similar to the strict case, the number of participants is a decisive factor. This can be observed in Figure 4.11(c) and 4.11(d) showing the uncertainty probabilities for the example scenario and transactions with n=2. Here, the rate of extended uncertainty situations increases from 1.9 % with n=3 to 5 % with n=2 in the multi-hop scenario with t'_p=20 s.

4.5.3 Summary - Blocking caused by Participants

In this section, I have presented a calculation model to predict the blocking and uncertainty risks due to participant failures with and without cooperative recovery. Results for the example MANET scenario show that the risk of PA suffering blocking (i) is very low if multi-hop routing and cooperative recovery is used. In fact, without cooperative recovery, the risk is below 4 % for t_p<15 s, while cooperative recovery reduces this risk below 1 %.

In the semantic model, the probability of extended uncertainty caused by a failure of PA is considerably higher compared to the strict model, while cooperative recovery is also very efficient. Generally, the probability of blocking or extended uncertainty with cooperative recovery is strongly influenced by the number of participants involved in a transaction. This observation is a primary motivation for the design of the SLS presented later in Chapter 5.

Transactions with just two participants show the highest risk of blocking in the strict model for the example scenario, because cooperative recovery with n=2 is less effective and the probability that such a transaction reaches the decision phase in the strict model is considerably higher than with n>2.

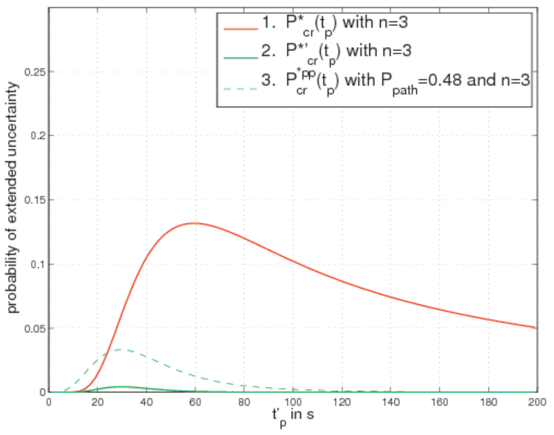
In the semantic model, a transaction with two participants shows the highest risk for extended uncertainty (i) measured so far in the example scenario. Hence, semantic transactions with few participants will most likely require recovery schemes in addition to cooperative recovery to compensate for extended uncertainty.

By comparing most analytical predictions to simulation results, I showed that abstractions made within the probabilistic models presented are feasible and hence, that the models can be used to predict blocking and extended uncertainty caused by participants failures realistically.

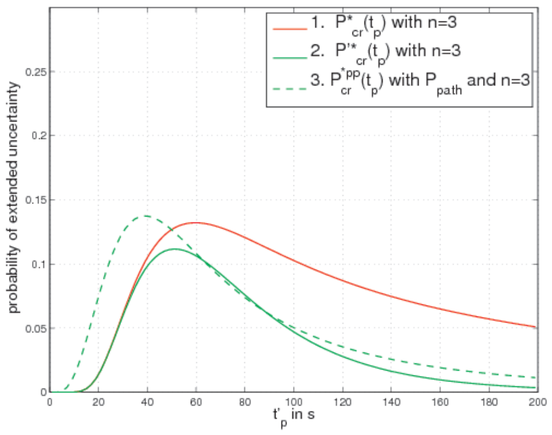
4.6 Blocking Caused by Coordinator Failures

While the previous section was concerned with blocking situations caused by participant failures, this section examines the probability of blocking caused by a node failure of the transaction coordinator. In the literature, this situation is assumed to be the more severe case, because the failure of the central coordination entity may cause blocking of multiple participants, while failure of a participant results in blocking of that participant only.

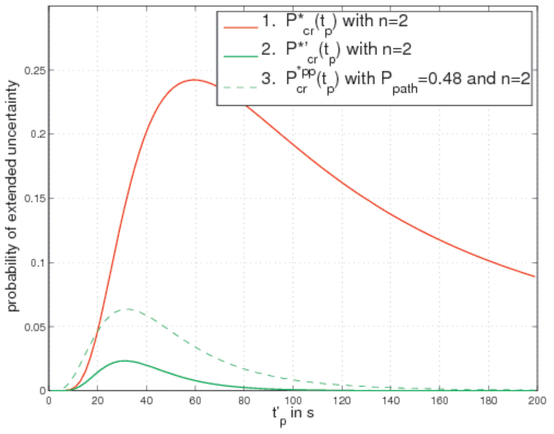
I demonstrate in the following that the probability of blocking caused by a node failure of the coordinator, i.e. the probability of blocking (ii) and extended uncertainty (ii) is very low for the example MANET scenario. The calculations presented in the following omit recovery of path breaks and therefore provide an upper bound for blocking (ii) and extended uncertainty (ii) probabilities. Like in Section 4.5, I will first consider the strict transaction model and then the semantic model.



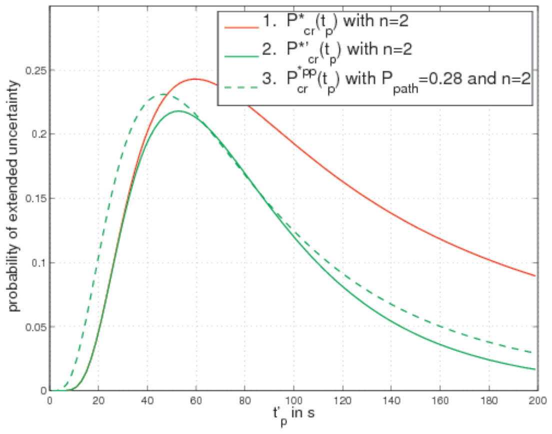
(a) Probability of extended uncertainty (i) with cooperative recovery and multi-hop routing.



(b) Probability of extended uncertainty (i) with cooperative recovery and without multi-hop routing.



(c) Probability of extended uncertainty (i) with n=2 and multi-hop routing.



(d) Probability of extended uncertainty (i) with n=2 and without multi-hop routing.

Figure 4.11: Probability of extended uncertainty (i) in the example MANET scenario with cooperative recovery.

4.6.1 Probability of Blocking (ii)

In the strict model, the most decisive factors in the computation of the blocking (ii) risk of a participant are the probabilities for entering the decision phase and that ΔU is extended to ΔU_{max}. The probabilities for PA to enter an uncertainty window of ΔU_{min} or of ΔU_{max} are given by the probability that the coordinator awaits time-out Δ_{vo} or not. I already calculated these probabilities as $P_{U_{min}}(t_p)$ and $P_{U_{max}}(t_p)$ in Section 4.5.1 in Formulae (4.27) and (4.25).

In fact, a time-out Δ_{vo} may also happen if no failure of a node in PA_{other} happens until t_p, but in the interval $[t_p, t_p + 2\delta_m]$. Here, a time-out is caused by a participant if a general failure happens within $[t_p, t_p + \delta_m]$ or a communication failure occurs within $[t_p + \delta_m, t_p + 2\delta_m]$. I do not consider these cases here, because the probability of such an event is negligible, as the intervals are of size δ_m only. Generally, I neglect events that occur in intervals smaller than $2\delta_m$ in the following.

I denote the probability of a node failure of the coordinator within interval $[t_p, t_p + \Delta U_{min}]$ by $CF_{U_{min}}(t_p)$, which is given by $F_N(t_p..t_p + \Delta U_{min})$. $CF_{U_{max}}(t_p)$ is defined analogously. The probability that PA does not encounter any failure until $t_p + \Delta U_{min}$, given by $1 - F(t_p + \Delta U_{min})$, is denoted by $PA_{U_{min}}(t_p)$. Analogously I define $PA_{U_{max}}(t_p)$.

The probability that PA enters an uncertainty window of size ΔU_{max} or of ΔU_{min} and, while uncertain about the global decision, the coordinator suffers from a node failure and thus PA is blocked is now given by $P_v(t_p)$.

$$\begin{aligned} P_v(t_p) &= PA_{Umax}(t_p) \cdot CF_{Umax}(t_p) \cdot P_{Umax}(t_p) \\ &\quad + PA_{Umin}(t_p) \cdot CF_{Umin}(t_p) \cdot P_{Umin}(t_p) \end{aligned} \tag{4.54}$$

Predictions of $P_v(t_p)$ for the example MANET scenario are presented at the end of this subsection.

Blocking (ii) with Cooperative Recovery

If PA suffers blocking (ii), a cooperative recovery scheme is initiated.

The success of this scheme is given by the probability that PA can reach at least one node in PA_{other} that is not blocked. Recall that here I only consider the case that a coordinator failure during ΔU leads to blocking. Now, if PA is blocked, all nodes of PA_{other} that also received the prepare message are blocked too. Only nodes that encountered an unrecognized communication failure remain unblocked and thus are potential cooperative partners for PA. Such a partner is reachable for PA if it is still alive and no communication failure between them has happened within $[t_s, t_p + \Delta U + \delta_m]$. As above, I distinguish the two cases that ΔU is either of length ΔU_{min} or ΔU_{max}. First, I consider case ΔU_{max}.

Again, I consider the case that at least one unrecognized failure leads to ΔU_{max}. In Formula (4.55), I investigate probabilities for all combinations of events that lead to at least one unrecognized failure and additionally let j nodes of PA_{other} remain unblocked. I use two nested sums that enumerate combined events. The outer sum selects subsets X of nodes of PA_{other} that do not encounter a node failure. The inner sum then selects from X the subsets Y of nodes that have suffered from an unrecognized communication failure by t_p. All j nodes in subsets Y are unblocked and potential recovery partners for PA.

If all j nodes are unreachable, because of a communication failure with PA, cooperative recovery is unsuccessful.

$$\begin{aligned} CR1(t_p) &= \sum_{i=0}^{n-1}\Bigg[\binom{n-1}{i}\cdot P_{o<f_n}(t_p)^i\cdot\big[1-F_N(t_p)\big]^{n-1-i} \\ &\quad\cdot\sum_{j=0}^{n-1-i}\binom{n-1-i}{j}\cdot P_{o<f_c}(t_p)^j\cdot\big[1-F_c(t_p)\big]^{n-1-i-j} \\ &\quad\cdot\big[F_c(t_p+\Delta U_{max})\big]^j\Bigg] \qquad (4.55) \end{aligned}$$

As $CR1(t_p)$ also includes the case that no node of PA_{other} encounters a failure (i=j=0), which leads to ΔU_{min}, one needs to subtract the probability of this event leading to $CR2(t_p)$.

$$CR2(t_p) = CR1(t_p) - \big[1-F_N(t_p)\big]^{n-1}\cdot\big[1-F_c(t_p)\big]^{n-1} \qquad (4.56)$$

In the case that ΔU_{min} is entered, all nodes have voted and thus are uncertain. Then no partner for cooperative recovery exists. The probability that ΔU_{min} is entered is given by $P_{U_{min}}(t_p)$, as calculated in Formula (4.27). The probability that PA is blocked due to a node failure of the coordinator during ΔU_{min} and cannot recover cooperatively is now given by $P_{v,cr}(t_p)$.

$$\begin{aligned} P_{v,cr}(t_p) &= PA_{Umax}(t_p)\cdot C_{Umax}(t_p)\cdot CR2(t_p) \\ &\quad + PA_{Umin}(t_p)\cdot C_{Umin}(t_p)\cdot P_{Umin}(t_p) \qquad (4.57) \end{aligned}$$

Predictions Blocking (ii)

Figure (4.12) presents blocking probabilities calculated by $P_{v,cr}(t_p)$ for the example MANET scenario of this work. The important result here is that the probability of PA suffering blocking due to a node failure of the coordinator in ΔU is very small and in fact negligible. For the example MANET scenario, the probability of blocking (ii) is in the 10^{-4} domain and is further reduced by cooperative recovery as shown in Figure 4.12(a) and 4.12(b) by Curve 2.

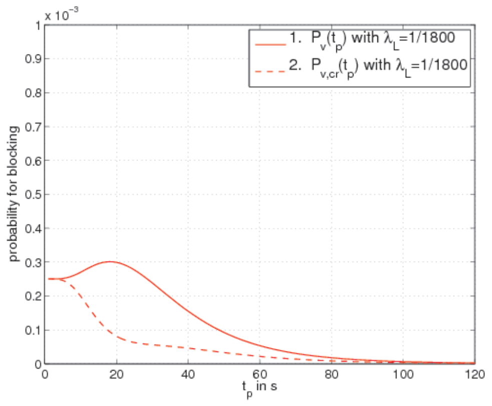
Even if the probability of a node failure is drastically increased, e.g. by assuming that the expected sojourn time of mobile nodes in $\mathcal{A}$ is only 10 min instead of 30 min, the probability of blocking caused by a node failure of the coordinator does not leave the 10^{-3} domain, as shown in Figure 4.12(b).

The main reason for the small blocking risk induced by coordinator node failures is the small size of ΔU with 2PC. The probability of a node failure even within ΔU_{max} is negligible in the example scenario. For the example MANET scenario used here, and scenarios where the probabilities for node and communication failures show similar dimensions, it can be derived that blocking caused by a node failure of the coordinator is a rare event that can be neglected for most scenarios in the strict transaction model.

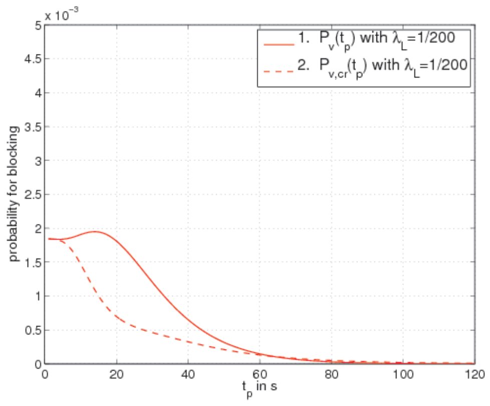
However, calculations presented here are crucial to identify MANET scenarios where blocking (ii) occurs more frequently.

4.6.2 Probability of Extended Uncertainty (ii)

In the semantic model, the end of the processing phase is given by t'_p, which is the time the last operation for PA_{last} is issued by the coordinator. At time



(a) Probability of blocking (ii) in the example MANET scenario with and without cooperative recovery.



(b) Probability of blocking (ii) with and without cooperative recovery and an increased node failure probability (λ_L=1/200).

Figure 4.12: Probability of blocking (ii) for the example MANET scenario with n=3, δ_m=180 ms, Δ_{vo}=1 s, and multi-hop routing.

$t_u = t'_p + \Delta_{ex} + \delta_m$, the coordinator derives the global decision as described in Section 3.6.3. Δ_{ex} also serves as time-out, i.e. if the coordinator does not receive an acknowledgment until t_u, it suspects PA_{last} to be failed and decides on abort.

In contrast to the strict model, all participants but PA_{last} enter uncertainty already during $[t_s, t'_p]$ with acknowledgment of their last operation at t_o. PA_{last} enters uncertainty at $t'_p + \delta_m + \Delta_{ex}$ and remains uncertain for $2\delta_m$. In the following, I will first consider the risk of extended uncertainty in the interval $[t_s, t'_p]$ and afterwards in $[t'_p, t_u]$ from the perspective of PA that is not PA_{last}.

In the interval $[t_s, t'_p]$, a node failure of the coordinator causes an extended uncertainty (ii) situation of PA if the failure occurs after t_o and PA did not previously cause transaction abort. This probability is computed by $Pv_1(t_{f_{n,c}})$, where $t_{f_{n,c}}$ denotes the time of the coordinator's node failure.

$$Pv_1(t_{f_{n,c}}) = \int_{o}^{t_{f_{n,c}}} o(t_o) \cdot \big[1 - F(t_o)\big] dt_o \tag{4.58}$$

The probability that PA is not uncertain and has not caused an abort until $t_{f_{n,c}}$ is given by $Pnv_1(t_{f_{n,c}})$, where $\big[1 - F(t_{f_{n,c}})\big]$ is the probability that PA_{last} does not cause an abort of the transaction until $t_{f_{n,c}}$.

$$Pnv_1(t_{f_{n,c}}) = \int_{t_{f_{n,c}}}^{t_p} o(t_o) \cdot dt_o \big[1 - F(t_{f_{n,c}})\big] \tag{4.59}$$

The calculation of the probability that PA is uncertain and the coordinator suffers a node failure in $[t'_p, t_u]$ has to consider that the transaction has not previously aborted. This probability is given by $Pv_2(t'_p)$.

$$Pv_2(t'_p) = \big[1 - F(t_u)\big] \cdot \big[1 - P_{o>f}(t'_p)\big]^{n-1} \cdot F_N(t'_p..t_u) \tag{4.60}$$

The probability that PA suffers extended uncertainty (ii) in $[t'_p, t_u]$ is directly given by $Pv_2(t'_p)$. For extended uncertainty (ii) caused in $[t_s, t'_p]$, PA is required to be uncertain when the coordinator node failure happens (Line (1) of Formula (4.61)), while $n-2$ nodes in PA_{other} are uncertain or not, which is considered in Line (2) of Formula (4.61) by enumerating all possible combinations of i uncertain and $n-2-i$ certain nodes in PA_{other}. The last participant PA_{last} is required not to cause abort of the transaction in $[t_s, t'_p]$. For the probability that PA suffers from extended uncertainty I now gain $P^*_v(t'_p)$.

$$\begin{aligned} P^*_v(t'_p) \quad = \quad & \int\limits_0^{t'_p} \Bigg[f_N(t_{f_{n,c}}) \cdot Pv_1(t_{f_{n,c}}) \\ & \cdot \sum_{i=0}^{n-2} \left[\binom{n-2}{i} Pv_1(t_{f_{n,c}})^i \cdot Pnv_1(t_{f_{n,c}})^{n-2-i} \right] \\ & \cdot [1 - F(t_{f_{n,c}})] \Bigg] \, dt_{f_{n,c}} + Pv_2(t'_p) \end{aligned} \tag{4.61}$$

Results of $P^*_v(t'_p)$ for the example MANET scenario are given at the end of this subsection.

Extended Uncertainty (ii) with Cooperative Recovery

If PA does not receive the global decision until $t_u + \delta_m$, it executes a cooperative recovery scheme. I compute the probability for this scheme to be unsuccessful.

The probability that PA cannot reach a participant that is certain depends on the probability that all certain participants have suffered a node failure after $t_{f_{n,c}}$ or a communication failure with PA until $t_u + 2\delta_m$. I denote this probability by $C'(t_{f_{n,c}})$.

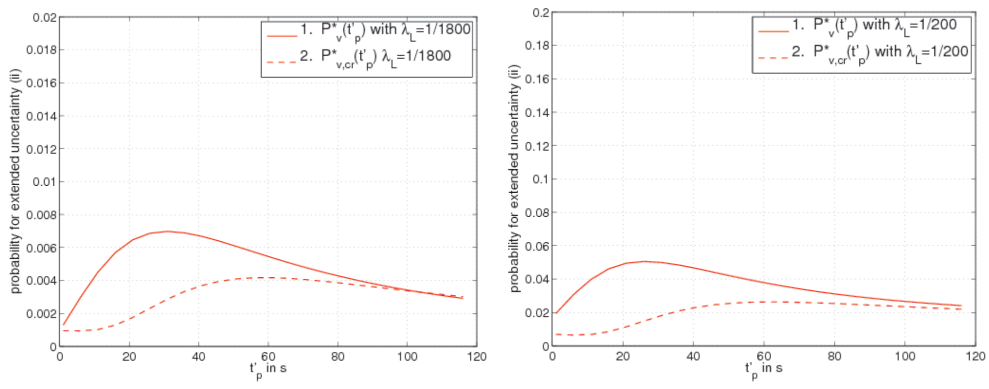
$$C'(t_{f_{n,c}}) = F_N(t_{f_{n,c}}..t_u) + F_c(t_u + 2\delta_m) - F_N(t_{f_{n,c}}..t_u) \cdot F_c(t_u + 2\delta_m) \tag{4.62}$$

In Formula (4.61), I already distinguished between certain and uncertain participants in PA_{other}. To derive the probability that PA suffers from extended uncertainty (ii) and cooperative recovery is not successful, i.e. PA remains uncertain, I expand Formula (4.61) with the probability that no certain participant (PA_{last} and $n-2-i$ of PA_{other}) is reachable for PA, while reachability is computed by $C'(t_{f_{n,c}})$. I then derive $P^*_{v,cr}(t'_p)$.

$$\begin{aligned} P^*_{v,cr}(t'_p) \quad = \quad & \int\limits_0^{t'_p} \Bigg[f_N(t_{f_{n,c}}) \cdot Pv_1(t_{f_{n,c}}) \\ & \cdot \sum_{i=0}^{n-2} \left[\binom{n-2}{i} Pv_1(t_{f_{n,c}})^i \cdot \left[Pnv_1(t_{f_{n,c}}) \cdot C'(t_{f_{n,c}}) \right]^{n-2-i} \right] \\ & \cdot [1 - F(t_{f_{n,c}})] \cdot C'(t_{f_{n,c}}) \Bigg] \, dt_{f_{n,c}} + Pv_2(t'_p) \end{aligned} \tag{4.63}$$

Predictions for the Example MANET Scenario

Figure (4.13) depicts results computed by $P^*_{v,cr}(t'_p)$ for the example MANET scenario of this work. Although the probability for uncertainty caused by a node failure of the coordinator is significantly higher than in the strict case, e.g.



(a) Probability of extended uncertainty (ii) with λ_L=1/1800.

(b) Probability of extended uncertainty (ii) with λ_L=1/200.

Figure 4.13: Probability of extended uncertainty (ii) in the example MANET scenario and transaction parameters n=3, δ_m=180 ms, Δ_{ex}=1 s, and λ_{to}=1 s.

at maximum 0.7 % at t'_p=30 s, it is still low compared to the uncertainty risk induced by participant failures, i.e. of extended uncertainty (i).

For values of t'_p with moderate abort rates ($t'_p \leq 20$ s) the probability of extended uncertainty (ii) is smaller than 0.6 %. Increased uncertainty risks compared to the strict case are caused by the fact that a node failure of the coordinator in $[t_s, t'_p]$ can already cause extended uncertainty (ii) in the semantic model, while in the strict case, only node failures in the interval $[t_p, t_p + \Delta U]$ are relevant.

Cooperative recovery compensates extended uncertainty (ii) situations especially well for small t'_p. For values of t'_p showing a moderate abort probability, the probability of extended uncertainty (ii) and unsuccessful cooperative recovery remains smaller than 0.2 %, as shown in Figure (4.13) by Curve 2. Hence, the probability for uncertainty caused by the coordinator is also negligible for the example MANET scenario and semantic transactions.

Figure 4.13(b) shows the increased probability of extended uncertainty (ii) if a high node failure probability (λ_L=200^{-1}) is assumed. While the risk is significantly increased, cooperative recovery reduces this risk below 1 %. Without cooperative recovery, risks reach at maximum 5 % in the example MANET scenario as shown by Curve 1 in Figure 4.13(b). However, at λ_L=200^{-1} a high abort rate has to be expected possibly rendering transaction processing unfeasible.

4.6.3 Summary - Blocking caused by Coordinator Failure

In this section, I proposed a calculation model for the probability that PA will suffer a blocking or extended uncertainty situation that is caused by a coordinator's node failure.

For the example MANET scenario, I showed that these probabilities are smaller than 1 % for strict and semantic transactions. The risk of PA to suffer extended uncertainty (ii) is with up to 0.7 % larger than blocking (ii) risks in the strict model. Here, the probability of blocking (ii) is smaller than 0.03 %. Even if the node failure probability is drastically increased, blocking (ii) and extended uncertainty (ii) probabilities do not reach significant values. Furthermore, the calculation model presented does not consider recovery from path breaks and therefore provides an upper bound of expected blocking and extended uncertainty risks. The real risks have to be expected to be even smaller, since recovery of communication paths reduces the probability of ΔU to be extended to ΔU_{max}. Additionally I showed that cooperative recovery can reduce these risks even further.

It can be concluded that blocking (ii) and extended uncertainty (ii) situations are less relevant than blocking (i) and extended uncertainty (i) situations in MANET scenarios that are similar to the example MANET scenario of this work. In this section, I presented a probabilistic model to calculate these risks for arbitrary scenarios to identify situations where blocking (ii) and extended uncertainty (ii) is a more relevant situation.

4.7 Case Study - Mission Coordination

In the previous sections, I presented calculation models that were applied to an example MANET scenario without a concrete transaction in mind, but for varying t_p and t'_p. In this section, I apply the calculation model to a concrete transaction scenario, to show abort and blocking risks for a realistic example.

Transaction Description

Th examined transaction scenario is placed in the often-used disaster setting: Assume that multiple rescue teams move into a disaster area and coordinate their missions in a transactional manner. Imagine that an earthquake hits a major city and destroys most of the infrastructure. Medical, fire, police, and rescue units move into the area, and each unit is equipped with a PDA or notebook and connected to a MANET that is formed collaboratively by all rescue units. Rescue missions are coordinated in an ad-hoc manner.

Assume the following situation: a police patrol is informed that residents are suspected to be trapped under a collapsed building. On its way to the collapsed building, the police commander uses his PDA to discover rescue resources and to initiate and coordinate a rescue mission. To initiate the mission, several heterogeneous resources must collaborate at the mission site. The mission control system is assumed to know by configuration that the following resources are required: (i) a rescue unit (trackers) to locate trapped residents and to provide first aid; (ii) technical resources with heavy equipment to remove construction waste to access the trapped. I assume that one part of a technical unit are supply trucks carrying basic supplies such as water, fuel, or blankets.

The atomicity semantic required here is: only if all required resources are available within the right time at the right place can the mission be successful. If one resource, such as trackers, is not available, other units should not be blocked for an arbitrary time, but are better used within other missions. To guarantee successful mission initiation, either all required resources must commit to the mission, or the mission is aborted. I assume that resources are available and therefore no aborts due to unavailability occur.

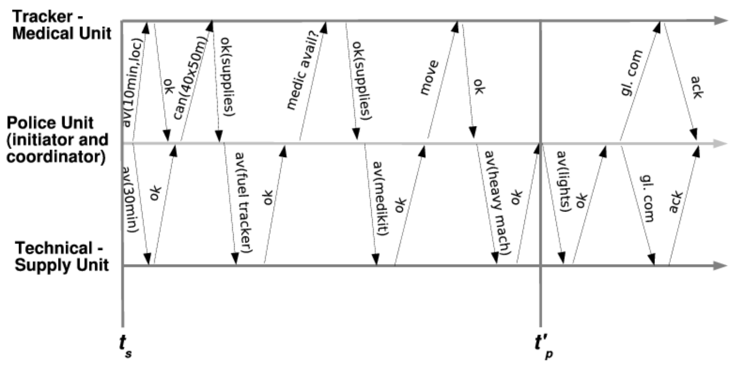


Figure 4.14: Transactional mission control

Rescue resources are contacted and commit themselves to a mission within a transaction. Figure 4.14 depicts the message exchange between the coordinator and resources. The first query issued to every resource checks if the resource is generally available. In the remainder of the processing phase, details and additionally required resources and supplies are negotiated. If negotiation is successful, the resource commits itself to the mission and moves towards the mission site. However, only when the global commit message is received can the resource be sure that the mission is really happening. Hence, a semantic transaction is assumed here. In the following, the semantic of the messages exchanged is described briefly:

Tracker Unit: First a query is sent to check if the tracker team is available and can make it to the mission site in less than 10 min (msg. [av(10 m,loc)]). The mission client running on the track team commander's PDA knows that the tracker team is not involved in another mission yet; it checks the current position of the team and estimates if the team can make it to the mission site in time. Positive feedback (msg. [ok]) is sent to the coordinator. In the following interaction, additional parameters of the mission are negotiated. The coordinator sends an additional query asking if personnel and equipment suffices to search 150 m * 50 m (msg. [can(150x50)]). The tracker unit acknowledges the request positively. To search an area of this size, additional supplies are required, e.g. bottled water, and generators to recharge track robots. This supply request is issued in the [ok(supplies)] message. The coordinator checks back with the technical unit if it can provide the required supplies for the track team. On the mission site, a doctor is also required to provide first aid to salvaged residents. Hence, the coordinator asks if the tracker team has medical personnel who can provide the required service. The tracker unit acknowledges this request positively, but requests additional supplies. The coordinator requests these supplies at the technical unit and gets them (msg. [med. supply ok]). Negotiation with the tracker team is concluded by a [move] message. By acknowledging this message, the tracker team commits to the mission. If all other resources also commit to the mission, global commit is received (msg. [gl. com.]) later.

Technical - Supply Unit: The unit first receives a query if it is available and if it can reach the mission site within the next 30 min. The mission software of the technical team checks whether heavy machines and supply trucks can be moved to the mission site and answers affirmatively. Part of the technical unit are supply trucks, carrying major supplies such as fuel, water, and general rescue equipment, like medical supplies. These supplies are dispatched to mission sites. After confirming availability, the technical unit receives several requests for supplies required by other units (tracker supplies and medical supplies) or required due to special circumstances at the mission site (heavy machines, spotlights etc.). On affirmative feedback on all supply requests, the coordinator derives the global commit decision after the last request is acknowledged and a global commit message is sent to all units. With the reception of the global commit message, the units are certain that the mission was successfully initiated and will actually take place.

Transaction Parameters

From the descriptions above, I derive the transaction parameters t'_p, n, and $o(t_o)$. The processing time t'_p of the transaction depends on the time resources require locally to decide if participation in the mission is possible. For example, if a resource is asked to be in place within 10 min, then it has to calculate e.g. using a navigation system if this is possible. I assume that each resource requires 5 s to decide whether it can participate in the mission. This calculation is done in parallel by the tracker and the technical resource. I assume that the tracker team mission software requires 3 s to estimate the additional supplies required to search the area as well as to decide on the additional resources required by its medical personnel.

I assume that the technical team answers supply requests within 2 s, because the resource management software of the technical team must possibly check sensors or RFIDs to estimate the remaining amount of supplies like water or blankets. Therefore, the processing time of the transaction sums up to t'_p=5s+3s+2s+3s+2s+2s+2s=19s. I assume that an additional second is required for message delays, resulting in t'_p=20 s.

By observing the message flow in Figure (4.14), I derive that the last operation is received within the second half of t'_p. I assume a uniform distribution of t_o in this period. Hence, $o(t_o)$ for the tracker resource is given by

$$o(t_o) = \begin{cases} 0 & for\, t_o < \frac{t'_p}{2} \\ \frac{2}{t'_p} & for\, t_o \geq \frac{t'_p}{2} \end{cases} \tag{4.64}$$

The technical unit is PA_{last} and receives its last operation at t'_p.

MANET Parameters

I assume that the disaster site $\mathcal{A}$ spans 2000 m * 2000 m and 40 rescue units roam within the area. Rescue units generally move within the destroyed streets by trucks, jeeps, or other vehicles. I assume uniform distributed speeds of 1.0–5.0 mps for all teams on the site. Every commander of a rescue team carries a PDA or another mobile device equipped with radio interfaces similar to the 914 MHz Lucent WaveLAN DSSS card with approximately 170 m radio range. The specification of radio and antenna characteristics can be found in

Appendix B.1.1.2. For $F_N(t'_p)$, I assume the distribution as described in Section 4.1.

If the transaction is initiated with participants in 1–2 hop distance, a log-normal distributed cdf $F_C(t_p)$ with μ=3.153 and σ=1.146, a message delay of approx. δ_m=200 ms, and for $F_{RC}(t'_p)$ a log-normal distribution with parameters μ=3.584 and σ=0.820 is derived.

Probability of Abort and Extended Uncertainty

Given the parameters above, the proposed calculation model can now be used to derive abort and uncertainty risks for the given transaction scenario. The derived values are summarized in Table 4.1. I assume that the probability of transaction abort is barely tolerable with 21 %. In the case where all participants know each other and thus, cooperative recovery is possible, the risk of a participant to suffer from an extended uncertainty situation is predicted by 3.7 %–4.5 %. If cooperative recovery is not possible, the risk of extended uncertainty situation is high at 39 %–41 %. The probability that a participant suffers from extended uncertainty (ii) is predicted by 0.3 %–0.8 %.

As in a disaster scenario assumed here a rescue unit should always be freed from stale commitments, a probability of 4.5 % to suffer from extended uncertainty (i) is unacceptable. Hence, recovery schemes in addition to cooperative recovery should be considered for this transaction.

The remainder of this thesis is concerned with such approaches.

Probability	Formula	Value
Abort	$P_a^*(t'_p)$	0.21
Extended uncertainty (i) with path recovery.	$P_u^{*'}(t'_p)$	0.39
Extended uncertainty (i) without path recovery.	$P_u^*(t'_p)$	0.41
Extended uncertainty (i) with cooperative recovery and with path recovery.	$P_{u,cr}^{*'}(t'_p)$	0.037
Extended uncertainty (i) with cooperative recovery and without path recovery.	$P_{u,cr}^*(t'_p)$	0.045
Extended uncertainty (i) with cooperative recovery using P_{path}.	$P_{u,cr}^{*pp}(t'_p)$	0.05
Extended uncertainty (ii).	$P'_u(t'_p)$	0.008
Extended uncertainty (ii) with cooperative recovery.	$P'_{u,cr}(t'_p)$	0.003

Table 4.1: Abort and uncertainty risk with t'_p=20 s and n=2.

4.8 Summary and Conclusion

This chapter presented an in-depth investigation of abort and blocking probabilities in the strict and semantic transaction model for MANETs. I proposed a probabilistic model to estimate the blocking and abort rates of transactions in an arbitrary MANET scenario. The model was verified by simulation for an example MANET scenario.

The measured and calculated abort and blocking probabilities for the example scenario showed some interesting results that can be expected to hold for similar transaction and MANET scenarios: only if failure probabilities are high, blocking and extended uncertainty situations can be observed at an noticeable rate. But, high failure probabilities predominantly increase the probability for transaction abort, which prevents a strong increase of blocking risks. I showed that the simple rationale that a lot of failures cause a lot of blocking and extended uncertainty situations is not true. High blocking probabilities are only observed for specific combinations of transaction and MANET parameters. Therefore, reasoning about the relation of failure probabilities and abort as well as blocking probabilities is complex and requires a formal model as presented in this chapter. In short, the most important results presented in this chapter are:

- Increased failure probabilities predominantly raise the susceptibility of transaction abort, while the increase of blocking risks is moderate or even recurrent, since less transactions enter the decision phase. Abort risks increase fast for increasing t_p and therefore a calculation model as developed here is required to identify the spectrum of t_p with moderate abort probabilities.

- Blocking situations that cannot be immediately compensated for by cooperative recovery are observed only for a certain range of t_p in a MANET scenario. To identify these specific transaction sizes, a calculation model as presented here is important.

- It showed that the abstractions made by the presented model are sufficient to provide accurate predictions of abort and blocking risks. Hence, the model can be used to decide whether additional recovery schemes such as the SLS presented in the following chapter should be integrated for a given transaction.

- The semantic transaction model is generally more susceptible to extended uncertainty, than the strict model is to blocking. It is also less susceptible to abort as no collection phase exist.
  Semantic transactions with two participants is the transaction scenario with the highest probability of extended uncertainty identified.

- Cooperative recovery is a most efficient recovery scheme in all blocking and extended uncertainty situations. For most strict transactions, cooperative recovery reduces the probability for blocking below 1 % in the example MANET scenario.

- From the two defined blocking situations; blocking (i) and blocking (ii) and extended uncertainty (i) and extended uncertainty (ii) respectively, the situations caused by participant failures are much more likely, than the situations caused by a node failure of the coordinator. In fact, the probability that a node failure of the coordinator causes blocking of a participant is negligible for the MANET scenarios considered here.

The model presented is useful, as it allows to decide, whether it is feasible to execute a transaction with a given processing time, or if a transaction will most likely abort. Proposed predictions of blocking rates can be used to decide if additional recovery schemes like the SLS, as presented in the following chapter, or a backup coordinator as proposed in Chapter 6 should be integrated in transaction processing. A prophylactic integration of the SLS or a backup coordinator respectively is not recommended, as such schemes always induce an additional message overhead.

Additional recovery schemes like the SLS are generally required, as I showed in the previous sections that transactions scenarios exist, where even cooperative recovery does not reduce the risk of extended uncertainty sufficiently. The remainder of this work will be concerned with such recovery schemes reducing the probability of blocking.

The problem of high abort rates requires work on advanced transaction models that are more failure tolerant within the processing phase. In further research, the concepts of advanced transaction models described in Chapter 3 should be integrated in the calculation model presented here. I consider such an enhancement of the calculation model as straightforward. Reduced abort rates of advanced transaction models are expected to cause higher blocking and extended uncertainty risks, since more transaction will enter the decision phase. This will also increase the demand for advanced recovery schemes such as the SLS or backup coordinators to compensate for blocking.

Chapter 5

Shared Log Space (SLS)

This chapter describes the idea, architecture, and implementation of the Shared Log Space (SLS). The SLS is a shared distributed storage that allows to save the global decision of a transaction within a MANET at a defined availability. In this chapter, I will describe how the SLS is used to compensate for blocking (i) and extended uncertainty (i) situations. In Chapter 6 it will be shown how the SLS is used with a backup coordinator to compensate for blocking (ii) and extended uncertainty (ii) situations. Embedding the SLS into atomic transaction processing allows one to guarantee that successful recovery from blocking or extended uncertainty has a defined probability.

The probability of successful recovery depends on the availability of the global transaction decision at recovery time. The basic idea of the SLS is to control this availability by storing the decision log on multiple nodes of the MANET. From the application developer's perspective and for the designer of recovery protocols, the SLS provides a convenient abstraction, since the details of preserving the commit decision at desired availability as well as the retrieval of the decision at recovery time are completely hidden.

Although the SLS is presented here in the context of atomic transaction processing, its functionality is also relevant in other domains. For example, in [30] it is proposed to write signed messages of eCommerce transaction to the SLS to allow for weak fairness [9] in transactional goods exchange. In [65] a distributed reputation system benefits from the SLS by preserving information within the SLS that proves unfair behavior of nodes.

To implement the proposed abstraction, two major problems have to be addressed: (i) since node movement and node failures influence the probability that a blocked node can access the decision log, the problem of estimating the optimal set of nodes to place log copies on has to be solved; and (ii) in coordination with the dissemination of the decision, the according retrieval operation executed by an uncertain participant at recovery time has to locate the decision log fast and at low message expenses.

The remainder of this chapter is structured as follows: In the first three sections of this chapter, I present the general concept of the SLS, while a detailed description of implementation issues and an evaluation is given within the last five sections. The conceptualization given in the first sections is biased by the lightweight implementation approach, which is the main implementation approach followed in this work. Hence, some concepts only relevant for this approach are also described with the general SLS concepts.

In more detail, Section 5.1 describes the basic idea of the SLS, related work, and the concrete failure cases addressed, while Section 5.2 provides a formalization of the SLS operations. Section 5.3 presents how SLS operations are integrated into recovery protocols of strict and semantic transactions. Sections 5.4 and 5.5 present major implementation problems and their solutions such as derivation of node failure probabilities and the log availability model underlying the lightweight approach. In Section 5.6 the implementation of SLS operations in the lightweight approach is described and evaluated afterwards in Section 5.7. Section 5.8 briefly presents and evaluates an alternative implementation approach of the SLS based on a cluster overlay. Finally, Section 5.9 concludes the chapter.

5.1 Idea and Related Work

The main problem addressed in this chapter is compensation for blocking and extended uncertainty situations that are caused by node or communication failures of transaction participants. In the following, these situations are briefly reviewed and the basic idea of how the SLS increases the chance of participants to leave uncertainty is introduced. Since the SLS is based on controlling the availability of the decision log within a MANET, related work on data availability in MANETs is discussed.

5.1.1 Problem and Idea

Blocking (i) and extended uncertainty (i) situations occur if a participant moves into uncertainty and does not receive the global decision due to a communication failure with the coordinator or because the participant disconnects from the MANET. From the coordinator's perspective, it is indistinguishable whether a participant experiences such a situation, or whether the acknowledgment message for the global decision was lost and the participant was able to leave uncertainty. Hence, the coordinator cannot securely decide on whether a participant is uncertain or not.

In contrast to the blocking situations caused by a node failure of the coordinator, in blocking (i) and extended uncertainty (i) situations the coordinator is still connected to $\mathcal{A}$ and therefore the global decision is generally existent in $\mathcal{A}$. Thus, the main issue of efficient transaction termination is to deliver this transaction decision to the uncertain participant rather than deriving a global decision.

The basic structure of the participant's termination protocol executed in case of blocking (i) and extended uncertainty (i) is to loop through all possibilities to learn about the global decision as shown in Figure 5.1. These options are: (i) to contact other participants; and (ii) retry to contact the coordinator. The main idea of the SLS architecture is to allow uncertain participants to leave the termination loop faster, by adding the SLS as third option to learn about the global decision to the termination protocol. In the following, the time the termination protocol of the participant is executed is denoted by $t_{ter,p}$, while the coordinator executes its termination protocol at time $t_{ter,c}$.

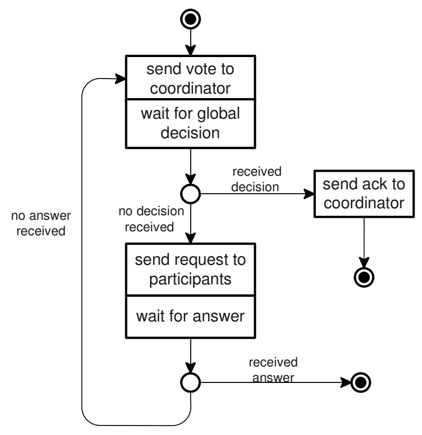


Figure 5.1: Participant termination protocol with cooperative recovery.

Since the coordinator can neither decide whether a participant is uncertain, nor whether a participant is blocked due to a communication or node failure, the coordinator's termination protocol has to consider all three situations. In the standard termination protocol, the coordinator passively waits for a request of the restarting or terminating participant. The coordinator cannot decide on the state of participants, i.e. whether a possibly uncertain participant is still connected to $\mathcal{A}$. Hence, the coordinator cannot decide on whether a participant executes a termination or restart protocol.

The participant's restart protocol is executed at reconnection to $\mathcal{A}$ if a participant suffered a node failure while uncertain. The restart protocol executed has the same structure as the participant's termination protocol. It loops through the two options: (i) contacting other participants; and (ii) contacting the coordinator to learn about the global decision. Hence, the same idea as above applies here: increased availability of the decision log also raises the probability of the restarting participant to leave uncertainty. Similar to the participant termination protocol, efficiency of the restart protocol can be increased by adding a third option to learn about the global decision.

The important difference between the termination and restart protocols is the time when the protocols are initiated. While the participant's termination protocol in the strict model is executed at time $t_{ter,p} = t_p + \Delta U_{max} + 2\delta_m$ and at $t'_{ter,p} = t'_p + \Delta_{ex} + 2\delta_m$ in the semantic model, the restart protocol of participants is executed at the time when the uncertain participant reconnects to $\mathcal{A}$. I denote this point in time by t_{res}, while t_{res} is unknown. Since the coordinator cannot distinguish whether a participant suffered a communication or a node failure, it cannot decide when the decision log is required by the uncertain participant. Therefore, increased availability of the decision log has to be assured until the latest possible time, when recovery of the participant is expected.

As described above, recovering participants remain blocked because communication between the participant and the coordinator fails, either because the coordinator has left the network or because no communication path is available.

The key idea of SLS is to solve this dilemma by decoupling communication of nodes from their presence in $\mathcal{A}$ at recovery time. This idea is inspired by the concepts of shared communication spaces such as Linda [51] or by its counterpart in mobile environments Lime [104, 103]. Both systems decouple communication between processes (Linda) and mobile nodes (Lime) in time and space, i.e. nodes or processes need not necessarily be connected at the same time to exchange messages.

The idea proposed here, adapts such concepts to communicate the global decision of a transaction to recovering participants, independently of the presence of previous transaction partners and the coordinator in $\mathcal{A}$ at recovery time. A recovering node which is not able to reach the coordinator should be able to learn about the global decision using the SLS to leave uncertainty at high probability. Therefore the SLS establishes a distributed shared storage for small data items guaranteeing a certain availability of such data in presence of churn and constant changes in the network topology of $\mathcal{A}$.

Several other works addressed the problem of data availability in MANETs. I therefore review work related to the SLS in the following, before the architecture of the SLS is described in more detail.

5.1.2 Related Work

Related work dealing with data availability in MANETs can be found in different areas. The by far most publications dealing with data availability can be found in the context of data replication [131, 111, 69, 140, 76, 125, 130, 68]. Other works like [157, 92] address data availability in the context of caching or hoarding.

The main objective in data replication is to provide a high availability of consistent data that is distributed among multiple nodes, while various nodes may update data. The main problems to solve in data replication are: (i) allocation of replicas to achieve a high availability; (ii) propagation of updates to all nodes holding a copy of a manipulated data object; and (iii) providing consistency of accessed data, i.e. anticipating or resolving conflicts due to concurrent updates. While (ii) is mostly solved by flooding approaches such as hyper-gossiping, probabilistic flooding, or multicast schemes and message overhead is minimized by adapting propagation intervals to access frequencies, (iii) is a more complex problem.

In the area of distributed databases, various replication and replica control schemes have been proposed, which can be classified as *strict replica control* protocols and *lazy replica control* protocols [160]. Strict replica control enforces one-copy equivalence. Prominent protocols of this class are ROWA and ROWA-A that require the availability of all replicas or quorum schemes that require nodes to obtain a quorum for update operations [143]. Lazy replication protocols provide eventual consistency only, as updates are performed and propagated to the other copies in a lazy manner. Several approaches have been proposed based on time stamps or node priorities to solve update conflicts. However, with lazy replication temporary inconsistent states are inevitable. Strict as well as lazy replication control approaches have been transferred to MANETs. For example, in [76], the authors propose to maintain consistency based on a quorum approach in MANETs, while in [125] lazy update propagation is proposed with a conflict resolution based on time stamps. The authors of [125] propose that all nodes maintain update operations on data objects within a tree structure, allowing to reconciliate updates in arbitrary orders. Additionally, a local and global consistency level is defined for replicas and methods are proposed to guarantee consistency among replicas.

Problem (i) is related to this work in the sense that in replica allocation it is decided on which nodes replicas are saved to achieve high data availability. Numerous approaches have been proposed for MANETs. I refer to [68] for a comprehensive overview of allocation schemes in MANETs. Proposed schemes allocate replicas based on access frequencies (of single nodes or within a neighborhood) or consider the network topology to decide on which nodes to place copies on. The idea of allocation schemes considering the network topology is to place replicas in a way that in case of network partitioning replicas are available in each partition, while access frequencies are used to allocate copies within partitions. As prediction of network partitioning is one of the hardest and still unsolved problems in MANETs, the general approach is to discover network components with a high degree of connectness, as such components will most likely not be partitioned. Discovery of connected components requires either knowledge of the complete network topology and can then be done using graph algorithms or clustering concepts. As the topology of a MANET constantly changes, the allocation of copies must be constantly maintained and controlled, resulting in a high message overhead observed for these approaches.

As the SLS does not allow for update operations on decision logs, problems (ii) and (iii) induced by the update anywhere property are not relevant for this work. Allocation schemes using access frequencies are not applicable in the context of the SLS as the decision log is at most read once by a node. The idea of considering the network topology to achieve high log availability is adapted in the cluster-based implementation approach of the SLS presented in Section 5.8.

Research on caching in MANETs is related to this work in some ways, as the objective of caching in MANETs is to increase the data availability in areas of the network where it is frequently accessed. As with replication schemes the main challenge here is to develop appropriate allocation schemes [157, 92]. While the main objective of caching is to decrease the time needed to access a data object, the primary concern of the SLS is that a decision log can be retrieved at all within defined time bounds.

5.2 SLS Architecture

In this section, the basic abstractions of the SLS system are introduced. As mentioned above, the SLS is a distributed system formed by collaborative and fair acting nodes in the system model of this work. For simplicity, it is assumed that all nodes of a MANET participate in the SLS System.

5.2.1 Preliminaries and Definitions

First, let me introduce some important concepts and definitions required later. A central problem to be solved by an implementation of the SLS is to predict and control the availability of a decision log at an arbitrary time t in the future. Among other things, this calculation requires to estimate the probability that a specific node is connected to $\mathcal{A}$ at time t. I call this the *probability of presence* of

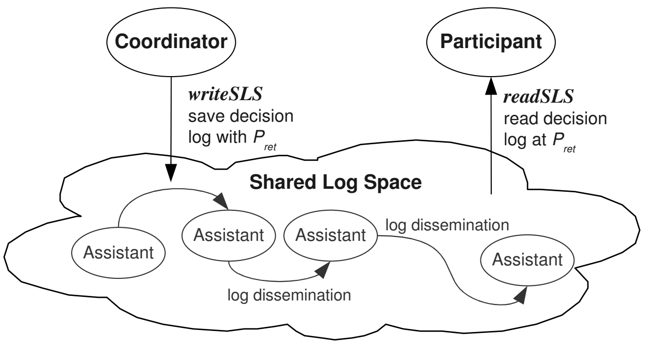


Figure 5.2: Abstract SLS model.

i in $\mathcal{A}$ at time t; it is denoted by $P_{i,pr}(t)$. The probability that a node i holding a log is connected to $\mathcal{A}$ and an uncertain participant can reach this node is called *reachability* of i. Nodes that hold a decision log of a transaction, although they have never participated in the transaction, are called *assistant* nodes.

Before a participant moves into uncertainty, it announces a so-called *retrieval probability* denoted by P_{ret} to the coordinator. This is the probability that the participant wants to be able to retrieve the decision log in case it suffers a blocking or extended uncertainty situation.

Since providing availability of a decision log for unlimited time is expensive in terms of messages, a time limit until the retrieval probability P_{ret} of the decision log should be provided is defined. Therefore, every retrieval probability announced by a participant is combined with a so-called *mission time* denoted by t_m, describing the duration the participant requires P_{ret} to be maintained. Another reason to introduce t_m besides high message costs is the fact that it is hard to decide when a log item is obsolete. This requires consensus among all transaction participants on whether the log is still needed and hence is as hard to solve as atomic commit itself.

5.2.2 SLS Abstractions and Operations

The SLS is collectively formed by assistant nodes that cooperatively store transaction logs to increase their availability for recovering participants. The basic concept of the SLS is depicted in Figure 5.2. It is assumed that a middleware component, a so-called Local Log Space (LLS) component resides on every node of the MANET. An application accesses the SLS through the LLS by the *writeSLS*, *readSLS*, and *register* operation. In the following, a brief formalization of these operations is given while a more detailed discussion of the problems to be solved by each operation is given later.

- **writeSLS[**tid, log,P_{ret},t_m,k,Δw**]:** The *writeSLS* operation is invoked by the coordinator to preserve a decision log within $\mathcal{A}$ at availability P_{ret} until t_m. The operation is not allowed to terminate before the promised log availability is reached. However, the *writeSLS* operation is not allowed to execute for an arbitrary long time, but has to terminate at least after time Δ_w. If available, a so-called dissemination plan k is considered, that is derived by executing the *register* operation.

- **readSLS[*tid*,*k*]**$\rightarrow$ *log*: A recovering node can execute the *readSLS* operation to retrieve a decision log previously written to the SLS by a *writeSLS* operation. The *tid* parameter identifies the transaction the decision log should be retrieved for, while k is the dissemination plan used by the according *writeSLS* operation. The concept of k is described below. k can be null if unknown. The *readSLS* operation has to return the desired log item with probability P_{ret} until t_m within time Δ_r.

- **register[*tid*,t_m,P_{ret},t_p]**$\rightarrow$ k: The register operation asserts whether P_{ret} can be generally provided in a MANET and in some scenarios prepares a so-called *dissemination plan k.* A dissemination plan k is a concept of the lightweight implementation approach and allows for a message efficient dissemination and retrieval scheme of decision logs if k is communicated to participants that expect to execute a *readSLS* operation. Parameters t_m, P_{ret}, and t_p are required to prepare k to meet P_{ret} until t_m. t_p is required to consider aging of k. Plan aging and the abstraction of k is described below. Whether a plan k can be used depends on the MANET scenario, i.e. whether multi-hop routing is used or not.

In the following, the concept of the dissemination plan is described and responsibilities and problems to solve by each SLS operation are discussed.

5.2.2.1 Dissemination Plan

To achieve the demanded availability of the decision log at minimum cost, the dissemination plan k is central. In the lightweight implementation approach k defines which assistant nodes will be used to place the decision log on. k contains either a set of assistant nodes identified by their nodeId. For example, if $k = \{23, 12, 55\}$, then the retrieval process expects that the decision log was distributed to nodes with nodeId 23, 12, and 55. If multi-hop routing is used, such information allows for efficient dissemination and retrieval schemes that do not require expensive broadcast and flooding mechanisms, because the nodes in k can be directly addressed. However, it is not always possible to determine a set of defined assistant nodes a priori, either because not enough nodes can be found in the current vicinity or because the reachability of assistant nodes will most likely have changed when the *writeSLS* operation is executed. Hence, k can also contain an undefined set of assistant nodes. The undefined set of assistant nodes is simply a number of nodes, e.g. $k = \{23, 12, 55, (3)\}$ states that the decision log should be disseminated to nodes 23, 12, 55, and additionally to any three other nodes.

The dissemination plan is determined by the *register* operation and must be made available to participant nodes before a *readSLS* operation is executed. The most severe problem of the dissemination plan is its validity. If the time between calculation of k and realization of k is large, nodes available at calculation time are possibly not reachable when k is executed. I call this problem *plan aging.* The age of k is simply measured by the time elapsed since *register* was called and is denoted by Δ_k.

However, a dissemination plan is only feasible in MANET scenarios where multi-hop routing is used, as only then a recovering node can address assistant nodes defined in k directly. If no multi-hop routing scheme is used, k is not used.

5.2.2.2 SLS Operations

In the following, the responsibilities and problems to be solved by implementations of the SLS operations are discussed. The operations are described as used in the lightweight implementation approach, where the register operation plays a decisive role. Problems to be solved by operations differ for scenarios with and without multi-hop routing.

register - Operation

In the case where multi-hop routing is used, the main objective of the register operation is to derive a dissemination plan k, which is then used by the *writeSLS* and *readSLS* operations. Three main actions must be performed within the *register* operation: (i) it has be assured that the demanded log availability P_{ret} can be provided at all in the given MANET; (ii) potential assistant nodes have to be discovered; and (iii) based on the discovered assistant set the optimal dissemination plan k has to be derived. In the following, these actions are described in more detail:

(i) Assert demanded log availability. Before any calculations to derive k are done, the *register* operation has to assert, whether the demanded log availability P_{ret} can be achieved at all. This is generally done by comparing the demanded P_{ret} to the maximal possible log availability of the given MANET scenario denoted by P_{max}. P_{max} is achieved if the log is saved by $n_{\mathcal{A}}$ nodes of $\mathcal{A}$. I will show in Section 5.5.4 that high levels of P_{ret} cannot be met in all scenarios, especially if the node failure probability and churn rate is high and t_m is large. The main practical problem here is to derive P_{max}.

(ii) Discovery of potential assistant nodes. To determine a defined set of assistant nodes to be used in k, reachable nodes have to be discovered first. These nodes may be discovered in the direct neighborhood or in multi-hop distances, while the message complexity of the discovery process should be small. To derive the optimal dissemination plan, the *short-term* and *long-term reachability* of nodes has to be considered. Short-term reachability is the probability that an assistant node discovered at the time when the *register* operation is executed is reachable to save the decision log when the *writeSLS* operation is initiated. The long-term reachability of a node is the probability that an assistant node can be reached by a recovering node at time t_{res} and $t_{ter,p}$ respectively. As these probabilities may vary among the potential assistant nodes, such reachability information has to be acquired within the discovery process.

(iii) Calculation of dissemination plan. After the available assistant nodes and their failure probabilities are discovered, such information is used to calculate an optimal k. The optimal k is the set of assistant nodes that, within

t_m, guarantees availability of the decision log that is larger than P_{ret} using the smallest set of discovered assistants.

A major problem to consider is aging of k. Individual nodes defined in k are possibly not reachable when the *writeSLS* operation is called. The probability of this situation increases if the period between execution of the *register* and *writeSLS* operation is large. Therefore, the timing of the *register* method in relation to a *writeSLS* invocation is critical in protocol design. Considering the short-term reachability of potential assistant nodes is crucial to anticipate plan aging. The log availability achieved by a plan k considering plan aging is called $LA^*(k, t_m)$. Hence, the plan derived by the *register* operation has to assure that $\forall t \in [t_{ter,c}, t_m] \mid LA^*(k, t) \geq P_{ret}$.

writeSLS - Operation

The *writeSLS* operation is responsible for placing the decision log on assistant nodes such that the demanded availability P_{ret} of the decision log is assured. This is achieved by considering the dissemination plan k but also requires to compute the actually achieved log availability at execution time. I call the realized plan k_r while the plan derived by the *register* operation is denoted by k. In case k cannot be realized due to plan aging, new assistants have be discovered within the *writeSLS* operation to assure that $\forall t \in [t_{ter,c}, t_m] \mid LA(k_r, t) \geq P_{ret}$, where $LA(k_r, t)$ describes the log availability achieved by the actually realized dissemination.

Another main requirement of the *writeSLS* operation is fast execution. This is crucial as a node failure interrupting the *writeSLS* operation jeopardizes the promised availability of the decision log if it hits the coordinator before a sufficient availability of the decision log is achieved. While the *writeSLS* operation is only allowed to terminate after sufficient log availability is assured, an upper time bound for the execution time denoted by Δ_w should not be exceeded. To solve this problem, the *writeSLS* operation has to provide a contingency plan if the execution limit Δ_w approaches and a sufficient log availability is not reached yet. A possible contingency plan is to initiate a broadcast-in-time scheme that disseminates the decision log to all nodes in $\mathcal{A}$. The major challenge to be solved by an implementation of the *writeSLS* operation is to find a balance between observing Δ_w and fallback to a suboptimal contingency plan (e.g. flooding).

In case no multi-hop routing is used, k is null and thus not considered. k_r then contains only assistants discovered at execution time of the *writeSLS* operation. P_{ret}, t_m, and Δ_w are mandatory parameters as well as log and tid.

readSLS - Operation

The responsibility of the *readSLS* operation is to discover at least one assistant node holding the decision log for transaction tid and return it to the requesting application. The main challenge here is to find a message-efficient and fast scheme to locate assistant nodes. Knowledge provided by the dissemination plan k can be used to directly contact nodes carrying the desired decision log.

As mentioned above, the *readSLS* operation must return the desired decision log after Δ_r at a probability equally to or larger than P_{ret} until t_m. Δ_r depends on the implementation of the *readSLS* operation.

Discovering an assistant node at minimum cost is the central problem to solve by an implementation of the *readSLS* operation, especially in a scenario where no routing is used and hence no k is given. In this case, a search request has to be distributed within $\mathcal{A}$, e.g. using a ring search or flooding protocol. The problem to overcome here is to find a balance between expensive broadcast-in-space schemes and cheaper schemes querying only a certain area or zone of $\mathcal{A}$.

5.3 Integration of SLS in Recovery Protocols

Using the SLS in atomic commit protocols requires the integration of the *register*, *writeSLS,* and *readSLS* methods into given transaction protocols. In the following, I will describe the coordinator termination protocol and the participant termination and restart protocols using the SLS for the strict and semantic transaction models.

Before these operations can be used, coordinator and participants must agree on values for P_{ret} and t_m. I assume that participants demand individual P_{ret} and t_m. Considering individual demands of participants on P_{ret} and t_m is straightforward if the *writeSLS* operation assures the availability required by the possibly uncertain participant, i.e. the participant from which no acknowledgment was received. However, k has to meet the highest P_{ret} requested by participants.

The *writeSLS* operation is embedded in the termination protocol of the coordinator, while the *readSLS* operation is integrated into the termination and restart protocols of participants. On transaction completion the coordinator executes the *writeSLS* operation if a participant is suspected to have suffered a failure during uncertainty.

5.3.1 Coordinator Termination Protocol with SLS

If an acknowledgment for the global decision is missing, the global decision has to be saved to the SLS to allow recovering participants to retrieve the decision log from the SLS. The *writeSLS* operation is expensive in terms of messages and therefore should only be executed if participants are actually suspected to be blocked and not in a prophylactical manner. A prophylactical execution of the *writeSLS* operation is not feasible, since the the probability of blocking is low (see Chapter 4) and therefore also the execution of a corresponding *readSLS* operation.

Except for the case when the acknowledgment of a participant for the global decision is lost, the only operation causing message overhead in the failure-free case is the *register* operation. This is inevitable, as this operation has to be executed before any participant moves into uncertainty to assure that participants know k at recovery time. If no multi-hop routing is used, the *register* method does not derive a dissemination plan, but asserts only whether $P_{ret} < P_{max}$, which causes no extra message overhead if P_{max} is known. In the following, the coordinator's termination protocol is described for the strict and semantic transaction model.

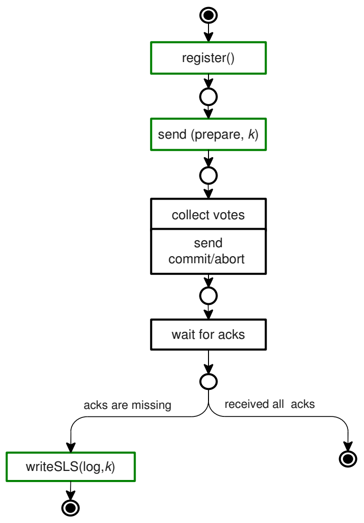


Figure 5.3: Coordinator termination protocol with SLS in the strict model.

5.3.1.1 Strict Transaction Model

In Figure 5.3, the termination protocol of the coordinator is depicted for the strict transaction model. In the strict model, all nodes move into uncertainty simultaneously when answering the prepare request. Thus, the prepare message is the last message received by a participant before entering uncertainty. For message efficiency, the dissemination plan k is sent together with the prepare command as a single message. Important is that the dissemination plan k is valid with high probability when communicated to participant nodes, i.e. k should be as new as possible. With increasing time elapsed between the creation of k and execution of the *writeSLS* operation, the problem of plan aging becomes more relevant, reducing message efficiency of the approach.

The problem to solve is to execute the *register* method early enough to not artificially delay the processing phase, and late enough to minimize the effect of aging of k. Δ_k is given by $t_p - t_{tr} + \Delta U$, where t_{tr} is the point in time when the *register* operation terminates. Hence, the minimum age of k is given by ΔU in case $t_{tr} = t_p$. Since ΔU is small in the strict model (see Chapter 4), aging of k is not as severe as in the semantic transaction model which is described in the following.

5.3.1.2 Semantic Transaction Model

In the semantic transaction model, nodes move into uncertainty after acknowledging their last operation. In contrast to the strict scheme, it is possibly required to execute the *writeSLS* operation during the processing phase of a transaction in the interval $[t_s, t'_p]$. A plan k has to be available and known by uncertain participants before the *writeSLS* operation is executed. Since, the uncertainty windows of participants are considerably larger in the semantic model than in the strict model and k has to be derived before any participants move

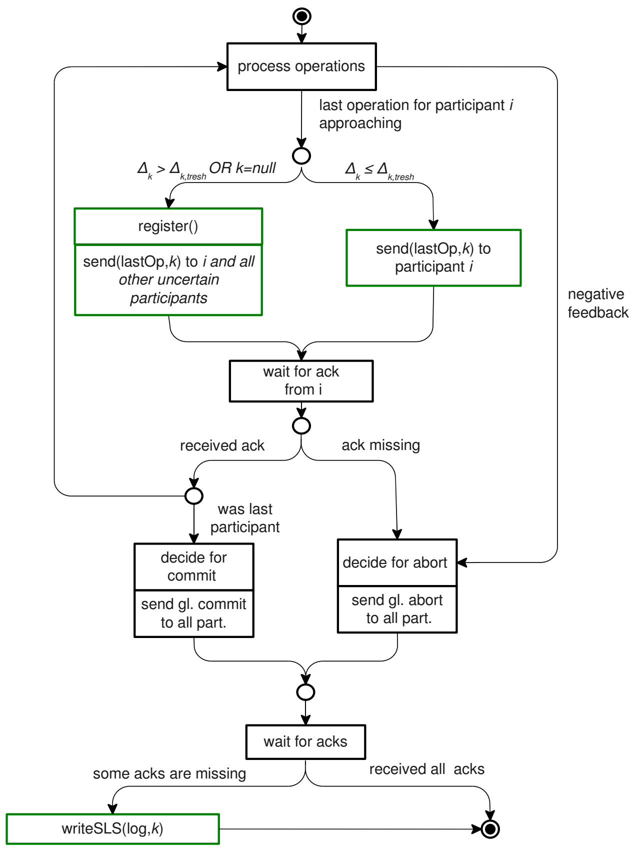

process operations
last operation for participant *i* approaching
$\Delta_k > \Delta_{k,tresh}$ OR k=null
$\Delta_k \leq \Delta_{k,tresh}$
register()
send(lastOp,*k*) to *i and all other uncertain participants*
send(lastOp,*k*) to participant *i*
negative feedback
wait for ack from i
received ack
ack missing
was last participant
decide for commit
send gl. commit to all part.
decide for abort
send gl. abort to all part.
wait for acks
some acks are missing
received all acks
writeSLS(log,*k*)

Figure 5.4: Coordinator termination protocol with SLS in the semantic model.

into uncertainty, plan aging is a more severe problem. Figure 5.4 depicts the proposed termination protocol for semantic transactions considering aging of k.

I propose the approach to renew k if Δ_k reaches a threshold and to propagate the new k to all uncertain participants. However, it is not guaranteed that every participant can receive the most recent k, but the chance that an uncertain participant knows a more recent k is increased compared to the approach to derive a single k before the first participant enters uncertainty. Another approach would be to review k by checking the reachability of the assistant nodes defined in k. However, this increases message overhead and is only feasible if k can be revised at no additional cost, e.g. if a lookup on the local routing table is possible to verify k.

In this work, I propose to use a threshold value $\Delta_{k,thr}$ indicating the allowed age of a plan. Figure 5.4 depicts the termination protocol considering revision of k depending on its age: before the last operation for a participant i is issued, the coordinator checks whether k exists and verification is required. If Δ_k exceeds threshold $\Delta_{k,thr}$ or no k has been derived yet, the *register* operation is called. If k does not require verification, it is sent to participant i as a single message with the last operation of i. If a new plan is derived, k is not only sent to i but also to all participants that have already moved into uncertainty. Afterwards, the coordinator waits for acknowledgments from participant i for successful execution of its last operation. If the acknowledgment is missing, the coordinator decides on global abort and sends its global decision to all participants. If at least one acknowledgment message for the global decision is not received, then the *writeSLS* operation is executed by the coordinator. If a failure occurs with a participant i in $[t_s, t_{i,o}]$ the transaction is aborted and the *writeSLS* operation is executed if at least one acknowledgment of uncertain participants is missing.

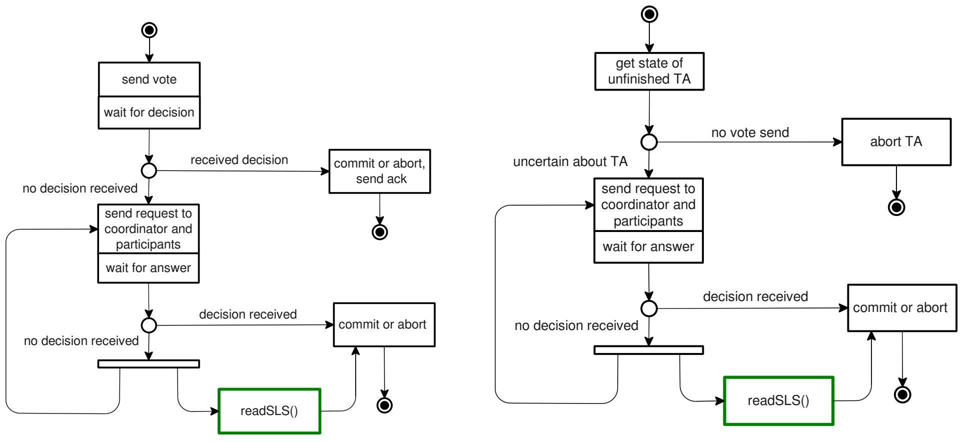
If there is no failure, the coordinator will decide on commit if the last participant acknowledges its local transaction branch. Only if an acknowledgment for the global commit message is missing from a participant, the *writeSLS* operation is executed using the latest k. In the case where all acknowledgments are received, the only message overhead is produced by the *register* operation.

5.3.2 Participant Termination and Restart using the SLS

To integrate the SLS into the termination and restart protocols of participants, the *readSLS* operation is embedded in the standard participant protocols. Recall that the restart and termination protocols are considered to be executed in $\mathcal{A}$ only. Nodes disconnected from $\mathcal{A}$ defer execution of the restart protocol until reconnected to $\mathcal{A}$.

Integration of the *readSLS* operation in participant termination and restart protocols is straightforward. After all standard options to learn about the transaction decision: (i) contacting the coordinator and (ii) contacting the other participants, are not successful, the *readSLS* operation is executed. The motivation to try (i) and (ii) first, is the potentially high message cost of the *readSLS* operation.

In the following, the participant termination and restart protocols for strict and semantic transactions are described.



(a) Termination protocol with SLS. (b) Restart protocol with SLS.

Figure 5.5: Participant termination and restart protocols with SLS in the strict transaction model.

5.3.2.1 Participant Termination Protocol with SLS

The participant termination protocol is executed at time $t_{ter,p}$. Figure 5.5(a) depicts the termination protocol in the strict case. As mentioned above, it is feasible to initiate a cooperative recovery cycle first, before the *readSLS* operation is executed. If cooperative recovery is not successful, the *readSLS* operation is called and executes in parallel to the repeatedly executed cooperative recovery scheme as shown in Figure 5.5(a). The *readSLS* operation returns the decision log with probability P_{ret} after time Δ_r. If cooperative recovery is successful before, the protocol terminates, as well as the *readSLS* operation terminates the protocol when successful.

Figure 5.6 depicts the participant termination protocol with SLS for the semantic transaction model. The protocol has the same structure as the protocol of the strict case. The main difference is that a compensation transaction has to be executed if the global decision conflicts with the local one.

5.3.2.2 Participant Restart Protocol with SLS

Figures 5.5(b) and 5.7 depict the restart protocol with SLS for the strict and semantic models. These protocols are similar in their structure to the termination protocols described above, with the difference that the decision phase is not considered. In the strict and semantic cases, it is first checked whether a transaction is undecided. In such a situation, the coordinator and all participants of the transaction are contacted first to learn about the global decision. If no answer is received, the *readSLS* operation is executed in parallel to cooperative recovery like in the termination protocols.

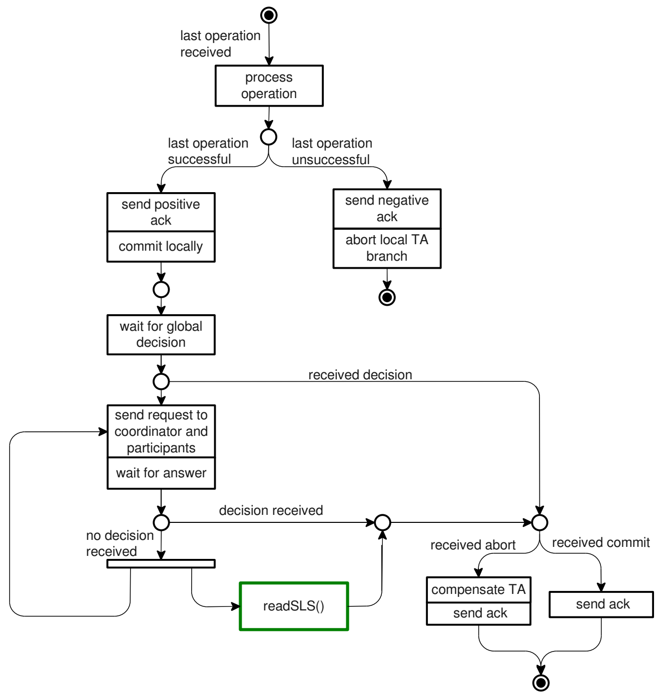


Figure 5.6: Participant termination protocol with SLS in the semantic transaction model.

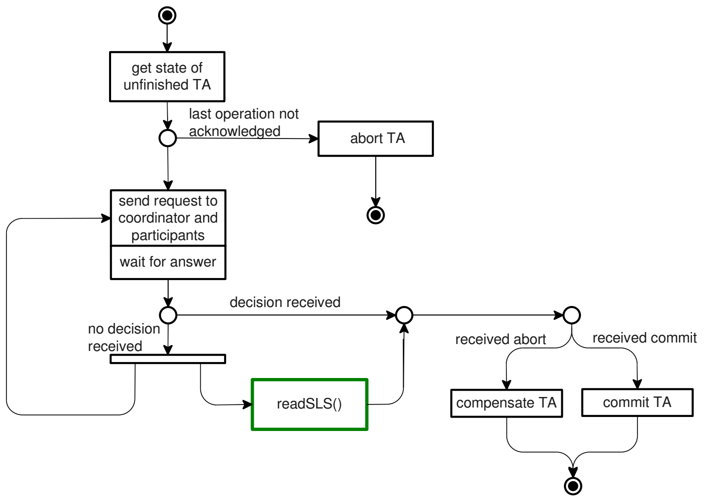


Figure 5.7: Participant restart protocol with SLS in the semantic model.

5.4 SLS Lightweight Approach

While the previous sections described the abstractions of the SLS and the integration of its operations in termination and restart protocols of strict and semantic transactions, this section introduces issues of the lightweight implementation approach. The basic idea of the approach is to place the decision log on a small set of assistant nodes that provides the desired log availability P_{ret} for time t_m. The smaller this node set, the more message efficient is the preservation process. The lightweight approach does not consider the topological structure of the network and basically works on a "fire-and-forget" basis, i.e. the decision log is placed on assistants and after a sufficient availability is reached, the availability of the decision log is no longer maintained. In contrast, dissemination and retrieval approaches can also be built on top of a topology-aware overlay structure, e.g. a cluster overlay, where the cluster-head is responsible for maintaining availability of a decision log in its cluster. I describe such an implementation in Section 5.8.

The main objective of the lightweight approach is to discover a small set of assistant nodes, providing the demanded availability, and place logs on them in an efficient way. Therefore, four main problems have to be addressed: (i) dissemination and retrieval strategies are required for a message efficient placement of the decision log; (ii) assistant nodes have to be discovered; (iii) the short-term reachability of assistant nodes has to be evaluated to address the problem of plan aging; and (iv) a mathematical model to calculate the log availability achieved by an assistant set is required. This section addresses problems (i)–(iii), while (iv) is discussed within the next section.

5.4.1 Dissemination and Retrieval Strategies

Generally, I distinguish two major approaches of log placement in the lightweight approach: (i) controlled dissemination where the exact set of assistant nodes is known; and (ii) an uncontrolled dissemination where the dissemination process has no exact knowledge on which assistant nodes the log is placed on.

The most message efficient combination of log dissemination and log retrieval is a controlled dissemination scheme and a retrieval process which has exact knowledge of the assistant set. I already mentioned this scheme in the context of the *register* operation in Section 5.2, where one intention of the *register* method is to define the assistant set (plan k) before any participants move into uncertainty, and to let the *readSLS* operation know k. However, this scheme does not make sense in all scenarios, e.g. if no multi-hop routing is used, then knowledge of k is of less benefit, as the retrieval process cannot directly send a message to these nodes.

A controlled dissemination strategy requires acknowledgments of assistants that saved the log to be collected, to allow the coordinator to decide on the achieved log availability. The decision log can be placed either using direct messages or by broadcasting the log. If the log is directly placed on nodes defined by k the dissemination is called *directed*, while I call the dissemination process *undirected* if the decision log is broadcasted until enough nodes acknowledged receipt of the decision log.

While directed as well as undirected dissemination requires time to collect acknowledgments of assistants that received the decision log, an uncontrolled dissemination based on a flooding scheme requires only the time to initiate flooding. This requires only one broadcast message. In the following, the dissemination schemes used for the SLS implementation are described.

5.4.1.1 Directed Dissemination

Directed dissemination allows for the most message efficient dissemination and retrieval process. If the log is placed on $\#k$ individual nodes known by the *readSLS* operation, the maximum number of messages required to disseminate and retrieve the decision log is $4\#k$. Note that I use the $\#$ to denote the cardinality of a node set, i.e. $\#k$ is cardinality of set k. Since placement and retrieval is done using directed messages, this scheme can only be applied in scenarios with multi-hop routing. A major disadvantage of the directed dissemination strategy is that reliable assistant nodes have to be discovered and defined within the *register* method, which requires additional message overhead as I will show in Subsection 5.4.2. Additionally the short-term reachability of assistants has to be considered to deal with the problem of plan aging as described in Subsection 5.3.

I consider the directed strategy as especially important in the real world, as this strategy allows to exploit individual failure probabilities of assistant nodes, i.e. few very reliable nodes can be chosen as assistants to achieve the desired availability. For example, assume that in the disaster scenario of Section 4.1 some command trucks are placed in certain positions in the disaster area. These trucks have a very low probability of node failures, because they will remain within the site much longer than other rescue resources and will not fail due to exhausted batteries, as they are possibly equipped with a diesel generator. Considering such individual failure probabilities is a main key to achieve high log availability at minimum cost, as I will show later.

Although aging of k can be considered, it may be possible that not all assistants defined in k are reachable at dissemination time. In such a case, a fall-back to undirected dissemination or flooding is possible.

5.4.1.2 Undirected Dissemination

If the set of assistant nodes is not defined before the *writeSLS* operation is executed, an undirected dissemination strategy is initiated by the *writeSLS* operation. Distribution of the decision log is undirected in the sense that assistant nodes are not defined a priori, but the decision log is broadcasted and all nodes receiving the log store it and acknowledge its preservation to the issuing node. Hence, the process of assistant discovery and log placement is combined in one message round. This scheme is especially feasible in scenarios where no multi-hop routing is used, as here a direct placement on defined nodes is not possible. However, the scheme can obviously be also applied in multi-hop scenarios.

To disseminate the decision log in an undirected manner, it is broadcasted in increasing hop ranges using a broadcast-in-space scheme. Acknowledgments of nodes that received the decision log are used to decide when sufficient log availability is reached and distribution of the decision log can be stopped.

As acknowledgment messages of assistant nodes can contain information about the individual node failure probability of the node that stores the decision log, individual nodes showing high reliability can be exploited to achieve the desired log availability with a small assistant set. However, this cannot be planned as with directed dissemination.

While in a scenario where multi-hop routing is available, an assistant node can send its acknowledgment message directly to the coordinator node; in scenarios where no multi-hop routing is used this is not possible. Here, the acknowledgment has to be broadcasted in the same hop distance as the decision log to assure that the node executing the *writeSLS* operation receives the acknowledgment message. This results in a high message load for large hop ranges. Thus, depending on the hop diameter of $\mathcal{A}$, from a certain broadcast range, a flooding scheme (uncontrolled dissemination) as described in the following subsection is feasible.

Since with undirected dissemination the retrieval process has no knowledge on which node the decision log resides, the *readSLS* operation cannot address an assistant node directly, but a search request has to be distributed within $\mathcal{A}$ to discover an assistant node carrying the desired decision log. The hop distance the search request is broadcasted depends on the availability calculation used at dissemination time. The calculation of log availability considers a path probability P_{path} as I will describe later. The path probability either describes the probability of a path with arbitrary hop distance, or a path with a bounded distance, e.g. 1–2 hops denoted by $P_{path,1-2hop}$. For example, if the path probability of a 1–2 hop path is used in availability calculations, then the log will be distributed to enough assistant nodes to assure that a recovering participant will find an assistant in its 1–2 hop neighborhood at P_{ret}. Hence, the hop radius used for searching an assistant, depends on P_{path} used in availability calculations.

In case multi-hop routing is available, I use the path probability for arbitrary long paths. The retrieval process then broadcasts the search request within the complete network and assistant nodes holding the decision log can send the log directly to the requesting node. If no multi-hop routing is used, $P_{path,1-2hop}$ should be used, as an assistant node cannot directly send the decision log to the requesting node, but the decision log has to be broadcasted in the same hop distance as the search request of the *readSLS* operation.

If the undirected dissemination schemes cannot place the decision log on enough assistant nodes within certain time bounds, a fallback to a dissemination based on flooding as described in the following is always possible.

5.4.1.3 Flooding

If the node executing the *writeSLS* operation resides in a small partition, directed and undirected dissemination can probably not reach enough assistant nodes to provide the desired availability of the decision log within Δ_w. If the distributing node suffers node failure before a sufficient log availability is achieved the whole SLS approach is failed. The major advantage of a flooding scheme in such a situation is that the *writeSLS* operation can terminate right after the flooding protocol is initiated. Even if the coordinating node suffers a node failure, the log dissemination will continue.

Using a broadcast-in-time protocol allows to eventually distribute a log item to all nodes of the network, but possibly requires some time to visit all nodes (see Section 2.1.4 for an introduction to broadcast schemes in MANETs). Log dissemination is uncontrolled, as the node that initiated log distribution has no knowledge which nodes are holding the log. Additionally, acknowledgments as used in the directed and undirected scheme are not feasible here, as this results in multiple flooding of the network.

The basic guarantee given by a broadcast-in-time scheme is that all nodes in the network are eventually visited. If the total number of nodes (or the average number of nodes) within the network $n_{\mathcal{A}}$ is known, then the eventually achieved availability of the decision log can be calculated (this calculation is presented in Section 5.6.1.1).

The disadvantage of this log dissemination strategy is that all nodes of the network are contacted and therefore much more assistant nodes than actually required are used, resulting in an unnecessarily high message load.

A major problem here is to decide when the initiated distribution should be stopped. There is no flooding approach that allows to terminate the dissemination process after a certain number of nodes have been reached, since every node has only a local view on the dissemination process. As I assume in my system model that new nodes enter $\mathcal{A}$, the distribution of the decision log will never stop, because of new nodes causing other nodes to rebroadcast the log. A safe way would be to terminate the distribution at an upper time limit, which is associated with a decision log. The mission time t_m is a conservative but safe time limit at which to stop the distribution of the log item. However, additional research is required to design a partition-aware flooding mechanism that terminates after reaching a certain number of nodes.

As long as such schemes are not available, I propose to use any partition-aware broadcast strategy or an integrated approach such as hypergossiping with an upper time limit for data distribution given by t_m.

For a concrete implementation of the SLS, I proposed a simple probabilistic flooding protocol based on a push-based strategy that broadcasts decision logs within random intervals to prevent broadcast storms. A decision log is not recognized for further rebroadcasts if it has reached a maximum broadcast number. The maximum number of broadcasts is approximated using t_m and the expected time required to execute a certain number of broadcasts. If a node receives a decision log, it is saved and a timer initiating the next broadcast is scheduled if not already pending.

If a decision log has been flooded in $\mathcal{A}$, a recovering node executing the *readSLS* operation can most likely discover the log in its direct neighborhood, i.e. in 1–2 hop distances using the same retrieval process as with undirected dissemination.

5.4.2 Discovery of Assistants

In the directed dissemination scheme potential assistant nodes have to be discovered before a writeSLS operation is possibly executed. The objective of the discovery process is to locate nodes that will be reachable for the coordinator in case the *writeSLS* operation is executed. This is done either by broadcasting a discovery message or by looking up a local routing table, e.g. as provided by AODV or DSDV routing protocols to identify neighbors in direct vicinity.

5.4.2.1 Using a Discovery Message

An implementation of the approach using a discovery message is straightforward: a so-called SLS assistant request message (SARQ) is broadcasted and every node receiving such a packet answers with a so-called SLS assistant reply message (SARP). The set of discovered assistant nodes, called k_{dis}, is then given by all nodes from which a SARP message was received. A SARQ message contains three fields to control the broadcast range of the SARQ message: the MAX_HOPS, CUR_HOPS, and an ID field. A node receiving a SARQ message rebroadcasts the message only if it has not already rebroadcasted a message with the same ID and if CUR_HOPS $<$ MAX_HOPS. Before the message is rebroadcasted, the CUR_HOPS value is increased, while initially CUR_HOPS=1.

A major advantage of assistant discovery using SARQ messages is that context information such as movement direction, battery level, or individual node failure probabilities can be transported within the SARP message. Such context information is used (i) to rank the nodes in k_{dis} according to their short-term reachability to identify the assistants that will most likely be reachable when the *writeSLS* operation is executed; and (ii) to calculate the long-term reachability of an assistant node required to derive $LA(k,t)$.

Calculations for (i) and will be presented in Subsection 5.4.3, while (ii) is addressed in Section 5.5.

5.4.2.2 Using Local Routing Information

If topology-based multi-hop routing is used, neighbor discovery is also done on the routing layer. For example, in AODV every node periodically broadcasts HELLO messages to announce its presence to its current vicinity. Another scheme is to use Link Layer feedback from the MAC protocol as provided in 802.11 to sense the presence of nodes in direct radio range, without issuing any messages. Independently of how the routing scheme discovers neighbors, the routing table can be used to obtain k_{dis} without extra message cost at the application layer. A lookup in an AODV routing table reveals all nodes that are direct neighbors, while in multiple-hop distances, not all available nodes are necessarily listed (see Section 2.1.3).

In Chapter 2, I presented the routing information provided by AODV in Table 2.1(a). In contrast to the scheme where SARP messages can be used to transport context information, the routing table does not provide any further context information about assistant nodes besides hop count and the age of a routing entry. Here, an integrated cross-layer approach is required to transport context information within node discovery of the routing layer.

5.4.3 Evaluate Short-Term Reachability of Assistants

To consider the problem of plan aging in the directed dissemination scheme, the short-term reachability of a discovered assistant has to be taken into account. The short-term reachability is the probability that a node discovered and selected within the *register* operation can actually be reached to preserve the decision log when the *writeSLS* method is executed. One probability is to use $F_C(t)$ as given by the system model to evaluate the short-term accessibility of an assistant node.

If additional information about the movement of assistants is available, like position, speed, and direction, more accurate predictions for reachability of assistants for the coordinator can be derived within a short time horizon. Such movement information can be transported within SARP messages as described above. In the following, I will describe one approach based on the Link Expiration Time (LET) to estimate the probability that a direct link of the coordinator with an assistant node fails.

The LET can be used to rank assistant nodes according to their short-term reachability. The LET was originally presented in [139] and is given in Formula (5.1). I use the LET to derive a probability of a direct communication link after time t between the coordinator i and an assistant node j initially at positions (x_i, y_i), (x_j, y_j) and moving with velocities v_i, v_j in directions θ_i and θ_j. The LET denoted as δ_{let} is not a probability but the time the direct connection between two nodes holds, if speed and direction of movement is retained.

$$\delta_{let} = \frac{-(ab + cd) + \sqrt{(a^2 + c^2)r^2 - (ad - bc)^2}}{a^2 + c^2} \tag{5.1}$$

with:

$$\begin{aligned} a &= v_i cos\theta_i - v_j cos\theta_j \\ b &= x_i - x_j \\ c &= v_i sin\theta_i - v_j sin\theta_j \\ d &= y_i - y_j \end{aligned} \tag{5.2}$$

Note that calculation of δ_{let} assumes the simple Free Space radio propagation model [121]. The main idea I propose is to interpret δ_{let} as a probability variable called D_{let}, which is subject to a certain dispersion. Additionally, the time $t_{ter,c}$ when the dissemination plan k is executed is also considered to be a random variable called T_{dis} spreading around the expected value $t_{ter,c}$.

The desired probability that an assistant node is reachable at dissemination time is then given by the probability

$$P(D_{let} < T_{dis}) = P(D_{let} - T_{dis} < 0) = P(X < 0) \tag{5.3}$$

with $X = D_{let} - T_{dis}$. As D_{let} and T_{dis} are subject to numerous (mostly unknown) influences, a normal distribution of D_{let} and T_{dis} with parameter $\mu_{let} = E(D_{let}) = \delta_{let}$, ${\sigma^2}_{let} = Var(D_{let})$, $\mu_{dis} = E(D_{dis}) = (t_p + U_{max})$ and $\sigma^2_{dis} = Var(D_{dis})$ is assumed based on the central limit theorem. As D_{let} and T_{dis} are normal distributed, X is also normal distributed with parameters $\mu_X = E(X) = \mu_{let} - \mu_{dis}$ and ${\sigma^2}_X = Var(X) = {\sigma^2}_{let} + {\sigma^2}_{dis}$.

The probability for the event $D_{let} < T_{dis}$ is now given by $F_X(X = 0)$:

$$P(X < 0) = F_X(X = 0) = \frac{1}{\sigma_X \sqrt{2\pi}} \int_{-\infty}^{0} e^{-\frac{1}{2}(\frac{t-\mu_X}{\sigma_X})^2} dt \tag{5.4}$$

With the transformation $z = \frac{t-\mu_X}{\sigma_X}$ the formula to calculate the probability that an assistant node is in direct connection range at time t_{dis} is then given by:

$$P = F_X(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\mu_X}{\sigma_X}} e^{-\frac{1}{2}z^2} dz \tag{5.5}$$

Formula (5.5) requires μ_X and σ_X as input. While μ_X is derived from δ_{let} and t_{dis}, σ_X is not directly given and hence, has to be derived by simulation or other measurements for a specific MANET scenario. However, calculation of σ_{let} or σ_{dis} is a statistical standard method and therefore not presented here. For a more detailed presentation of the approach including calculations and simulation results, I refer to [142, 31].

As multi-hop routing is assumed to be available if direct dissemination is used, the failure of a direct link does not necessarily mean that communication between interaction partners fails, since an alternative multi-hop route is possibly available. However, $F_X(t)$ is a good indicator for the real short-term reachability as shown in [142] and therefore useful to rank assistant nodes according to their reachability at $t_{ter,c}$.

The main disadvantage of this approach is that position information must be available and that only the Free Space radio propagation model is considered, which is not realistic in urban areas.

5.4.4 Summary - SLS Implementation

In this section, I presented the basic idea of the lightweight implementation approach of the SLS and introduced some problems to be solved by an implementation. A major issue is the efficient dissemination of the decision to assistant nodes. Dissemination can happen either in a controlled manner if the coordinator tracks the assistant nodes that received the decision log or uncontrolled if the decision log is disseminated within the network while the coordinator does not receive any feedback which nodes have saved the decision log. Controlled dissemination is either directed, if a dissemination plan is given or undirected if assistant nodes are discovered first when the *writeSLS* operation is executed. Uncontrolled dissemination is mainly considered as a fallback if a controlled dissemination scheme does not reach enough participants within time Δ_w.

While for the discovery of assistant nodes, a cross-layer approach would be most feasible, failure probabilities of assistant nodes can yet be only acquired using a message-based discovery scheme. For the implementation of the SLS I have proposed such a scheme that allows to transport information used for evaluating the short-term reachability of discovered nodes. The short-term reachability can be derived by considering position and speed information and calculating the probability that a direct link exceeds before the *writeSLS* operation is executed. This scheme is published in [31].

However, the major problem to be solved by the presented approach is to calculate the availability of the decision log achieved by an assistant set. The calculation model to estimate such availability is presented in the following section.

5.5 Log Availability Model

While the short-term reachability describes the probability that an assistant node defined in k can be reached by the coordinator when the *writeSLS* operation is executed, the long-term reachability describes the probability that an assistant can be reached by a recovering participant, i.e. at the time the *readSLS* operation is executed. This probability is called *long-term,* because its prediction horizon is t_m and hence, reaches further into the future. The long-term reachability is required to derive the probability of a restarting participant to reach an assistant that holds the decision log. It is composed of two parts: (i) the probability that at recovery time t_{res} and $t_{ter,p}$ respectively, the assistant is present in $\mathcal{A}$; and (ii) the probability of a communication path to be existent between the recovering transaction participant and the assistant node.

By computing the probability of presence in $\mathcal{A}$ and P_{path}, the long-term reachability can be derived for an assistant node. This allows to calculate the log availability $LA(k,t)$ for a given k and therefore to evaluate whether a realization of k meets P_{ret}.

Calculations of the probability of node presence will be presented for different scenarios. First, I will assume common node failure probabilities for all nodes in $\mathcal{A}$, which is a significant simplification. These basic results are then enhanced to consider individual node failure probabilities and recovery from node failures. In the real world, nodes show individual probabilities of node failures. E.g. in the disaster scenario, it can be assumed that some rescue units remain in the disaster area longer than others, e.g. a truck dispatching supplies will not remain as long in $\mathcal{A}$ as a command unit and will therefore show a lower probability of presence in $\mathcal{A}$.

In the following, I will first present how the probability of presence in $\mathcal{A}$ can be computed and how individual node failures probabilities can be derived. Afterwards, the calculation of path probabilities in $\mathcal{A}$ is discussed, and finally, formulae for $LA(k,t)$ are presented and applied to the example scenario of this work.

5.5.1 Node Presence in $\mathcal{A}$

In the system model of this work, the presence of an individual node in $\mathcal{A}$ depends on the probability of a node failure described by $F_N(t)$. I am interested here in the probability that an assistant node defined in k survives t_m in $\mathcal{A}$ and can be reached by a recovering participant. It is desirable to choose the nodes from k_{dis} for k which show the highest probability to survive t_m. This results in a small set of assistant nodes and therefore in a more message-efficient dissemination and retrieval process.

In the following, I will first consider common node failure probabilities and individual node failure risks afterwards. Finally, the log availability is calculated when assistant nodes are assumed to recover from node failures.

5.5.1.1 Common Node Failure Probabilities

In the simplest case, the same probability for disconnection from $\mathcal{A}$ is assumed for all assistant nodes. In this case, the probability that an assistant node is present at time t is directly given by $F_N(t)$, as defined by the system model. This is a major simplification, because in the real world, nodes that disconnect from $\mathcal{A}$ may reconnect later. This is reflected in the system model by the rejoin probability $F_J(t)$. As the probability of presence is of interest for long periods (i.e. the mission time), it is likely that assistant nodes experience multiple disconnection and reconnection cycles.

If a common node failure probability $F_N(t)$ is assumed, the probability that at least l of $\#k$ assistant nodes are present at time t is given by $P_{pr}(t)$.

$$P_{pr}(l,t) = \sum_{i=0}^{\#k-l} \binom{\#k}{i} \cdot F_N(t)^i \cdot \left[1 - F_N(t)\right]^{\#k-i} \tag{5.6}$$

5.5.1.2 Individual Node Failure Probabilities

To consider individual node failure probabilities, the system model is enhanced by assuming an individual node failure pdf $f_{i,N}(t)$ and reconnection pdf $f_{i,J}(t)$ for a node i. In this case, I assume that nodes show some advanced technical configurations to learn about their individual failure and reconnection probabilities and thus are able to communicate their individual failure characteristics within a SARP message.

One approach to let nodes learn their individual node failure probability is presented in Section 5.5.2. The probability of presence for such an assistant node is then directly given by $F_{i,N}(t)$ if recovery is not considered. To calculate the probability of at least l of $\#k$ assistant nodes with individual node failure probabilities to be present at time t, all possible combinations of assistant nodes to survive t have to be considered.

To identify subsets of k, I use the following notation: I denote the j-th subset of k with i elements by $k_{i,j}$ and define $k_{0,j}$=k. The probability of all nodes of this subset to experience node failure until t is called $F_{N,k_{i,j}}(t)$. By $F_{k \setminus k_{i,j}}(t_t)$ I denote the probability that all nodes in the set difference $k \setminus k_{i,j}$ disconnect from $\mathcal{A}$ until t. The probability that all nodes of the j-th subset with i elements suffer a failure until t, is given by the product of the individual failure probabilities $F_{N,k_{i,j}}(t) = \prod_{m=1}^{i} F_{N,k_{i,j,m}}(t)$, where $F_{N,k_{i,j,m}}(t)$ is the probability of a node failure of the m-th element of $k_{i,j}$.

The probability that l of $\#k$ nodes with individual node failure probabilities are present in $\mathcal{A}$ at t is now given by $P_{pr,ind}(l,t)$.

$$P_{pr,in}(l,t) = \sum_{i=0}^{\#k-l} \sum_{j=1}^{\binom{\#k}{i}} F_{N,k_{i,j}}(t) \cdot \left[1 - F_{N,k \setminus k_{i,j}}(t)\right] \tag{5.7}$$

5.5.1.3 Recovery from Node Failures

If recovery from node failures is considered, each node i experiences a sequence of failures and recovery events. The probability to be calculated here is the probability that an assistant node is present in $\mathcal{A}$ at time t. This probability is described by $F_N(t)$ and $F_J(t)$ and can be modeled using a Markov chain, if common failure and recovery probabilities are assumed.

To model the failure-and-recovery process as a Markov chain, the failure and recovery pdfs $f_N(t)$ and $f_J(t)$ must fulfill the Markov property. The Markov property states that given a system is in state i at time t, the probability for a future state does not depend on any previous states[1]. Hence, the following calculations only apply to node failure and rejoin pdfs that fulfill this property. Exponentially distributed $F_N(t)$ and $F_J(t)$ as assumed by the system model of

[1]Note that here I consider a first-order Markov Chain only.

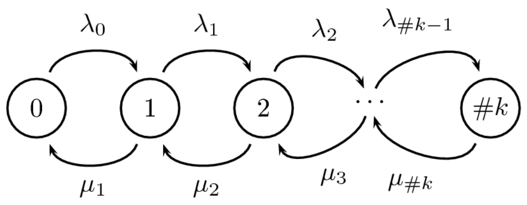


Figure 5.8: Transition diagram of $\#k$-component system.

this work fulfill this property. Note that $F_N(t)$ is exponentially distributed if energy related failures are considered to be exponentially distributed as described in Section 4.1.1.

In the following, I will not consider an individual assistant node, but the probability that at least l of $\#k$ nodes are present at time t. To model this probability by a stochastic process, the state-space of the Markov chain is given by the cardinality of k. Figure 5.8 depicts the transition diagram of the modeled process with $\#k$ assistant nodes.

Initially, the system is in state 0 and all $\#k$ assistant nodes are connected to $\mathcal{A}$ (no assistant has suffered node failure yet). With proceeding time, assistant nodes can fail and reconnect to $\mathcal{A}$. A node failure of an assistant node in state i brings the system into state *i+1*. For example, state 4 implies that 4 of $\#k$ nodes are not present in $\mathcal{A}$. A recovery event brings the system into the next lower state, i.e. if the system is in state 5 and an assistant node reconnects to $\mathcal{A}$, the process is taken from state 5 in state 4.

The probability that at least one assistant is connected to $\mathcal{A}$ at time t is given by the probability that the according Markov process is not in state $\#k$ at time t. This probability is given by $1 - P_{\#k}(t)$. Hence, with probability $P_{\#k}(t)$, recovery from blocking or extended uncertainty is not possible at t, because no assistant is present in $\mathcal{A}$ that holds the decision log. In the following, the state probabilities of the process are derived.

In the case of identical independently distributed failures and recovery probabilities of the assistant nodes, the transition rates λ_i and μ_i of the Markov chain are given by:

$$\begin{aligned} \lambda_i &= (\#k - i) \cdot \lambda \\ \mu_i &= i \cdot \mu \end{aligned} \tag{5.8}$$

In the following, I call λ_i and μ_i transition rates. λ_i is derived by the following consideration: for an infinitesimally small time step Δt, the probability of a single assistant to experience a node failure is given by λ. Because all i remaining assistants show the same failure probability, the probability that one of i nodes suffers failure[2] within Δt is given by $i\lambda$. μ_i is derived analogously. The probability of transition from state i to state j within time interval Δt is described by the probability $P(X(t + \Delta t) = j|X(t) = i)$, which I denote by

[2]Note that it is assumed that within Δt only one node can fail. Simultaneous nodes failures in Δt are not allowed.

$P_{ij}(\Delta t)$, where X is a random variable that takes the possible states as values. The transition probabilities are then defined as:

$$a_{ij} = \lim_{\Delta t \to 0} \frac{P(X(t+\Delta t) = j | X(t) = i)}{\Delta t} = \lim_{\Delta t \to 0} \frac{P_{ij}(\Delta t)}{\Delta t} = \dot{P}_{ij}(0) \tag{5.9}$$

In the following, the time derivative is denoted by $\dot{P}_{ij}(t) = \frac{d}{dt} P_{ij}(t)$. Using the Chapman-Kolmogorov equation, it can be derived that

$$\dot{P}_{ij}(t) = -P_{ij}(t) \cdot \sum_{\substack{k=0 \\ k \neq j}}^{\#k} a_{jk} + \sum_{\substack{k=0 \\ k \neq j}}^{\#k} P_{ik}(t) \cdot a_{jk} \tag{5.10}$$

For a detailed description of the steps to derive Formula (5.10) I refer to [84, 85]. As the initial state of the process is known to be 0 at t=0, Formula (5.10) can be simplified by omitting the index i, which results in:

$$\begin{aligned} \dot{P}_j(t) &= -P_j(t) \cdot \sum_{\substack{k=0 \\ k \neq j}}^{\#k} a_{jk} + \sum_{\substack{k=0 \\ k \neq j}}^{\#k} P_k(t) \cdot a_{jk} \\ P_1(0) &= 1, \quad P_k(0) = 0 \;\; for\, k \neq i \end{aligned} \tag{5.11}$$

The state equation of Formula (5.11) can be written as

$$A \cdot P(t) = \dot{P}(t) \tag{5.12}$$

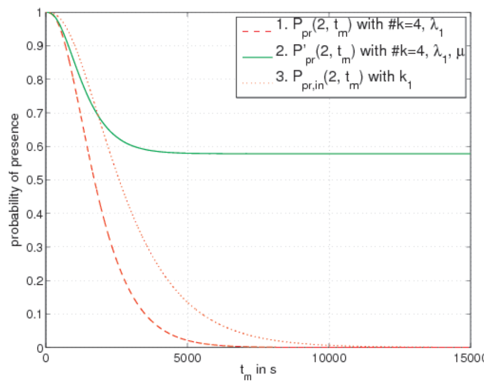
with

$$A = \begin{bmatrix} -a_{00} & a_{10} & a_{20} & \dots & a_{\#k0} \\ a_{01} & -a_{11} & a_{21} & \dots & a_{\#k1} \\ a_{02} & a_{12} & -a_{22} & \dots & a_{\#k2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_{0\#k} & a_{1\#k} & a_{2\#k} & \dots & -a_{\#k\#k} \end{bmatrix},$$

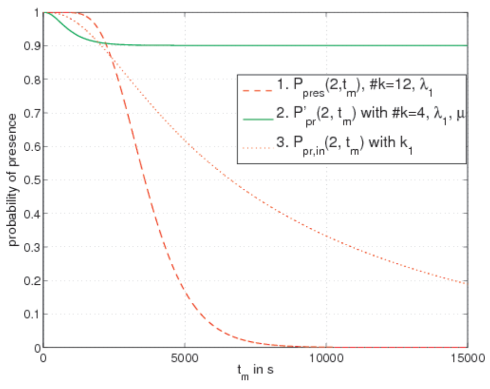
$$P(t) = \begin{bmatrix} P_0(t) \\ P_1(t) \\ P_2(t) \\ \vdots \\ P_{\#k}(t) \end{bmatrix} \quad and\, \dot{P}(t) = \begin{bmatrix} \dot{P}_0(t) \\ \dot{P}_1(t) \\ \dot{P}_2(t) \\ \vdots \\ \dot{P}_{\#k}(t) \end{bmatrix} \tag{5.13}$$

The transition matrix for the Markov chain modeled here is given by A_{SLS}:

$$A_{SLS} = \begin{bmatrix} -\lambda\#k & \mu & 0 & 0 & \cdots & 0 \\ \lambda\#k & -(\lambda(\#k-1)+\mu) & 2\mu & 0 & \cdots & 0 \\ 0 & \lambda(\#k-1) & -(\lambda(\#k-2)+2\mu) & 3\mu & \cdots & 0 \\ 0 & 0 & \lambda(\#k-2) & & \cdots & \vdots \\ 0 & 0 & 0 & \ddots & (\#k-1)\mu & 0 \\ \vdots & \vdots & \vdots & 2\lambda & -(\lambda+(\#k-1)\mu) & \#k\mu \\ 0 & 0 & 0 & 0 & \lambda & -\#k\mu \end{bmatrix} \tag{5.14}$$



(a) Presence probability of at least l=2 assistant nodes of k (#k=4). $\lambda_1 = 1800^{-1}$, $\lambda_2 = 18000^{-1}$, $\mu = 5400^{-1}$, and $k_1 = \{\lambda_1, \lambda_1, \lambda_1, \lambda_2\}$.



(b) Presence probability of at least l=2 assistant nodes of k. $\lambda_1 = 1800^{-1}$, $\lambda_2 = 18000^{-1}$, $\mu = 900^{-1}$, and $k_1 = \{\lambda_1, \lambda_1, \lambda_2, \lambda_2\}$.

Figure 5.9: Probability of assistant presence in $\mathcal{A}$ for the example MANET scenario of this work.

Using: (i) the fact that $\sum_{j=0}^{\#k} P_j(t) = 1$; (ii) the known initial states given in (5.11); and (iii) the system of differential equations given by

$$A_{SLS} \cdot P(t) = \dot{P}(t) \tag{5.15}$$

The state probabilities $P_j(t)$ for $j = 0, 1, 2, \ldots, \#k$ can be calculated by solving this system. For the sake of clarity a detailed description of how to solve this system of linear differential equations is omitted here.

In the context of the SLS, I am interested in the probability that one or l nodes are present in $\mathcal{A}$ at any time during mission time t_m. Given the calculations above, the probability that at least one node from k is present in $\mathcal{A}$ at t_m is given by $1 - P_{\#k}(t_m)$.

The probability that at least l of $\#k$ nodes are present in $\mathcal{A}$ calculates as the complementary probability that the system is in none of the states where more than $\#k - l$ nodes are not present in $\mathcal{A}$, given by $P'_{pr}(l, t)$.

$$P'_{pr}(l, t) = 1 - \sum_{i=0}^{l} P_{(\#k+1-i)}(t) \tag{5.16}$$

5.5.1.4 Discussion and Conclusion

In the following, calculations developed above are applied to the example scenario of this work defined in Chapter 4. The main objective here is to show the dimensions of achieved node presence and to demonstrate the influence of the model enhancements: (i) consideration of node recovery and (ii) taking individual node failure distributions into account. It has beed shown that especially (ii) allows for high presence probabilities with small $\#k$ in the example scenario, while (i) shows a stable probability of node presence for large mission times. I will first analyze the probability of presence with the standard node failure probabilities of the example scenario in Figure 5.9(a), before I vary $\#k$ as well as node and recovery probabilities.

In Figure 5.9(a), the probability of two assistant nodes out of four to be present at time t_m is plotted for $P_{pr}(t_m)$, $P_{pr,in}(t)$, and $P'_{pr}(t_m)$ for the example scenario of this work. If a common node failure probability (with $\lambda = 1800^{-1}$) is assumed, the probability that two assistants carrying the decision are present falls below 40 % after 33.3 min, as shown by Curve 1 in Figure 5.9(a).

If a particular stable node with $\lambda = 18000^{-1}$ is in k, while the three other nodes show an exponentially $F_N(t)$ with $\lambda = 1800^{-1}$ (plan k_1), the probability of at least two nodes to be present after 33 min increases to 65 % as depicted by Curve 3 in Figure 5.9(a). In the case where recovery from node failures is assumed, the probability of at least two nodes to be present in $\mathcal{A}$ is higher than 88 % for mission times smaller 33 min, if assistant nodes are expected to remain disconnected from $\mathcal{A}$ for 1.5 h. Additionally, the presence probability stabilizes around 68 % for large t_m, as shown by Curve 2 in Figure 5.9(a).

The results presented in Figure 5.9(a) show that a high probability of assistant presence (e.g. larger 90 %) can only be guaranteed for short periods, i.e. $t_m{<}800$ s. Hence, such a period is sufficient to read the decision within a termination protocol at $t_{ter,pa}$, but not necessarily at t_{res}. I therefore examine in Figure 5.9(b) how the presence of assistants increases for large t_m by (i) increasing $\#k$, (ii) including more stable nodes in k, and (iii) assuming longer sojourn times in $\mathcal{A}$ and shorter expected disconnection periods.

Increasing $\#k$ to 12 nodes provides a presence of two assistant nodes above 90 % for 40 min, as shown by Curve 1 in Figure 5.9(b). In the scenarios with low common node failures and a large $\#k$, the probability of at least two assistant nodes to be present in $\mathcal{A}$ decreases fast, e.g. after 2.2 h the presence probability of at least two assistants is close to zero.

In case two of the four assistant nodes are less susceptible to node failures, e.g. node failures are exponentially distributed with $\lambda = 5400^{-1}$, a probability of presence similar to the previous case with $\#k{=}12$ above 90 % is achieved for 40 min. However, the assistant presence decreases much slower as shown by Curve 3 in Figure 5.9(b) and is above 40 % for 2.2 h.

If recovery of node failures is assumed, decreased disconnection times allow to preserve the probability of two assistant nodes to be present continuously at levels above 90 % at $\#k{=}4$, as shown in Figure 5.9(b), where assistant nodes are expected to remain disconnected only for 0.25 h, instead of 1.5 h as in the previous situation.

The presentation above has shown that to provide a high probability of presence for large t_m, the most efficient approach is to consider recovery of assistant nodes as modeled in the AGB model.

In case recovery of assistant nodes is disregarded, the decision log dies out in $\mathcal{A}$ if not redistributed subsequently. An efficient scheme to increase the presence probability at reasonable cost and to delay die-off of the decision log is to choose especially stable nodes that will most likely remain in $\mathcal{A}$ for long time. For the example scenario, I showed that such a scheme achieved a higher probability of presence with a third of assistant nodes compared to the case of low common node failures probabilities. In other scenarios, the situation might be different and hence, calculations as presented here are required to predict the probability for a specified scenario.

Since considering individual node failure probabilities shows to be highly efficient, the following subsection presents an approach on how such probabilities can be derived in practice.

5.5.2 Deriving Individual Node Failure Rates

For calculations above, I simply assumed that assistants can provide individual pdfs $f_{i,N}(t)$. In this section, I show how this can be derived. The most decisive factor for node failures in the system model of this work is movement between clusters, i.e. transition of a node from $\mathcal{A}$ to another area $\mathcal{B}$ (see Section 2.3.1). The basic idea to derive an individual node failure probability is to let nodes measure their individual sojourn times within a cluster to derive $f_{i,L}(t)$ by statistical estimation. Therefore, nodes have to be able to sense disconnection and reconnection events in different clusters.

In the following, I briefly describe the idea and some simulation results of this approach. For a more detailed description of the implementation and additional simulation results, I refer to [118].

From a node's perspective, a random sample $\{x_1, x_2, \ldots, x_n\}$ of continuous stays (in seconds) within a certain cluster is derived over time. Assuming that an idea about the underlying parametric distribution is given, $f_L(t)$ and $f_J(t)$ can be approximated using statistical standard estimates. Additionally, the precision of the estimate, which depends on the size of the sample set, should be considered. As an exponential distribution with parameter λ is assumed for $f_{i,L}(t)$ and $f_{i,J}(t)$, the parameter estimate λ' can be obtained from a given sample by calculating the value with the *maximum likelihood*.

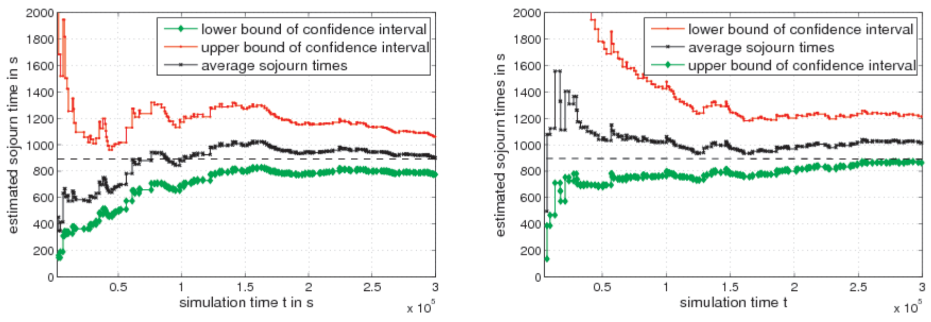
The estimation of parameter λ for the node recovery probability $f_{i,J}(t)$ is derived analogously using a sample of disconnection periods instead of sojourn times.

The point estimate described above provides no information about the precision of the estimated parameter. But for the calculation of $LA(k,t)$, the confidence of a node in its estimate should be considered. The confidence in an estimate depends on the number of samples used and the standard techniques to describe the reliability of estimates are confidence intervals. Confidence intervals enclose the desired parameter at a certain confidence level $1-\epsilon$, which means that with a probability of ϵ the parameter is not contained in the interval.

The bounds of the confidence interval provide lower and upper bounds for λ. As λ can be thought of as the disconnection rate of mobile nodes, the secure estimate is the upper bound λ'_i of the confidence interval. The real value of λ is with probability $(1-\epsilon)$ higher than λ'_i.

For the implementation of the SLS, I assume a constant confidence level of 95 %. By using the upper bound of the confidence interval, nodes with little confidence in their estimate will report a larger value for λ than nodes with higher confidence in their estimate and the same real λ. Such nodes will therefore be favored for k over nodes with less confidence in their estimate. If a node receives an SARQ message from a coordinator, it calculates its failure rate for the current area with ϵ=0.05 and returns the derived value within a SARP message to the requesting node.

To demonstrate the applicability of the approach, I present some simulation results in the following to show that nodes can derive a good approximation of λ after reasonable time in a MANET scenario. For two random nodes of the example MANET scenario, Figure 5.10 shows how the estimation develops over



(a) Estimates of a random node *a*. (b) Estimates of a random node *b*.

Figure 5.10: Estimation process of λ at 95 % confidence level and λ=900^{-1}.

time.

To simulate the estimation process, a mobility scenario with two clusters $\mathcal{A}$ and $\mathcal{B}$ according to the AGB model is used. Nodes remain in $\mathcal{A}$ and $\mathcal{B}$ for exponentially distributed sojourn times before transiting to the other area. The duration for which nodes remain within each area is exponentially distributed with parameters λ_A and λ_B.

The following behavior was implemented in the *ns2* network simulator: nodes check their positions every second and match it to the area they are currently in. In case a node senses that it has left a cluster, it stores a new sample with the measured sojourn time for the area it has left.

Figures 5.10(a) and 5.10(b) present the derived estimates over time for two randomly chosen nodes.

The black dashed line represents the real λ ($\lambda = 900^{-1}$), while the black curve is the point estimate, and the green and red curves are the limits of the confidence interval at a confidence level of 95 %. The green curve presents the upper bound λ'_i of the confidence used as conservative estimate for individual node failure rates.

In case of the node observed in Figure 5.10(b), a good approximation is reached after 5.5 h, while the node observed in Figure 5.10(a) derives less accurate estimation. However, the simulation proves that for the purpose of this work, a good estimation can be derived in reasonable time.

5.5.3 Path Probability

The probability of presence of an assistant node, as calculated in Section 5.5.1 is not sufficient to predict the probability of successful log retrieval. Partitioning of the MANET in $\mathcal{A}$ can prevent communication between the recovering participant and an assistant node at recovery time.

To predict the probability that an assistant node is actually reachable, the probability of successful communication between an assistant node and a recovering node is required. This probability is given by the path probability P_{path} as introduced in Chapter 2.

At the time of writing this thesis, there is no analytical approach available to estimate P_{path} for multi-hop communication paths. Such values have to be derived by simulation as I presented in Section 4.1.2. However, the probability of one and two hop paths can be approximated analytically as shown by Bettstetter in [17, 16]. Note that even if an analytical approach would exist to calculate P_{path} for 2+ hops in a geometric graph, communication in the real world would still suffer from imperfect routing schemes, i.e. although a path exists it is possible that it is not discovered by the routing algorithm.

One-hop path probability is especially of interest in scenarios where no multi-hop routing is available. Here, P_{path} is given by the probability that the Euclidean distance between two randomly chosen nodes does not exceed the radio range r_0 of nodes. In [17] the Euclidean distance between two randomly chosen nodes is interpreted as a random variable S. For nodes moving according to the RWP mobility model on a square of size $a \cdot b$ a pdf $f_S(s)$ can be derived as shown in [17]. In the following, I denote the path probability for a direct connection as $P_{path,1-hop}$. Note that the approach described does not require any information about the total number of nodes in $\mathcal{A}$.

If the total or average number of nodes $n_{\mathcal{A}}$ in $\mathcal{A}$ is known, the probability that two random nodes can establish a two-hop path can be approximated. I call this probability $P_{path,2-hop}$. The approach is given in [16] and is based on the idea to calculate the probability that a third node is in the intersection area of radio ranges r_0 of two other nodes that are in Euclidean distance S with $r_0 < S < 2r_0$.

Figure 5.11 plots analytical and simulative derived path probabilities for the example MANET scenario of this work for different square sizes of $\mathcal{A}$ with a constant number of 15 nodes. At a size of 1000 m * 1000 m, the probability for a single or two-hop communication path between two random nodes is only at 5 % as shown by Curve 3 of Figure 5.11.

For the example MANET scenario where a 500 m * 500 m area is assumed, the probability for a one or two-hop path is slightly higher at 20 %. The path probability for arbitrary hop paths is considerably higher if AODV routing is used, as shown by experimental results in Figure 5.11. The probability of a working communication path between two random nodes is then given by approximately 70 %.

5.5.4 Calculation of Log Availability

Given the probability of assistant presence and the probability for a communication path between two nodes, the availability of the decision log achieved by a dissemination plan k for a recovering participant can now be calculated. The log availability depends on the presence of assistants and that the recovering node can reach at least one of the assistants present in $\mathcal{A}$.

In the following, I will calculate the log availability achieved by a given plan if: (i) common node failure probabilities are assumed; (ii) individual node failure probabilities are given; and (iii) if recovery from node failures is considered. Afterwards I apply the proposed availability model to the example scenario of this work to demonstrate the log availability achieved for different assistant sets.

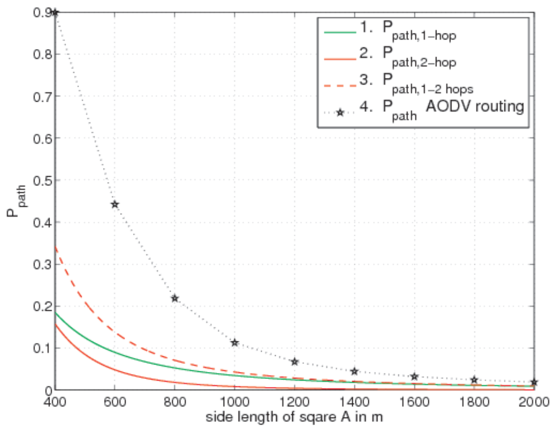


Figure 5.11: Approximations for P_{path} and experimental results for the example MANET scenario with $n_{\mathcal{A}}$=15 and r_0=100 m.

5.5.4.1 Common Node Failure Probabilities

In case a common node failure probability $f_N(t)$ for all nodes in $\mathcal{A}$ is assumed, the probability that l of $\#k$ assistant nodes are present at time t is calculated by $\binom{\#k}{l} \cdot F_N(t)^{\#k-l} \cdot [1 - F_N(t)]^l$. The decision log is available to the recovering participant if communication with at least one of the l present assistant nodes is successful. This probability is given by $PA(l) = 1 - (1 - P_{path})^l$. Thus, the log availability achieved by k denoted by $LA(k,t)$ is given by

$$LA(k,t_m) \quad = \quad \sum_{i=0}^{\#k-1} \binom{\#k}{i} \cdot F_N(t_m)^i \cdot \left[1 - F_N(t_m)\right]^{\#k-1-i} \cdot PA(\#k - i) \tag{5.17}$$

5.5.4.2 Individual Node Failure Probabilities

When assistant nodes in k can provide individual failure probabilities, the calculation of the log availability has to consider the probability of presence for all possible subsets of k, as calculated in Formula (5.7). For every set of assistant nodes present in $\mathcal{A}$, $PA(l)$ has to be considered. Hence, the availability of the decision log until time t_m is given by $LA_{in}(k,t_m)$, where the set notation as introduced in Section 5.5.1.2 is used.

$$LA_{in}(k,t_m) = \sum_{i=0}^{\#k-1} \sum_{j=1}^{\binom{\#k}{i}} F_{N,k_{i,j}}(t) \cdot \left[1 - F_{N,k \backslash k_{i,j}}(t)\right] \cdot PA(\#k - i) \tag{5.18}$$

5.5.4.3 Recovery from Node Failures

If recovery from node failures is considered, the probability that l assistant nodes are present in $\mathcal{A}$ at time t, is given by the state probability $P_{\#k-l}(t)$ of the Markov process presented in Section 5.5.1.3. Hence, the availability of the decision log for a recovering participant is given by $LA'(k,t_m)$.

$$LA'(k,t_m) = \sum_{i=0}^{\#k-1} P_i(t_m) \cdot PA(\#k-i) \tag{5.19}$$

5.5.4.4 Log Availability in the Example Scenario

In the following, I show how the log availability is influenced by the number of assistant nodes and by their node failure probabilities. The objective is to present the dimensions of P_{ret} and t_m that can be achieved by the lightweight approach in the example MANET scenario of this work. It will be shown that to achieve high values of P_{ret} for large t_m, either recovery from node failures has to be considered, or especially stable assistants have to be used. For scenarios where no multi-hop routing is used, the log availability achieved is low.

Log Availability with Common Node Failure Probabilities

Figure 5.12(a) depicts the log availability calculated by $LA(k,t_m)$ for different assistant sets if the same node failure probability is presumed for all assistants and a path probability of 0.7 is given, i.e. multi-hop routing is used. For an exponentially distributed $f_L(t)$ with $\lambda = 1800^{-1}$ as assumed for the example scenario, a plan with $\#k$=4 results in a log availability close to zero after 2.7 h, as shown by Curve 3 in Figure 5.12(a). In this setting, reasonable log availabilities larger than 80 % are only given for t_m<0.5 h.

To provide a high log availability for large t_m in the example scenario either a higher number of assistant nodes is required, or more stable assistant nodes are required, as depicted by Curves 1–2 and 4.

Curves 1 and 2 of Figure 5.12(a) depict the achieved log availability if the sojourn time of nodes in $\mathcal{A}$ is distributed exponentially with $\lambda = 5400^{-1}$, thus nodes are expected to remain connected to $\mathcal{A}$ for 1.5 h. In this case, a dissemination plan with $\#k$=4, provides an availability of the decision log larger than 80 % for 1.1 h, while a plan with $\#k$=15 achieves a log availability larger than 80 % for 2.7 h.

Curve 4 depicts the situation where the number of assistants is increased to 15 but node failures are still distributed with $\lambda = 1800^{-1}$. Here it showed that the log availability only slightly increases compared to a dissemination plan with $\#k$=4.

If no multi-hop routing is used, P_{path} is only 0.2 in the example MANET scenario, i.e. 1–2 hop paths are assumed. In this situation, the availability of the decision log is significantly lower as shown by Figure 5.12(b). If $\#k$=4 the log availability achieved is always smaller 60 %. In case of $\#k$=15, P_{ret}=0.8 is assured only for 0.4 h if node failure probability is high. With $\#k$=15 and $\lambda = 5400^{-1}$ a log availability larger 80 % is given for 1 h. Hence, if no multi-hop routing is used in the example scenario, a high P_{ret} can only be guaranteed for relatively short mission times.

Log Availability with Individual Node Failure Probabilities

The motivation to consider individual nodes failures is to exploit special stable nodes to assure a high P_{ret} with a small assistant set for large t_m.

Figure 5.12(c) shows the log availability for different plans denoted by k_1, k_2, and k_3. The node failure probabilities assumed by these plans are described in Figure 5.12(c). Curve 3 of Figure 5.12(c) confirms the feasibility of this scheme. Here, a plan with only two assistant nodes (plan k_3) provides a high availability of the decision log for large t_m, while one node is extra reliable with $\lambda_4 = 5 \cdot 10^{-5}$. For example, k_3 provides a log availability larger than 63.4 % for 13.8 h. In case assistant nodes are considered in k that are slightly more reliable than the common node failure probability, like in k_1, the achieved log availability is significantly increased, as observed by comparing Curve 1 in Figure 5.12(c) and Curve 3 in Figure 5.12(a). A low path probability in scenarios where no multi-hop routing is used results in a low log availability. Path probability is the main factor here, as shown by Curve 4.

Log Availability with Recovery from Node Failures

The log availability achieved in case recovery from node failures is assumed is depicted in Figure 5.12(d). High values of P_{ret} can be provided for large t_m, because the log availability does not converge to zero over time as in the situation where no recovery is considered. If nodes are expected to remain connected to $\mathcal{A}$ for long periods and disconnect only for short periods, log availability is nearly certain even with a small assistant set, as shown by Curve 2. For the example scenario of this work, a plan with four assistants provides a log availability larger 54 % for any t_m. This is depicted by Curve 1.

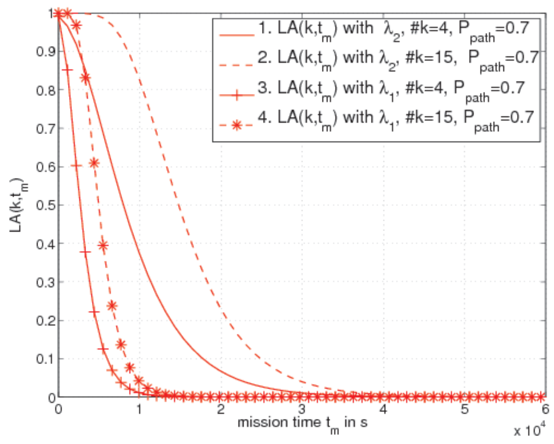
When no multi-hop routing is used in the example scenario, only a log availability of 34 % for large t_m can be provided, even in case the decision log is stored on all 15 nodes of $\mathcal{A}$ at dissemination time, as shown by Curve 3. If only four nodes are used as assistants, the predicted log availability is below 20 % for large t_m (see Curve 4).

5.5.5 Summary - Log Availability

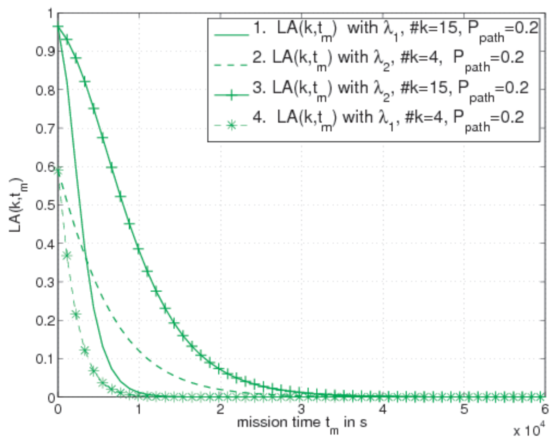
This section presented the calculation model to estimate the availability of decision logs to recovering participants in $\mathcal{A}$. Such calculations are a central contribution, since based on this probabilistic model the lightweight implementation approach of the SLS predicts and controls the availability of decision logs and therefore blocking risks.

The probabilistic model presented is based on the probability of node failures $F_N(t)$, probability of recovery from node failures $F_J(t)$, and on the path probability P_{path}. I showed how $F_N(t)$ and $F_J(t)$ can be derived by nodes autonomously in $\mathcal{A}$ over time and presented some analytical approximations for P_{path} based on the dimensions of $\mathcal{A}$ and $n_{\mathcal{A}}$. Note that P_{path} can also easily be obtained from the experiments proposed in Section 4.1.2 used to derive $F_C(t)$. Hence, I showed that the prerequisites of my calculation model can be established in practice.

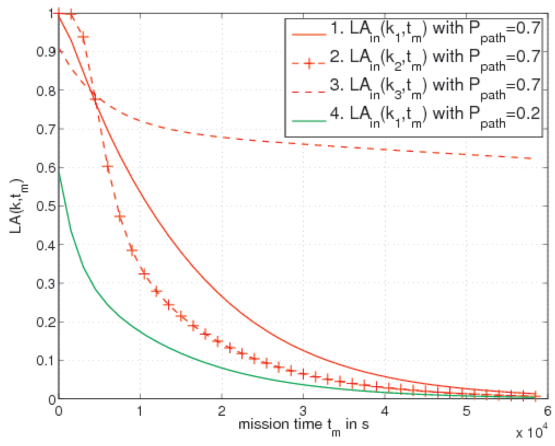
Within this model, I considered different scenarios: (i) a common node failure probability is known; (ii) individual node failure probabilities have been learned;



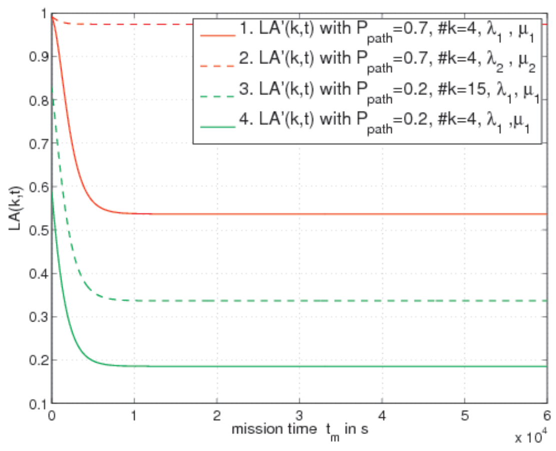
(a) Log availability if common node failure probabilities are assumed with $\lambda_1 = 1800^{-1}$, $\lambda_2 = 5400^{-1}$, and P_{path}=0.7 (multi-hop routing).



(b) Log availability if common node failure probabilities are assumed with $\lambda_1 = 1800^{-1}$, $\lambda_2 = 5400^{-1}$, and P_{path}=0.2 (no multi-hop routing).



(c) Log availability if individual node failure probabilities are assumed. k is given either by k_1={λ_1,λ_1,λ_3,λ_3}, k_2={λ_3,14*λ_1}, or k_3={λ_2,λ_4}, with $\lambda_1 = 1800^{-1}$, $\lambda_2 = 5400^{-1}$, $\lambda_3 = 12600^{-1}$ and $\lambda_4 = 500000^{-1}$.



(d) Log availability if recovery of nodes is considered. $\lambda_1 = 1800^{-1}$, $\lambda_2 = 5400^{-1}$, $\mu_1 = 5400^{-1}$, and $\mu_2 = 900^{-1}$.

Figure 5.12: Log availability in the example MANET scenario.

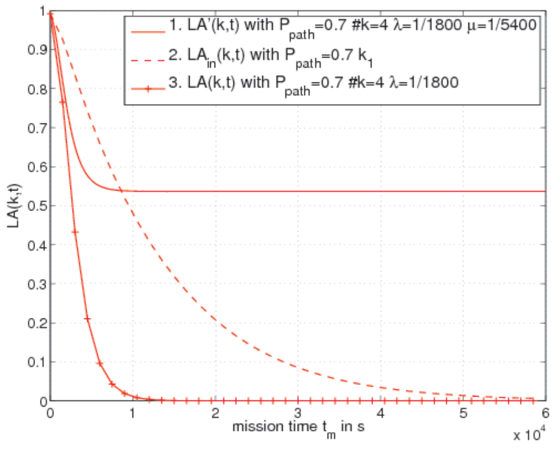


Figure 5.13: Comparison of calculation schemes. With k_1=$\{\lambda_1,\lambda_1,\lambda_3,\lambda_3\}$, λ_1=1800^{-1}, and $\lambda_3 = 12600^{-1}$.

and (iii) common node failure probabilities as well as recovery probabilities are known.

Figure 5.13 compares the log availabilities predicted for situations (i)–(iii) if four assistant nodes are used. While in (i) and (ii) the decision log is dying out over time, its availability in (iii) stabilizes at a constant level. Using extra reliable nodes as considered in (ii) is a promising approach to assure high log availabilities with small assistant sets.

5.6 Implementation of the SLS Operations

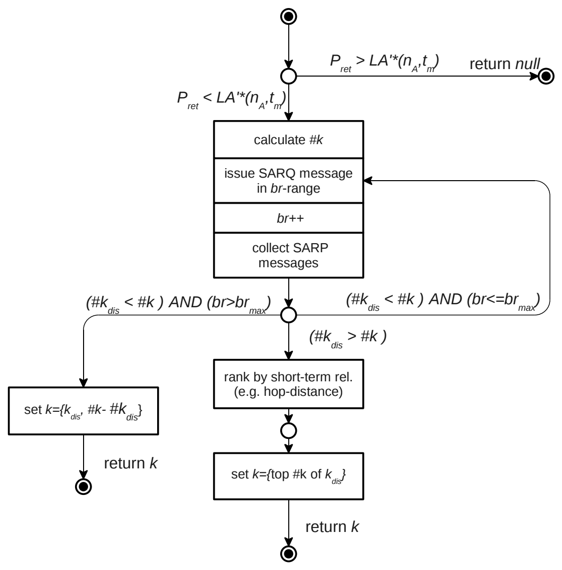
In the following, the implementation of the *register*, *writeSLS*, and *readSLS* operations for the lightweight approach is described. The implementation is based on the log availability and short-term reachability models introduced in the previous two sections.

The implementation is described for different variations of the system model assuming: (i) common or individual node and communication failure probabilities; and (ii) multi-hop or single hop communication within $\mathcal{A}$.

If multi-hop routing is used, message-efficient preservation and retrieval of the decision log is achieved by placing the log on a predefined set of assistant nodes known to transaction participants. Since assistant nodes can be addressed directly, recovering nodes can contact these nodes at low message complexity. If no routing is used, communication over multi-hop distance is expensive, as it is only possible by using broadcast schemes. Low values of P_{path} lead to a large $\#k$ and hence to a wide dissemination of the decision log.

5.6.1 Implementation of *register*

An implementation of the *register* method has to derive a dissemination plan k that provides availability of the decision log larger than P_{ret} until t_m at low



$P_{ret} > LA'^*(n_A, t_m)$
return *null*
$P_{ret} < LA'^*(n_A, t_m)$
calculate #*k*
issue SARQ message in *br*-range
br++
collect SARP messages
$(\#k_{dis} < \#k)$ AND $(br > br_{max})$
$(\#k_{dis} < \#k)$ AND $(br <= br_{max})$
$(\#k_{dis} > \#k)$
rank by short-term rel. (e.g. hop-distance)
set $k = \{k_{dis}, \#k - \#k_{dis}\}$
return *k*
set $k = \{\text{top } \#k \text{ of } k_{dis}\}$
return *k*

Figure 5.14: *register* operation if common failure probabilities are considered.

message cost. The cost metric used here is $\#k$, which is the main factor of required messages if the plan is realized later.

To determine k, the nodes reachable by the coordinator have to be discovered first. The methods to discover assistant nodes, (i) using a SARQ message, or (ii) by routing-table lookup, have been described in Section 5.4.2. In the following, I refer to the set of discovered nodes as k_{dis}.

While node discovery using the routing table is generally preferable since no additional message costs are induced, the message-based approach is mandatory if individual probabilities for node or communication failures have to be collected from potential assistants. In the description of the implementation, I follow the approach of SARQ messages.

After k_{dis} is derived, a dissemination plan k has to be found with $LA(k, t_m) > P_{ret}$ considering short-term and long-term reachability of nodes in k.

Implementations of the *register* operation differ for the situations where common failure probabilities are assumed and where individual failure probabilities are known.

The main difference is that if individual failure probabilities are assumed, the individual nodes in k_{dis} have to be analyzed whether the required log availability is achieved, while in scenarios with common failure probabilities the required number of assistant nodes can be calculated in advance and only the number of discovered assistants is relevant.

In the following, the implementations for the two different situations are described.

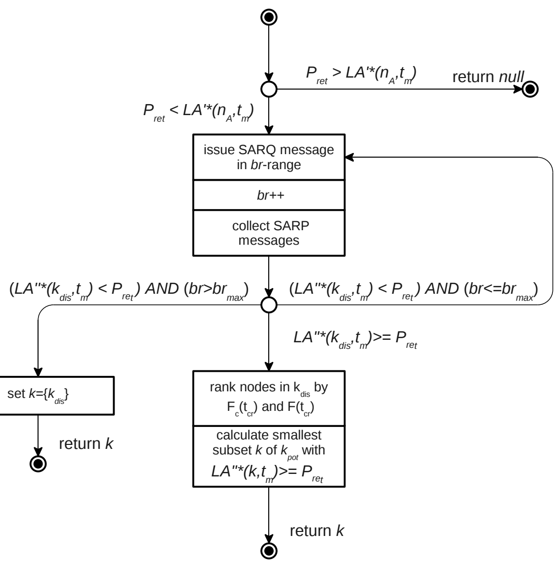

$P_{ret} > LA'^*(n_A, t_m)$
return *null*
$P_{ret} < LA'^*(n_A, t_m)$
issue SARQ message in *br*-range
br++
collect SARP messages
$(LA''^*(k_{dis}, t_m) < P_{ret})$ AND $(br > br_{max})$
$(LA''^*(k_{dis}, t_m) < P_{ret})$ AND $(br <= br_{max})$
$LA''^*(k_{dis}, t_m) >= P_{ret}$
set $k = \{k_{dis}\}$
return *k*
rank nodes in k_{dis} by $F_c(t_{cr})$ and $F(t_{cr})$
calculate smallest subset *k* of k_{pot} with $LA''^*(k, t_m) >= P_{ret}$
return *k*

Figure 5.15: *register* operation if individual failure probabilities are given.

5.6.1.1 Common Failure Probabilities

If all nodes in $\mathcal{A}$ show the same distribution for node failures and the same distribution for communication failures, the number of assistant nodes required to assure P_{ret} until t_m can be computed before the discovery process of assistants is initiated. This is feasible because every node is considered to be as reliable as any other node. Figure 5.14 depicts the implementation of the *register* method for this scenario.

First, it is checked whether the demanded log availability, given by P_{ret} and t_m, can be provided at all in the current MANET scenario. This is done by comparing the demanded log availability P_{ret} with the maximum log availability of the scenario. The maximum log availability is achieved if the log is distributed to all $n_{\mathcal{A}}$ nodes of $\mathcal{A}$. Hence, the demanded log availability can be provided if $P_{ret} < LA^*(n_{\mathcal{A}}, t_m)$. Otherwise, a null value is returned and it is left to the application to handle this situation. In contrast to the log availability calculated in Section 5.5, the log availability calculated by LA^* also considers the short-term reachability of assistants and therefore plan aging of k.

The inequation $P_{ret} \leq LA^*(\#k, t_m)$ is then used to calculate $\#k$, while $LA^*(\#k, t_m)$ is derived by the following considerations. The probability that a recovering participant can retrieve the decision log from an assistant node i depends on the following events: (i) the coordinator can place the decision log on node i at time $t_{ter,c}$; (ii) the assistant node is present in $\mathcal{A}$ at recovery time (t_{res} or $t_{ter,p}$); and (iii) a communication path between the recovering node and the assistant node is available at recovery time (t_{res} or $t_{ter,p}$).

Since the availability of the decision log is monotonically decreasing, it is sufficient to calculate the log availability at time t_m.

To derive the probability that events (i)–(iii) occur for at least one of $\#k$ assistants, all relevant event combinations are enumerated using two nested sums in Formula (5.20). The outer sum selects i assistants from k that do not receive the decision log at $t_{ter,c}$. From the $\#k-1-i$ nodes that received the decision log at time $t_{ter,c}$, j nodes are selected within the inner sum that are not present at time t_m. The $\#k-1-i-j$ nodes that are present in $\mathcal{A}$ at t_m are potential partners of the recovering participant for successful recovery. The probability of a communication path to be available with at least one of these nodes is given by $1-(1-P_{path})^{\#k-1-i-j}$. The probability of successful log retrieval for a given k and mission time t_m is then given by:

$$\begin{aligned} LA^*(\#k, t_m) &= \sum_{i=0}^{\#k-1} \binom{\#k}{i} \cdot F(t_{ter,c})^i \cdot \left[1-F(t_{ter,c})\right]^{\#k-1-i} \\ &\cdot \sum_{j=0}^{\#k-1-i} \binom{\#k-1-i}{j} \cdot F_N(t_m)^j \cdot \left[1-F_N(t_m)\right]^{\#k-1-i-j} \\ &\cdot \left[1-(1-P_{path})^{\#k-1-i-j}\right] \end{aligned} \tag{5.20}$$

If recovery from node failure is considered, the probability that $\#k-i-j$ nodes are present in $\mathcal{A}$ at recovery time (t_{res} or $t_{ter,p}$) is given by $P_j(t)$ as described in Section 5.5.1.3.

The availability of the decision log for an unrealized plan k is then simply given by replacing $\binom{\#k-1-i}{j} \cdot F_N(t_m)^j \cdot \left[1-F_N(t_m)\right]^{\#k-1-i-j}$ in Formula (5.20) with $P_j(t_m)$ resulting in $LA'^*(\#k, t_m)$:

$$\begin{aligned} LA'^*(\#k, t_m) &= \sum_{i=0}^{\#k-1} \binom{\#k-1}{i} \cdot F(t_{ter,c})^i \cdot \left[1-F(t_{ter,c})\right]^{\#k-1-i} \\ &\cdot \sum_{j=0}^{\#k-1-i} P_j(t_m) \cdot \left[1-(1-P_{path})^{\#k-1-i-j}\right] \end{aligned} \tag{5.21}$$

After $\#k$ is calculated using Formulae (5.20) or (5.21), the discovery process using SARQ messages is started in order to discover at least $\#k$ assistants in br_{max} hop distance, as shown in Figure 5.14, where $\#k_{dis}$ describes the number of discovered assistants within a certain broadcast radius.

If $\#k_{dis} < \#k$, k is given by k_{dis} and the number of missing assistants is given by $n_{mis} = \#k - \#k_{dis}$. Note that an undefined set of k such as n_{mis} can only be added if $n_{\mathcal{A}}$ is approximately known. If $n_{\mathcal{A}}$ is unknown and $\#k_{dis} < \#k$, the *register* method cannot safely provide a dissemination plan and returns null, as it is not assured that n_{mis} nodes can be found even if the complete network is flooded to realize k.

If the discovery process returns k_{dis} with $\#k_{dis} \geq \#k$, a subset of k_{dis} can be chosen for k considering the individual short-term reachability of nodes in k_{dis}. While individual short-term reachability of assistant nodes is not considered in (5.20) and (5.21), it can be regarded by ranking the nodes of k_{dis} according to their short-term reachability and choosing the top $\#k$. Hence, assistant nodes are chosen that are reachable for the coordinator at $t_{ter,c}$ with higher probability.

5.6.1.2 Individual Failure Probabilities

If individual node or communication failure probabilities are given, the number of required assistant nodes cannot be derived before the discovery process has been started, but a given k_{dis} has to be analyzed if the containing nodes provide the desired log availability. Hence, besides calculations used to calculate the log availability, the process of the *register* method also changes as shown in Figure 5.15.

To assert that a log availability larger than P_{ret} can generally be provided until t_m, the same calculation as in the previous section is used, while here an average failure probability must be assumed.

If P_{ret} can be provided until t_m, assistant discovery is started immediately using an iterative ring search similar to the situation with common failure probabilities. In every iteration of the ring search, the log availability achieved by the assistant nodes k_{dis} discovered so far is analyzed and compared to the required log availability. If the log availability guaranteed by k_{dis} does not meet P_{ret}, the search radius is increased successively. If the required availability has been achieved, the smallest subset of k_{dis} providing the desired log availability is used as k. If the maximum broadcast range of the ring search has been reached, k_{dis} is used for k and it is left to the *writeSLS* operation to discover additional assistants or fall back to flooding.

In the following, I will present calculations to derive the log availability provided by a discovered assistant set k_{dis} for the situations where (i) individual communication failure distributions $F_{i,c}(t)$ and common node failure distributions $F_N(t)$ are considered, and (ii) individual node failures $F_{i,N}(t)$ as well as individual communication failures $F_{i,c}(t)$ are assumed.

Situation (i) uses schemes to approximate the short-term reachability as presented in Section 5.4.3. These are either based on knowledge about position and movement direction of individual nodes (see Section 5.4.3) or based on information obtained from routing tables and in future, possibly through cross-layer approaches that use link layer information etc. to validate the reliability of paths. Note that the probability for a direct link break as derived in 5.4.3 is a pessimistic approximation for the short-term reachability if multi-hop routing is used.

If individual probabilities of communication failures are available, the individual short-term reachability of assistant nodes is used to consider plan aging, i.e. the probability that a discovered assistant is not reachable for the coordinator when the *writeSLS* operation is executed. Recall that in the previous calculations, $F_c(t)$ was assumed to be equal for all assistants, while I showed in Chapter 4 that this assumption does not hold for different hop distances.

Assuming individual probabilities for communication failures $F_{i,c}(t)$, leads to an individual general failure $F_i(t)$ of node i, calculated by

$$F_i(t) = 1 - \Big[\big[1 - F_N(t)\big] \cdot \big[1 - F_{i,c}(t)\big]\Big] \tag{5.22}$$

To calculate the log availability of k, the probabilities of different subsets of k to receive the decision log have to be considered. I use the set notation introduced in Subsection 5.5.1.2 to identify subsets of k in the following.

The availability of the decision log achieved by k is now calculated by $\widetilde{LA}(k, t_m)$ given by Formula (5.23). The only difference to $LA^*(k, t_m)$ (given by Formula (5.20)) is that all possible subsets of k have to be considered in $\widetilde{LA}(k, t_m)$, as now the log availability depends on which nodes are in k_{dis}. I use nested sums in Formula (5.23) to enumerate the relevant events and subsets of

k. The outer sum iterates the number i of failed assistants (node or communication failures), while the inner sum enumerates all $\binom{\#k-1}{i}$ subsets of k with i nodes. The probability that at least one node of the $\#k-1-i$ nodes that received the decision log is present in $\mathcal{A}$ is calculated as in Formula (5.20). The log availability $\widetilde{LA}(k, t_m)$ is now given by

$$\begin{aligned}\widetilde{LA}(k, t_m) &= \sum_{i=0}^{\#k-1} \sum_{j=1}^{\binom{\#k-1}{i}} F_{k_{i,j}}(t_{ter,c}) \cdot \left[1 - F_{k\setminus k_{i,j}}(t_{ter,c})\right] \\ &\cdot \sum_{m=0}^{\#k-1-i} \binom{\#k-1-i}{m} \cdot F_N(t_m)^m \cdot \left[1 - F_N(t_m)\right]^{\#k-1-i-m} \\ &\cdot \left[1 - (1 - P_{path})^{\#k-1-i-m}\right] \qquad (5.23)\end{aligned}$$

If recovery of nodes failures is assumed, the probability that m nodes are not present in $\mathcal{A}$ at time t is given by $P_m(t)$, as calculated in Section 5.5. The log availability is then derived by $\widehat{LA}(k, t_m)$.

$$\begin{aligned}\widehat{LA}(k, t_m) &= \sum_{i=0}^{\#k-1} \sum_{j=0}^{\binom{\#k-1}{i}} F_{k_{i,j}}(t_{ter,c}) \cdot \left[1 - F_{k\setminus k_{i,j}}(t_{ter,c})\right] \\ &\cdot \sum_{m=0}^{\#k-1-i} P_m(t_m) \cdot \left[1 - (1 - P_{path})^{\#k-1-i-m}\right] \qquad (5.24)\end{aligned}$$

The discovery radius is incremented until br_{max} is reached, as shown in Figure 5.15. If not enough assistant nodes are discovered within br_{max} distance, either k_{dis} is used as k or an undefined set of assistant nodes is added to k_{dis}. For an undefined assistant, the common failure distribution $F_c(t)$ is assumed, as in the previous scenario.

If k_{dis} provides sufficient log availability, the smallest subset of assistant nodes is derived by ranking nodes in k_{dis} by $F_{i,c}(t_{ter,c})$ and successively computing the log availability of top-k subsets until the subset is found that guarantees the demanded log availability.

In situation (ii), individual probability distributions for communication failures $F_{i,c}(t)$ and also individual probability distributions for node failures $F_{i,N}(t)$ are assumed. The probability of assistant presence has now to consider all relevant combinations of assistant nodes in k to derive the availability of the decision log, while the general failure probability of an assistant node i is given by

$$F_i(t) = 1 - \Big[\left[1 - F_{i,c}(t)\right] \cdot \left[1 - F_{i,N}(t)\right]\Big] \qquad (5.25)$$

The log availability at time t is given by $\ddot{LA}(t)$. I used the following notation to identify the different subsets of k: I denote $k_l = k \setminus k_{i,j}$ and the h-th subset with m elements of k_l by $k_{l,m,h}$. The probability that all nodes of $k_{l,m,h}$ suffer from a node failure until t is denoted by $F_{k_{l,m,h},N}(t)$ and given by $F_{k_{l,m,h},N}(t) = \prod_{g=1}^{m} F_{k_{l,m,h,g},N}(t)$, where $k_{l,m,h,g}$ is the g-th element of set $k_{l,m,h}$. $\ddot{LA}(t)$ is then given by

$$\begin{aligned} \ddot{LA}(k, t_m) &= \sum_{i=0}^{(\#k-1)} \sum_{j=1}^{\binom{\#k-1}{i}} F_{k_{i,j}}(t_{ter,c}) \cdot \left[1 - F_{k \setminus k_{i,j}}(t_{ter,c})\right] \\ &\cdot \sum_{m=0}^{(\#k-1-i)} \sum_{h=1}^{\binom{\#k-1-i}{m}} F_{k_{l,m,h},N}(t_m) \cdot \left[1 - F_{k_l \setminus k_{l,m,h},N}(t_m)\right] \\ &\cdot \left[1 - (1 - P_{path})^{(\#k-1-i-m)}\right] \end{aligned} \tag{5.26}$$

The process of the *register* operation for situation (ii) is equal to situation (i) where only individual communication failures are assumed (see Figure 5.15). The only differences are that different calculations are used, i.e. Formula (5.26) instead of (5.23) or (5.24).

However, to derive the optimal set k, a simple ranking as in situation (i) according to $F_{i,c}(t)$ is not enough, as the assumption of individual node and communication failures now leads to an optimization problem. This problem is not considered any further here.

5.6.2 Implementation of *writeSLS*

The *writeSLS* operation is responsible for implementing the dissemination of the decision log and to assure that its availability is larger than P_{ret} until t_m. The dissemination process of the decision log is time-critical, as a node failure during execution of the *writeSLS* operation can result in a log availability smaller than P_{ret}.

Although the probability of a node failure is negligible for short time periods, as shown in Chapter 4, it can take some some time until enough assistant nodes are reached especially in sparse MANET scenarios. To control the probability of a node failure to impede provisioning of the required availability of the decision log, I defined an upper bound Δ_w for the execution time of the *writeSLS* operation.

An implementation of the *writeSLS* operation has to guarantee termination after Δ_w at the latest. Such a bound of execution time can generally be implemented by using a fall-back mechanism from directed or undirected dissemination to a flooding strategy.

The basic implementation of the *writeSLS* operation is simple: SLS log storage (SLR) messages containing the decision log are distributed in a directed or undirected manner. Assistant nodes receiving such a message store the decision log locally and answer with a so-called SLS log saved message (SLSA). Depending on the scenario an SLSA message can contain the individual node failure probability of the node that stores the log. By collecting and analyzing the SLSA messages, it can be decided, whether a sufficient log availability is achieved or not. If a sufficient log availability is not reached within Δ_w, a flooding-based dissemination is initiated.

This general behavior has to be implemented differently for scenarios with and without multi-hop routing. When multi-hop routing is available, directed dissemination in multiple hop distances is possible using a plan k, while without multi-hop routing, directed dissemination is only feasible in a small hop range and no precalculated plan is given. In the following, I describe the implementation for both situations in detail.

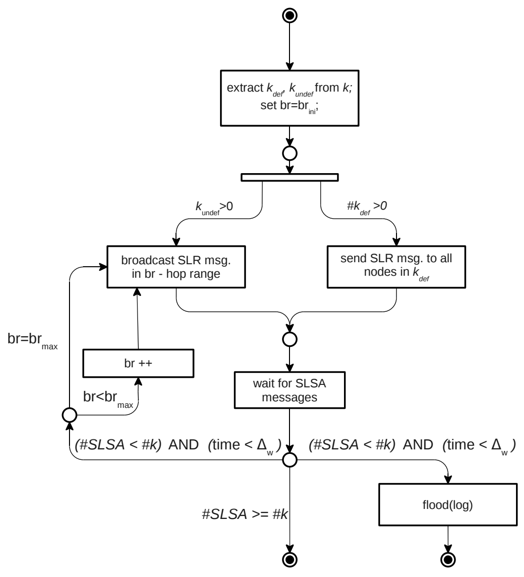

extract k_def, k_undef from k; set br=br_ini;
k_undef>0
#k_def >0
broadcast SLR msg. in br - hop range
send SLR msg. to all nodes in k_def
br=br_max
br ++
br<br_max
wait for SLSA messages
(#SLSA < #k) AND (time < Δ_w)
(#SLSA < #k) AND (time < Δ_w)
#SLSA >= #k
flood(log)

Figure 5.16: *writeSLS* operation if common node failure probabilities and multi-hop routing are assumed.

Multi-hop Routing

Figures 5.16 and 5.17 depicts the *writeSLS* operation if multi-hop routing is used. First, the set of assistant nodes defined by their nodeId (in the following, I call this set k_{def}) is extracted from k and an SLR message is sent directly to all nodes in k_{def}. If k contains an undefined component ($k_{undef} > 0$), an SLR message is broadcasted additionally in br_{ini}-range.

The initial broadcast range br_{ini} is set to one or two, depending on the expected number of direct neighbors in $\mathcal{A}$.

Nodes that receive an SLR message answer with an SLSA message that contains individual information about their probability of presence in $\mathcal{A}$. Note that such information is only included in scenarios where individual failure probabilities are known. After the first round of SLR messages is issued, SLSA messages are collected and analyzed.

If common failure probabilities are assumed, the number of received SLSA messages is compared to the number of assistants required to achieve P_{ret}. If the required number $\#k = \#k_{def} + k_{undef}$ is not reached ($\#SLSA < \#k$), additional SLR messages are broadcasted with increased hop range to discover more assistant nodes. Note that the required number of assistant nodes given by $\#k$ was calculated by the register operation.

Figure 5.17 shows the *writeSLS* operation for the case where individual node failure probabilities and multi-hop routing is assumed. In this scenario, SLSA messages contain individual failure probabilities of assistant nodes and the currently achieved availability of the decision log is calculated using Formula (5.18).

If the achieved log availability does not meet P_{ret}, the broadcast range is suc-

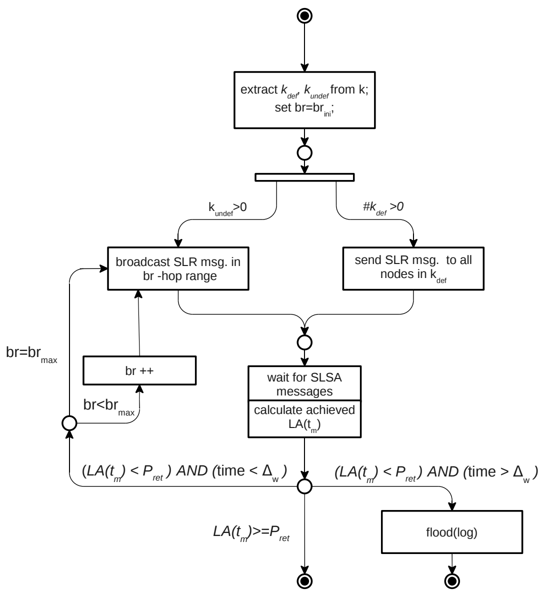

extract k_{def}, k_{undef} from k; set br=br_{ini};
k_{undef}>0
$\#k_{def}$ >0
broadcast SLR msg. in br -hop range
send SLR msg. to all nodes in k_{def}
br=br_{max}
br ++
br<br_{max}
wait for SLSA messages
calculate achieved LA(t_m)
($LA(t_m) < P_{ret}$) AND (time < Δ_w)
($LA(t_m) < P_{ret}$) AND (time > Δ_w)
$LA(t_m)>=P_{ret}$
flood(log)

Figure 5.17: *writeSLS* operation if individual node failure probabilities and multi-hop routing are assumed.

cessively increased by one, until the maximum broadcast range br_{max} is reached. Broadcast of SLR messages is repeated until either (i) the required availability of the decision log is reached or (ii) Δ_w is exceeded. In case (i), the operation terminates, while in case (ii), a flooding protocol is initiated to flood $\mathcal{A}$ with the decision log using a so called SLS flood log (SFL) message. Nodes receiving an SFL message simply store the decision log, while no SLSA messages are issued. The *writeSLS* operation terminates after initiating the flooding protocol.

No Multi-hop Routing

In a scenario where no multi-hop protocol is used, no dissemination plan k is provided and hence only undirected dissemination can be used. The implementation of the *writeSLS* operation in this situation is shown in Figure 5.18 for individual node failure probabilities and in Figure 5.19 for common node failure probabilities.

To disseminate the decision log broadcasting of the SLR message is repeated until either the desired availability of the decision log is reached or Δ_w is exceeded. The broadcast range can be increased until a certain br_{max} is reached, while br_{max} should be set to smaller values than with multi-hop routing. However nodes receiving SLR message cannot acknowledge this message directly but have to broadcast the SLSA message in the same hop distance to deliver their SLSA message to the node executing the *writeSLS* operation. This results in high message overhead.

In case individual node failure probabilities are assumed, SLSA messages contain the individual node failure probability of the sender. Individual failure

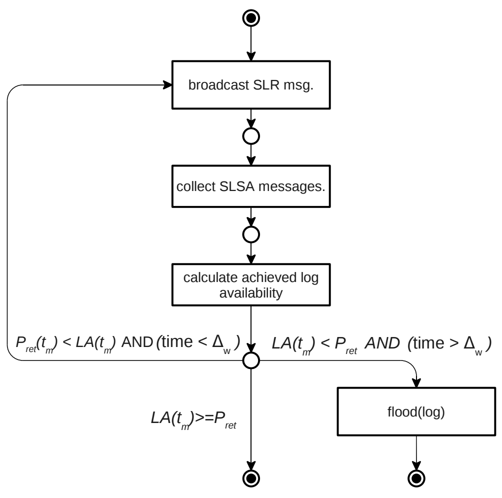

broadcast SLR msg.
collect SLSA messages.
calculate achieved log availability
$P_{ret}(t_m) < LA(t_m)$ AND (time < Δ_w)
$LA(t_m) < P_{ret}$ AND (time > Δ_w)
$LA(t_m)>=P_{ret}$
flood(log)

Figure 5.18: *writeSLS* operation if individual failure probabilities and no multi-hop routing are assumed.

probabilities are analyzed to compute the log availability achieved so far by Formula 5.18, as shown in Figure 5.18. If common node failure probabilities are assumed the number of received SLSA messages is compared to the required assistants $\#k$ calculated by Formulae 5.17 or 5.19. When Δ_W is exceeded, a flooding scheme similar to the previous situation with multi-hop routing is executed.

The main difference in calculations of the log availability compared to the situation with multi-hop routing is that P_{path} for 1–2 hop paths is used and not for n-hop paths. Hence, the number of assistant nodes required is much larger compared to the multi-hop situation, and a fallback to flooding is more likely.

5.6.3 Implementation of *readSLS*

The responsibility of the *readSLS* operation is to discover a node holding the decision log and to retrieve the log item at low message expense. Generally, the implementation of a *readSLS* operation has to be coordinated with the *writeSLS* operation for high message efficiency. In multi-hop scenarios this means that k is considered by the *readSLS* operation, and if no multi-hop routing is used, the broadcast horizon of the *readSLS* operation has to match the path probability used in computations of the according *writeSLS* operation.

Successful retrieval of the decision log is a certain event if the *readSLS* operation is executed often enough. However, the retrieval probability P_{ret} is the probability that the decision log is retrieved within the first recovery attempt. The time to execute this first recovery process is denoted by Δ_r.

In the following, the implementation of the *readSLS* operation for scenarios with and without multi-hop routing is described. A distinction between scenarios with common failure probabilities and scenarios with individual failure probabilities is not necessary here.

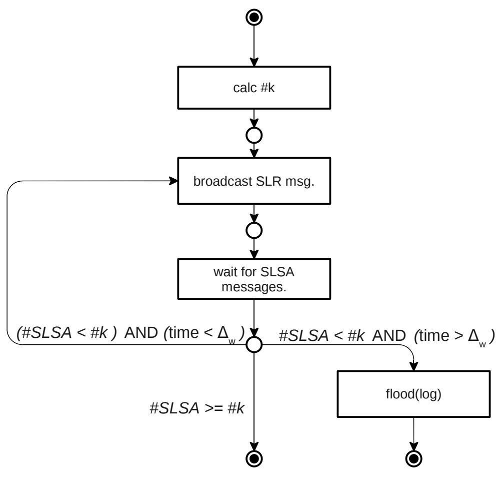


Figure 5.19: *writeSLS* operation if common node failure probabilities and no multi-hop routing are assumed.

Multi-hop Routing

If multi-hop routing is used, a dissemination plan k with a defined set of assistant nodes is given and can be used for retrieving the decision log. Figure 5.20 depicts this process. If k contains a defined set of assistant nodes (denoted by k_{def}), an SLS search request (SSR) message is sent to every assistant in k_{def}.

An assistant node that receives an SSR message and holds the desired decision log answers with an SLS log answer (SLA) message containing the decision log. If an SLA message is received, the *readSLS* operation returns the decision log and terminates as shown in Figure 5.20.

If no SLA message is retrieved by contacting the assistants defined in k (k_{def}), assistants holding the log have to be discovered. Whether a ring search is used or the network is immediately flooded with an SSR message using a broadcast-in-space scheme depends on what path probability P_{path} was used in computations of the *writeSLS* operation. I propose to use a value for P_{path} describing the path probability of paths with arbitrary lengths if multi-hop routing is used. In this case, immediate flooding of the SSR message in $\mathcal{A}$ is feasible. However, if a P_{path} value for 1–2 hop paths is used in calculations of the *writeSLS* operation, a broadcast of the SSR message in 1–2 hop distances should return the decision log at probability P_{ret} within Δ_r. Δ_r is given by the time required to query all nodes in k_{def}, flood the network, and to wait for SLA messages. Note that the *readSLS* operation as shown in Figure 5.20 terminates not before an SLA message is received, while such a message is received at probability P_{ret} within Δ_r.

No Multi-hop Routing

If no multi-hop routing is used, the *readSLS* operation cannot use a plan k and the only option to discover assistant nodes is to use a discovery message.

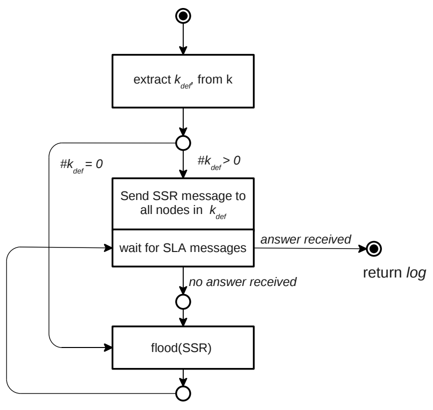


Figure 5.20: *readSLS* operation if multi-hop routing is used.

The maximal range in which the SSR message has to be distributed to discover an assistant node at probability P_{ret} depends on the path probability P_{path} used in computations of the *writeSLS* operation. For example, if the *writeSLS* operation uses the path probability for 1–2 hop paths to derive the availability of the decision log, then the SSR message has to be broadcasted in a 1–2 hop range to locate the decision log at P_{ret}. The initial broadcast range for the SSR message is called br_{ini}. If the decision log cannot be located in distance br_{ini}, the straightforward strategy is to successively increase the broadcast range to discover remote assistants. However, the probabilistic guarantee provided by the SLS is that the decision log is retrieved in br_{ini} range at probability P_{ret}. Hence, Δ_r is given by the time required to broadcast the SSR message in br_{ini} range and to distribute an SLA message in br_{ini} distance.

This process is depicted in Figure 5.21. In the implementation of this approach for evaluation purposes, I use an initial search range of 2 hops. The main reason to use this value is that there is a simple analytical way to calculate the probability for a 1–2 hop path, as presented in Section 5.5.3. In the implementation an upper bound for the distribution horizon of the SSR message is used for preventing a complete flooding of the network, which is not feasible if a 1–2 hop path probability is used within the *writeSLS* operation.

5.6.4 Summary - Lightweight Implementation

In this section, I have presented an implementation of the SLS operations based on the calculation models developed in the previous sections. The general approach followed by the implementation was a lightweight “fire-and-forget” scheme, based on the idea of identifying a small set of nodes providing sufficient log availability without further maintaining the availability of the decision log once disseminated.

The approach is lightweight in the sense that no additional overlay structure is used to discover and choose assistant nodes and the log availability is not maintained after the *writeSLS* method terminates. I will present a contrary implementation approach of the SLS based on a logical overlay structure in

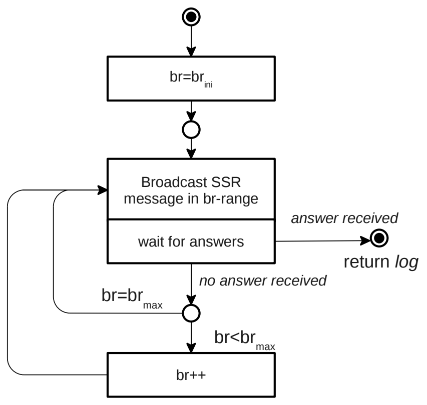


Figure 5.21: *readSLS* operation if no multi-hop routing is used.

Section 5.8, where the log availability is constantly maintained and assistants are chosen based on the overlay structure.

The internal workflows of the *register*, *readSLS,* and *writeSLS* operations presented above are tightly bound to the calculation model and whether multi-hop routing is assumed or not. The basic difference between implementations for scenarios with and without multi-hop routing is that no plan is prepared if no routing algorithm is used, while the underlying calculations are similar in both scenarios.

To show that the lightweight implementation approach is feasible in a real MANET, it has to be shown that the underlying log availability model is a feasible abstraction of the real world. In the following section, I will present simulations indicating correctness of the models presented here.

5.7 Simulative Evaluation

For a simulative evaluation of the lightweight implementation approach and its underlying calculation model I use the *ns2* network simulator [2]. Simulation results presented in the following show, that the computation model developed to calculate availability of decision logs provides a precise approximation of the real availability, and that blocking risks can be compensated for by the SLS as predicted.

First, I will evaluate the calculation model of log availability by simulation as this model is fundamental to the lightweight approach. Afterwards, I will present simulation results showing that the predicted reduction in blocking risks with the SLS can be actually observed in experiments for the example MANET scenario of this work. To conclude the simulation-based evaluation of the lightweight approach, I will briefly present some simulation results taken from [118]. These results demonstrate that P_{ret} can be provided in a wide range of MANET scenarios. More precisely, the density of the considered MANET scenarios is varied to observe the behavior of the SLS at different network characteristics.

While other MANET research is often primarily concerned with reduction of required messages (e.g. in routing, data dissemination, or service discovery) and therefore has to evaluate whether this objective is met or not, evaluation of the message load caused by the lightweight SLS implementation is problematic for different reasons:

(i) To derive the message load induced by the SLS correctly, messages issued on lower layers, e.g. on the routing layer, also have to be considered. It is erroneous to count an SSR message sent directly to an assistant node as one message, because on the routing layer a route to the receiving node possibly has to be discovered by flooding the complete network (see Section 2.1.3). The message load induced by a reactive routing scheme (e.g. AODV) depends on the general message load in $\mathcal{A}$, i.e. if a lot of messages are transferred, the probability of a route to be known already is higher compared to a situation where generally few messages are transferred in $\mathcal{A}$. Hence, to count messages by the allocation-by-cause principle, an additional traffic model is required describing the traffic caused by other applications and the transaction load processed in $\mathcal{A}$. Given such a model, the messages required to deliver a direct message over a known route depends on the hop count of the communication path, which again depends on node speeds, initiation distance in transaction processing, and network size. The message load obtained by simply counting all messages transferred in $\mathcal{A}$ with and without the SLS is therefore biased by the traffic model of other applications, by the transaction load, by node speeds, and also by the implementation and configuration of multi-hop routing and hence is hard to interpret.

(ii) Additionally, the message load induced by the SLS depends on the flooding scheme used. While in the implementation evaluated here a simple probabilistic flooding approach is used (see Subsection 5.4.1.3), more message-efficient schemes such as hyper-gossiping (see Section 2.1.4) have been proposed. More efficient flooding approaches have not been used in simulations as their implementation is complex and out of the scope here. Hence, the number of messages transferred is also implementation-dependent and therefore shows little significance for an evaluation of the general SLS scheme. I will therefore focus on verifying the abstractions and applicability of the proposed schemes by simulation.

In the following, I verify the log availability model and the predicted compensation for blocking (i) and extended uncertainty (i) situations by simulation for the example MANET scenario.

5.7.1 Evaluation of Log Availability

The implementation of the SLS as proposed in Section 5.6 assures a desired log availability by determining the required assistant nodes based on the calculation model proposed in Section 5.5 by Formulae (5.17)–(5.19). In the following, this model is analyzed by simulation. As the lightweight approach is mainly based on this model, correctness of this model is a strong indicator for the applicability of the lightweight SLS implementation.

To derive a significant number of tests within a feasible simulation time, I created relevant blocking situations artificially by letting one participant ignore the *commit* message. The uncertain participant then repeatedly initiates recovery using the SLS and records time and success of each recovery attempt. Note that cooperative recovery is not initiated by the recovering participant here.

While the implementation proposed in Section 5.6 differs depending on whether multi-hop routing is available or not, the same calculation model (Formulae (5.17)–(5.19)) is applied in both situations. Simulations of scenarios with AODV and without multi-hop routing schemes are used to prove that the calculation model actually provides accurate results in both situations.

In a first step, I use the example MANET scenario of this work to simulate scenarios where nodes are assumed to reconnect to $\mathcal{A}$, i.e. recovery from node failures is considered, and afterwards I simulate scenarios where extra stable nodes are chosen as assistants.

5.7.1.1 Considering Recovery of Assistants

Figure 5.22 depicts the measured success rates of the *readSLS* operation within its first internal message round.

For the situation where AODV is used at the routing layer (resulting in P_{path}= 0.7), Figure 5.22(a) compares the predicted log availability computed by $LA'(k,t)$ with $\#k = 4$ for two different combinations of failure and recovery probabilities. In both scenarios, a common node failure and recovery rate is assumed. Figure 5.22(b) depicts results for scenarios where no multi-hop routing is used.

Measurements depicted in Figure 5.22 are the proportion of successful recovery attempts (where at least one SLA message was received in Δ_r). The rates are derived from 1000 tests initiated within a simulation period of 500000 s, while for each test, 600 recovery attempts are initiated in intervals of 100 s. Hence, each measurement in the diagram presents the proportion of successful recoveries at recovery time t. The exact simulation parameters of the MANET are given in Appendix B.1.1.1.

The prediction made by $LA'(k,t)$ can be interpreted as the expected proportion of successful recovery attempts, while measured values show a dispersion around this value. The standard deviation of success rates presented in the diagram is calculated by the following considerations: The measured proportion of successful tests is derived from n_t=1000 tests, while the hypothesis is that the rate of successful tests is given by $LA'(k,t)$. The rate of successful tests can then be described by a binomial distribution:

$$P(l) = \binom{n_t}{l} \cdot p^l \cdot (1-p)^{n_t - l} \tag{5.27}$$

with n_t=1000 (total number of tests at time t) and $p = LA'(k,t)$. If $LA'(k,t)$ predicts the success rate, then measurements have to disperse around $LA'(k,t)$ with standard deviation σ given by

$$\sigma = \sqrt{n_t \cdot p \cdot (1-p)} \tag{5.28}$$

Hence, if $LA'(k,t)$ is correct, the majority of measurements is expected to occur in the interval $\big[(LA'(k,t){-}\sigma/n_t)\,,\,(LA'(k,t){+}\sigma/n_t)\big]$. In Diagrams 5.22(a) and 5.22(b), the bounds of this interval are depicted by the green curves.

Results presented in Figure 5.22(a) show that measured success rates disperse as expected within predicted intervals, while Curve 1 presents a situation where nodes remain connected to $\mathcal{A}$ longer than disconnected (expectations of exponential distributions are 1.5 h connected and 15 min disconnected). Curve 2 assumes nodes to experience longer disconnection periods (expectations: 30 min in $\mathcal{A}$ and 1.5 h disconnected). The other parameters of the simulation scenario are described in Appendix B.1.1.1. It therefore follows that $LA'(k,t)$ is a meaningful prediction for the log availability in the two considered scenarios.

If no multi-hop routing is used, the recovering node cannot directly address assistant nodes. Hence, repeated broadcast of an SSR message in 1–2 hop distance is used to discover an assistant that holds the decision log. Discovery of assistants in 1–2 hop distances results in a path probability of 0.2 used in calculations for the example MANET scenario.

While the implementation of the *readSLS* operation is different for non multi-hop scenarios, i.e. when no plan k is used, the log availability is predicted similarly as in the multi-hop case by Formula (5.19). Simulation results depicted in Figure 5.22(b) confirm the applicability of the proposed calculation model also for single-hop scenarios. Curve 2 of Figure 5.22(b) presents the resulting log availability for $\#k$=4 at the same failure rates as of Curve 2 in the multi-hop case (given in Figure 5.22(a)). It can be observed that the predicted decreased log availability is reflected by measured success rates in both scenarios.

Simulation results and predictions of Curve 1 in Figure 5.22(b) present the log availability of a scenario where flooding of the decision log to all nodes in $\mathcal{A}$ results in a low log availability of only 54 %. Hence, this scenario is an example with a low $P_{ret,max}$, requiring assertion whether the demanded P_{ret} can be provided at all (see Section 5.6.1).

Given the comparison of predictions and simulation results presented in Figure 5.22(a) and 5.22(b), I conclude that the proposed calculation model underlying the SLS accurately abstracts MANETs with and without multi-hop routing and common node failures.

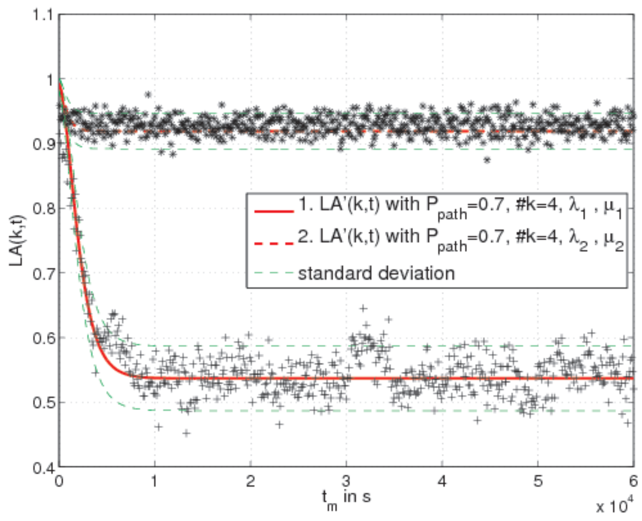
5.7.1.2 Considering Individual Failure Probabilities

If individual node failure probabilities of assistant nodes are considered, and recovery of assistants from node failures is not regarded, Formula (5.18) is used to derive the log availability. In the following, predictions of this formula are compared to simulation results. This evaluation is of special interest as it allows scenarios to be analyzed where more reliable nodes, such as stable command vehicles in the disaster setting, can be used to preserve a decision log. I consider such situations to be especially relevant in the real world.

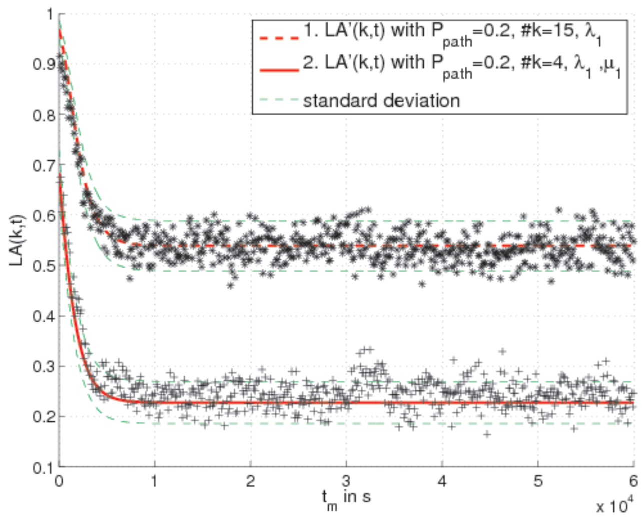
Figure 5.23 compares predictions by Formula (5.18) and simulation results for the example MANET scenario. Simulation results are presented for two different settings: (i) two assistants are used, while one of the two assistant nodes is quasi stable (e.g. a command truck) with an expected sojourn time in $\mathcal{A}$ of 138 h; and (ii) four assistant nodes are used, two with an expected sojourn time of 30 min and the other with 3.5 h.

Measurements of the proportion of successful *readSLS* operations are again obtained from 1000 artificially created blocking situations, while for each blocking situation 600 recovery attempts are initiated in intervals of 100 s. The exact parameters of the *ns2* simulation are given in Appendix B.1.1.1.

Curve 1 of Figure 5.23(a) shows the predicted success rate and measurements for setting (i) (plan k_1) if multi-hop routing is used. Here, analytical as well as simulative results show a log availability higher than 60 % over the complete simulation time, while the measured success rates remain within the bounds of the standard deviation.



(a) Simulation results of log availability for recovering nodes in the example scenario with multi-hop routing.



(b) Simulation results of log availability for recovering nodes without multi-hop routing.

Figure 5.22: Simulation results of log availability if assistants are assumed to recover from node failures. Node failures and recovery is exponentially distributed with parameters $\lambda_1 = 1800^{-1}$, $\lambda_2 = 5400^{-1}$, $\mu_1 = 5400^{-1}$ and $\mu_2 = 1800^{-1}$.

Curve 2 of Figure 5.23(a) depicts experimental results and predictions for setting (ii) with four assistants, while two assistants are slightly more stable with $\lambda_3 = 12600^{-1}$ than the other two with $\lambda_1 = 1800^{-1}$ (plan k_2). Considering measurements and deviations, I conclude that Formula (5.18) accurately predicts the real world.

If no multi-hop routing is used, the log availability is also predicted by Formula (5.18) and the expected decrease in log availability for plan k_1 and plan k_2 without multi–hop routing is again confirmed by simulation results, as presented in Figure 5.23(b) by Curve 1 and Curve 2 respectively.

I omit an experimental verification of Formula (5.17) here, as scenarios with common node failures and no recovery of assistant nodes are a special case of the situation evaluated here.

5.7.1.3 Summary - Evaluation Log Availability

The simulation results presented above indicate that the proposed log availability model accurately abstracts the real log availability in a MANET. Hence, I showed that the proposed SLS implementation can increase the probability of leaving uncertainty in a controlled way using the calculation model and the SLS operations proposed.

To give an idea of the possible benefit of the SLS in the example scenario, Figure 5.24 compares the probability for successful recovery with and without the SLS. If successful recovery depends only on the availability of the coordinator, the probability of successful recovery is 65 % two minutes after the coordinator decided on the global decision and decreases to 17.5 % after 2.5 h. In contrast, the probability for successful recovery remains at 58 % after 2.5 h if the SLS is used by the coordinator to preserve the decision log and six assistant nodes are used.

5.7.2 Reduction of Blocking Risk

While in the previous subsection the calculation model of log availability was evaluated, this subsection analyzes for an example scenario whether the predicted reduction in blocking risk is actually met when the SLS is used.

Hence, the situation I am interested in here is that: (i) a transaction participant PA suffers blocking situations addressed here, and (ii) the decision log is not available to the recovering participant at recovery time.

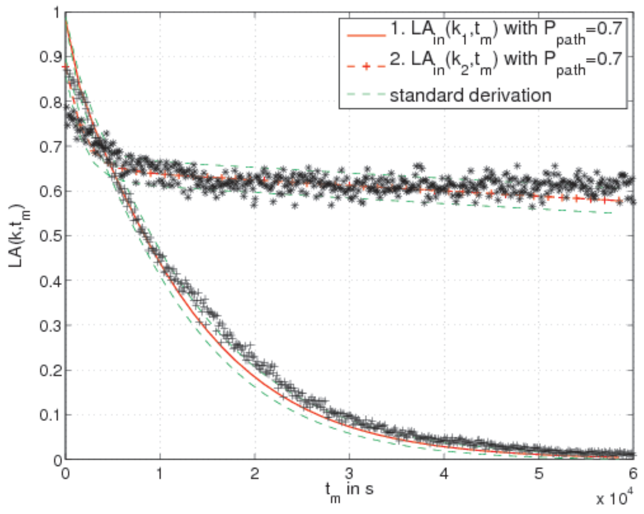
The probability for (i) was calculated in Chapter 4 by Formulae (4.29) and (4.28) for the strict transaction model, and by Formulae (4.45) and (4.44) for the semantic transaction model.

The risk of PA to suffer from blocking and unsuccessful recovery at time $t_{ter,p}$ or t_{res} if the SLS is used is now given by $P_{u,SLS}(t_p)$ for the strict model (recovery of communication failures is not assumed):

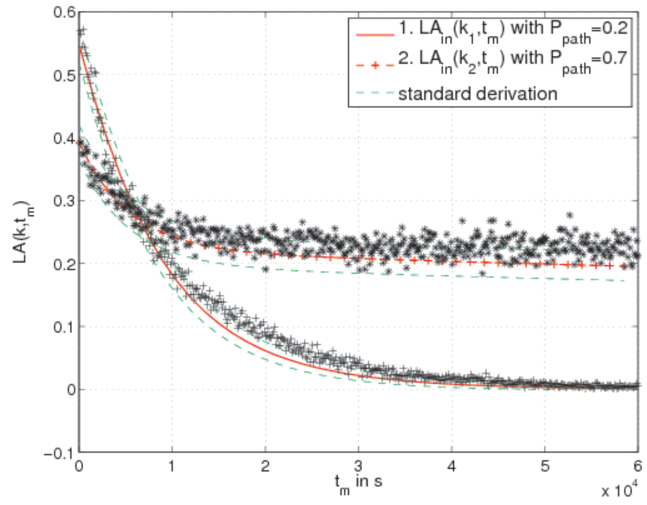
$$P_{u,SLS}(t_p) = P_u(t_p) \cdot [1 - LA(k, t_m)] \tag{5.29}$$

The log availability $LA(k, t_m)$ is calculated as described in Section 5.5.4.

If recovery of communication failures is considered, the risk of PA to suffer blocking is calculated by Formula (4.28) for the strict transaction model. The risk of blocking if the SLS is used is then derived by $P'_{u,SLS}(t_p)$.



(a) Simulation results of log availability if multi-hop routing is used.



(b) Simulation results of log availability if no multi-hop routing is used.

Figure 5.23: Simulation results of log availability if individual failure probabilities of assistants are considered but no recovery of assistant nodes is assumed. k is given either by k_2=$\{\lambda_1,\lambda_1,\lambda_3,\lambda_3\}$ and k_1=$\{\lambda_2,\lambda_4\}$, with $\lambda_1 = 1800^{-1}$, $\lambda_2 = 5400^{-1}$,$\lambda_3 = 12600^{-1}$, and $\lambda_4 = 500000^{-1}$.

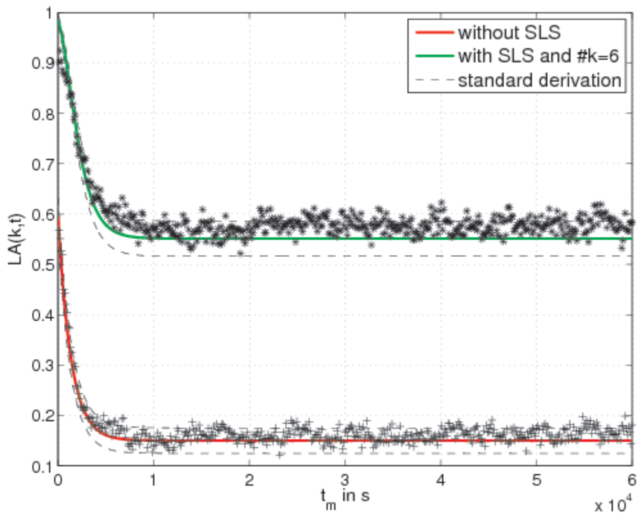


Figure 5.24: Comparison of log availability with and without SLS.

$$P'_{u,SLS}(t_p) = P'_u(t_p) \cdot [1 - LA(k, t_m)] \tag{5.30}$$

To evaluate whether the predicted reduction in blocking risk caused by the SLS can be observed in reality, i.e. in *ns2* simulations, I simulated transaction processing using the SLS for the example MANET scenario of this work.

For simulations, I assume exponentially distributed node failures and recoveries in $\mathcal{A}$ with parameters $\lambda = 5400^{-1}$ and $\mu = 1800^{-1}$ for all nodes. The measured rates of occurred blocking (i) situations is derived from 8000 transactions processed for different t_p.

Figure 5.25 depicts measured blocking rates as well as predictions made by $P_{u,SLS}(t_p)$ and by $P'_{u,SLS}(t_p)$ if cooperative recovery is not used but only the SLS. Curve 1 of Figure 5.25 shows the probability for blocking without any additional recovery scheme (i.e. only the coordinator is used), while Curves 2 and 3 depict blocking risks predicted by $P_{u,SLS}(t_p)$ and $P'_{u,SLS}(t_p)$.

It can be observed that predictions slightly overestimate the measured blocking risks, as shown by Curve 4. The main reason for this discrepancy is that calculations of Formulae (5.29) and (5.30) assume the log availability given at t_m, which is a pessimistic assumption and hence also an upper bound for the blocking risk. Nodes recovering early (at $t_{ter,p}$) benefit from a higher log availability, because the log availability is monotonically decreasing over time as also shown in the previous subsection and in Section 5.5.4. Since a majority of participants recover after suffering a communication failure at $t_{ter,p}$, the measured blocking rate is lower than the upper bound calculated by $P_{u,SLS}(t_p)$ and by $P'_{u,SLS}(t_p)$.

The simulation results demonstrate the effectiveness of the SLS approach, and of the lightweight implementation presented in previous sections. For example, the predicted probability of PA to suffer from blocking situation (i) is reduced from 3.0 % to 0.4 % at t_p=20 s in a controlled and predictable way.

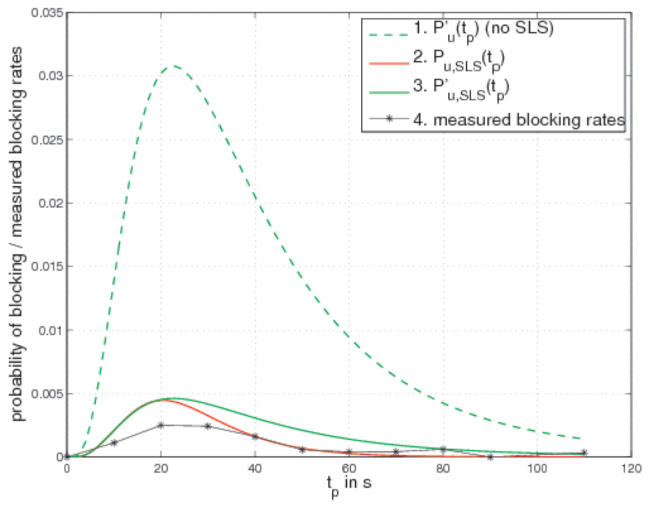


Figure 5.25: Reduction of blocking risk with SLS in the example MANET scenario if $LA(k, t_m)$=0.8 and $\#k$=4.

While the usefulness of the SLS is obvious, its benefit was only shown for the example scenario so far and not for a wide range of different scenarios. Such results are presented briefly in the following.

5.7.3 Additional Results

In this subsection, I will summarize some results of an extensive *ns2* simulation study carried out in [118].

Practical applicability of the SLS is demonstrated by showing that the success rate of *readSLS* operations executed by blocked transaction participants observed in simulations meets P_{ret} in different MANET scenarios. Results are obtained using *ns2* and the SLS lightweight implementation of [118].

To vary MANET scenarios, the network density is changed by assuming 10–50 nodes in $\mathcal{A}$, while the dimensions of $\mathcal{A}$ remain constant with 500 m * 500 m. Similar to the example MANET scenario of this work, nodes move according to the RWP mobility model, while the speed of nodes is uniformly distributed in an interval of 2–10 mps. The most important simulation parameters are listed in Table 5.1. For each network density, strict transactions with 1–8 participants are simulated with and without the SLS. In scenarios where the SLS is used, the success rate of executed *readSLS* operations is measured, while in scenarios where no SLS is used the proportion of recovery attempts with the coordinator that are successful within the first two message rounds is measured.

To measure a significant number of blocking situations, abort of strict transactions during the processing phase is artificially prevented, i.e. all transactions entered the decision phase. For commit processing, a vote time-out of Δ_{vo}=1 s is assumed.

For simulations, the following failure characteristics of nodes are assumed: node failures in $\mathcal{A}$ are exponentially distributed with parameter $\lambda_N = \lambda_E + \lambda_T$,

PARAMETER	VALUES
Participants n	1, 2, 4, 6, 8
Time-out in voting phase Δ_{vo}	1 s
Transaction initiation distance	1–2 hops
Node speed	2–10 mps
Radio range r_0	approx. 100 m
Dimensions of $\mathcal{A}$	500 m * 500 m
Total number of nodes $n_{\mathcal{A}}$	10, 20, 30, 40, 50
Parameter of $f_N(t)$ and $f_{RE}(t)$	$\lambda_N = 31/111200$, $\lambda_{RE} = 3600^{-1}$

Table 5.1: Simulation parameters of the lightweight approach.

while energy-related failures are distributed with parameter $\lambda_E = 3600^{-1}$ (batteries are expected to hold for 1 h) and technical failures are assumed to be rare with $\lambda_T = 1000800^{-1}$ (devices are expected not to suffer a failure for 278 h). Nodes recover from node failures after an exponentially distributed time with parameter $\lambda_{RE} = 3600^{-1}$. Hence, only recovery from energy-related failures is considered for node failure recovery and nodes are expected to remain disconnected from $\mathcal{A}$ for 3600 s (e.g. to recharge batteries). Other parameters of the MANET scenario, e.g. link model, MAC layer modeling etc. are chosen similar to the example scenario of this work and are listed in Appendix B.1.1.1. For transactions processing the SLS is asked to provide P_{ret}=0.7 for t_m=3600 s.

To verify that decision logs could be retrieved within t_m=3600 s at least at probability P_{ret}, the rate of *readSLS* operations that have been successfully executed within t_m is measured.

In Figure 5.26, the measured rates of successful *readSLS* executions are depicted. While only $P_{ret}(t_m)$=70 % was required, more than 90 % of the recovery attempts were successful. In contrast, when no SLS was embedded in transaction processing, the success rate of recovery attempts is smaller than 20 % and exceeds 10 % only at high network densities, as shown in Figure 5.26 by Curve 1. The reason for a higher success rate observed in simulations than required (P_{ret}=0.7), is given by the fact that a majority of nodes recovers at the beginning of t_m, where the log availability is higher than towards the end of the mission time. However, results show that the lightweight SLS implementation achieves the predicted log availability for blocked participants and therefore support applicability of the SLS approach in MANETs. For a more detailed description of this simulation study and further results proving applicability of the SLS in MANETs, I refer to [118].

5.7.4 Summary - Evaluation of the Lightweight Approach

In this section, I have shown by simulation that the probabilistic model underlying the SLS implementation accurately predicts the observed log availability and reduction in blocking risks measured in simulations. I have also shown that the probability of blocking (i) situations is reduced in simulations as predicted by the log availability model underlying the lightweight SLS approach. Additionally, simulation results for MANET scenarios with different densities have been presented showing that an implementation of the SLS as proposed in Section 5.4 compensates for blocking situations as predicted. I therefore conclude

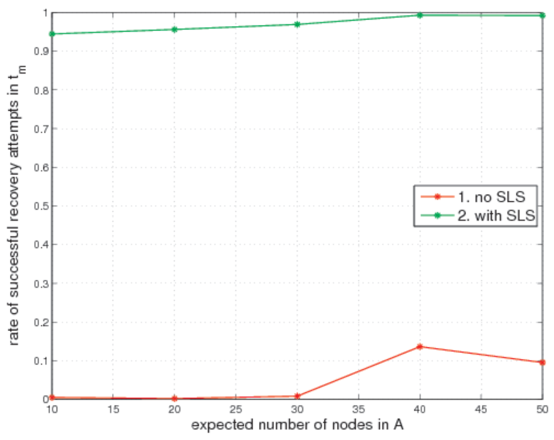


Figure 5.26: Experimental results of P_{ret} for different network densities.

that the lightweight implementation is applicable in MANETs and allows to reduce blocking risks as analytically predicted.

I argue that the general approach to using a small set of assistant nodes, which is determined using the availability model, results in a message-efficient scheme. This argument is supported by the results obtained from a cluster-based SLS implementation approach presented in the following section.

5.8 Overlay-based Implementation Approach

The implementation approach of the SLS presented above considers failure probabilities of nodes to decide on which nodes a decision log is placed on to achieve a desired log availability. The path probability P_{path} is used to estimate the reachability of an assistant node in the case of recovery. If a higher P_{path} can be assumed in a MANET scenario, fewer assistant nodes are required to guarantee the desired log availability.

A higher path probability is given if assistant nodes holding the decision log can be assumed to be in close hop distance (1–2 hops) to the recovering node. This is achieved if assistants are equally distributed over the network, i.e. assistant nodes are placed within $\mathcal{A}$ in a way that all areas and partitions are uniformly covered by neighborhoods carrying the decision log. A recovering node can then assume that at least one assistant resides in its current vicinity. Note that such a distribution is not assured with a flooding of $\mathcal{A}$ where a certain rate of nodes saves the decision log, because here a more random and not a uniform placement of the log is achieved, e.g. it is possible that all assistant nodes reside in one part of $\mathcal{A}$ at a time. To maintain a uniform placement of the decision log over time, the “fire-and-forget” approach of the lightweight SLS implementation is not adequate. To achieve a uniform placement of the decision log, an overlay structure is required allowing to control the dissemination of the decision log in different network areas and maintenance of its availability. Feasible overlay structures for the purpose of distributing the decision log equally within $\mathcal{A}$ are cluster overlays. Cluster structures are also used in MANETs for routing [156]

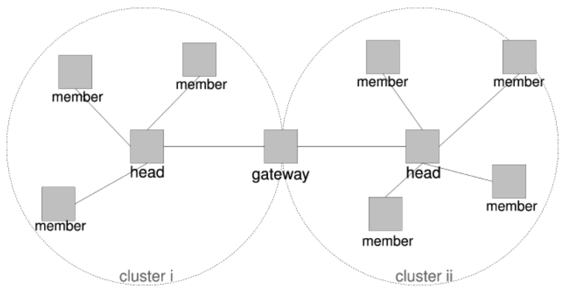


Figure 5.27: Cluster-based overlay network.

and for service discovery [95, 107].

In this section, I will present an alternative implementation approach of the SLS using an overlay structure for preservation and retrieval of the decision log. A calculation model to estimate the log availability achieved is not used, but generally a log availability close to 100 % is provided within t_m.

The idea of the alternative SLS implementation approach presented here is to use a cluster overlay to equally place the decision log within $\mathcal{A}$, i.e. the decision log is placed in a way that it uniformly resides in every part of $\mathcal{A}$ and achieves a high log availability with fewer assistant nodes than the lightweight approach.

The approach and its evaluation is described here briefly, while for a detailed description of the implementation and evaluation, I refer to [102]. I will show that the message overhead to create and maintain the cluster overlay is high and a flooding scheme where all nodes of the network save the log is cheaper. In fact this observation was a reason to primarily focus on the lightweight SLS implementation in this work.

In the following, creation and maintenance of the cluster overlay is described in short as well as the implementation of the *writeSLS* and *readSLS* operation. Finally, some simulation results are presented showing the success rate of the *readSLS* operation and the message load induced by cluster maintenance.

5.8.1 Creation and Maintenance of the Cluster Overlay

For the creation and maintenance of the cluster overlay, the well known clustering scheme presented in [159] is used. Every node within the network joins a cluster that is managed by a so-called *cluster head*. A cluster head periodically issues a beacon in single-hop range received by all cluster members. A node that is not a member of a cluster and receives the beacon of a cluster head informs the cluster head with a *join* message that it is now a member of its cluster. Nodes always join the bigger cluster if multiple beacons are received; if two cluster heads receive each other's beacons, the clusters are merged and the head of the bigger cluster becomes the new cluster head of the newly merged cluster. Nodes that receive beacons of two clusters can act as so-called *cluster gateways*. A detailed description of initiation and maintenance of the cluster overlay can be found in [102].

An example of the resulting overlay structure is depicted in Figure 5.27.

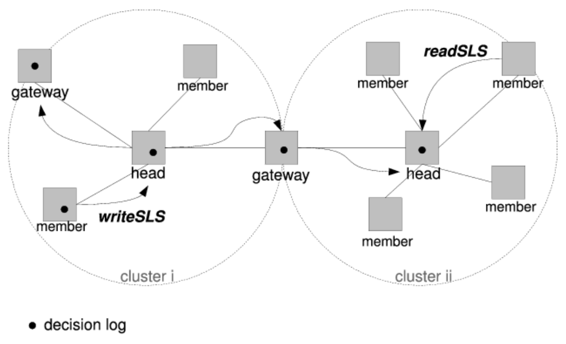


Figure 5.28: Basic scheme of the *writeSLS* and *readSLS* operation.

Based on this logical overlay, dissemination and retrieval of the decision log is implemented as described in the following.

5.8.2 Implementation of *writeSLS* and *readSLS*

The dissemination of the decision log within the *writeSLS* operation simply hands the decision log to the cluster head which the coordinator is currently connected to. The cluster head stores the decision log and forwards it to all other known cluster heads via its known cluster gateways. If a cluster head learns about the existence of another cluster head, decision logs of both cluster heads are synchronized. The distributed decision log contains the mission time t_m that terminates further log synchronization between cluster heads. In the implementation evaluated here, the decision logs are simply deleted after the mission time is exceeded. Figure 5.28 depicts the basic process of the *writeSLS* and *readSLS* operation.

To retrieve the decision log for a transaction, the *readSLS* operation executed by the uncertain participant simply sends an SSR message to its current cluster head. If the cluster head holds the decision log, it directly answers with an SLA message. If the cluster head does not hold the desired log, the SSR request is saved and answered when the cluster head receives the decision log through synchronization with other cluster heads. In this case, an answer is directly sent to the requesting participant nodes using the underlying multi-hop routing layer.

5.8.3 Evaluation

The SLS using a cluster overlay for dissemination and retrieval of the decision log was implemented as OSGi [3] component within the CoCoDa project [1] and evaluated using a MANET emulation system based on [12, 98] described in Appendix B.2.

The overlay-based approach was evaluated in a MANET with dimensions of 1000 m * 1000 m, while mobile nodes are assumed to move according to the RWP mobility model with speeds of 1–15 mps. Different values for $n_{\mathcal{A}}$ (20–70) are used to vary the network density in $\mathcal{A}$. Table 5.2 lists the most important parameters.

Parameter	Value
Routing algorithm	DSDV
Dimensions of $\mathcal{A}$	1000 m * 1000 m
Max. node speeds	1, 3, 6, 9, 12, 15
Number of nodes $n_{\mathcal{A}}$	20, 30, 40, 50, 60, 70
Radio Range r_0	100 m

Table 5.2: Simulation parameters of the cluster-based approach.

The overlay approach is compared to the probabilistic flooding scheme, which is also executed in the lightweight SLS implementation described in Section 5.6 if not enough assistant nodes are found until Δ_W. A recovering node then discovers the decision log at $P_{ret,max}$. In the probabilistic flooding scheme the probability of a log to be rebroadcasted depends on the number of neighbors of the receiving and sending node as well as on the network density as described in [102].

The questions answered by the evaluation of the overlay-based approach are: (i) whether this approach shows the same success rate as the approach based on probabilistic flooding; and (ii) whether the message overhead of the cluster-based strategy is acceptable.

Results

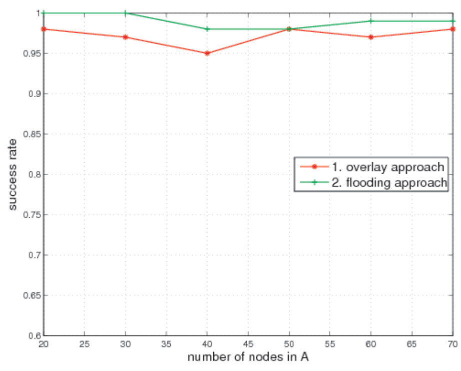
To answer the question of whether the overlay approach achieves a comparable success rate to the flooding-based approach, the overlay approach as well as the flooding-based approach was simulated with the parameters of Table 5.2. A retrieval operation is considered to be successful in the overlay approach if the cluster head of the cluster the recovering node is connected to, holds the desired decision log.

In the flooding approach, the *readSLS* operation is assumed to be successful if an SLA message is received after the SSR message is broadcasted in 1–2 hop range for the first time.

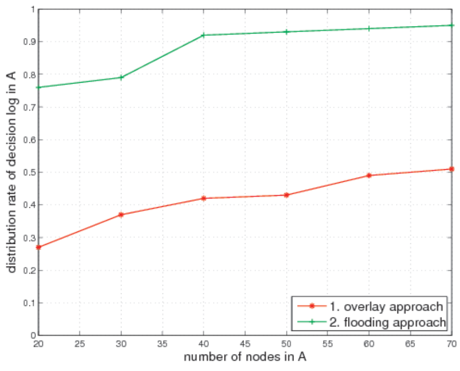
Figure 5.29(a) plots the measured rate of successful tests in both approaches for different network densities, while tests are randomly distributed within 1 h after execution of the *writeSLS* operation. It can be observed that both approaches show similar success rates, while the rate of the cluster approach is slightly lower.

The distribution rate describes the fraction of nodes that store the decision log. In the cluster-based approach, only cluster heads store the decision log, while in the probabilistic approach, all nodes that receive the decision log preserve the item. Figure 5.29(b) shows the distribution rate in the flooding and in the cluster approach. It is shown that the cluster-based approach achieves the same success rate with half of the distribution rate compared to the flooding approach. For example, at $n_{\mathcal{A}}$=40, 92 % of all nodes store the log item if the flooding approach is used, while only 41 % are used in the cluster approach. However, the lower distribution rate requires to maintain a cluster overlay.

In Figure 5.30(a), the total number of messages per node required to create and maintain the cluster overlay within a period of 1 h is presented. It can be observed that with increasing network density, the number of direct messages



(a) Success rate of different distributions for varying network densities.



(b) Distribution rate achieved.

Figure 5.29: Average success and distribution rates.

increases and a considerable message load of nearly 900 message for every node is reached at $n_{\mathcal{A}}$=70. Although this number is influenced by the clustering algorithm used, the basic rationale that increased network density requires more messages to organize nodes in a cluster overlay has to be true for all future clustering approaches. The reason is that every node has at least to announce its presence and membership to a cluster.

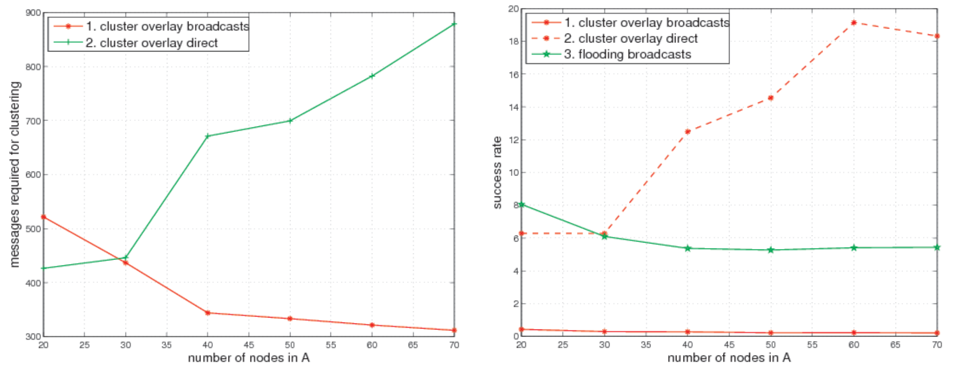
In contrast, the messages required by the flooding scheme are decreasing or remain constant for increasing network density as shown in Figure 5.30(b) by Curve 3. Here, the number of messages per node is always smaller than 8 for all network densities. The main reason for this effect is that to flood an area with fixed dimensions, a fixed number of broadcast messages is required to cover it, independently of the total number of nodes in this area. With increasing network density more nodes are reached by a single-hop broadcast and the message efficiency tends to increase, i.e. the effect of overhearing increases message efficiency. This effect should be observed with all flooding schemes.

Since blocking and therefore the execution of the *writeSLS* operation is a rare event as shown in Chapter 4, it can be concluded that the high message overhead of clustering is not beneficial in most MANET scenarios unless a cluster overlay is available for free, e.g. because it is also used for other purposes.

Additionally, I have shown for the scenarios analyzed here that even log dissemination within the existing cluster requires more messages than the flooding approach as shown by Curve 1 and Curve 2 in Figure 5.30(b). However, these results are mainly influenced by shortcomings of the underlying DSDV routing scheme used for simulation as described in [102] and therefore can not be considered to be generally valid.

5.8.4 Summary - Overlay-based SLS Implementation

While the SLS implementation proposed in Section 5.6 is based on a probabilistic model allowing for a “fire-and-forget” approach to preserve the decision log, the approach I presented in this section is based on the idea to constantly maintain the log availability within an overlay structure. In this approach, no calculation



(a) Messages required to create and maintain the cluster overlay structure

(b) Messages required for log dissemination by the cluster and flooding–based approaches.

Figure 5.30: Message load in the cluster-based and flooding approaches.

model is required because the distribution and maintenance scheme assures that the decision log is available within every cluster and, hence, every recovering node joining a cluster is able to retrieve the log.

The main advantage of the overlay-based approach is that a high availability of a decision log can be assured without any knowledge of node failure probabilities.

I showed that the overlay approach requires more messages than the worst case of the lightweight approach where the network is flooded. The high message load caused by cluster maintenance increases with $n_{\mathcal{A}}$, while the message efficiency of flooding schemes increases for higher network densities. I therefore argue that for most MANET scenarios the lightweight scheme will be more message efficient than an overlay-based strategy, unless an overlay is already available, e.g. for routing or service discovery. In fact, these results support the general observation in MANET research, where it is often found that lightweight schemes outperform schemes based on overlay structures.

5.9 Summary and Conclusion

This chapter presented an approach to compensate for blocking (i) and extended uncertainty (i) situations. With the SLS, I proposed an abstraction that is easy to integrate in strict and semantic transaction models and allows to compensate for blocking in a controlled manner. Two implementation approaches of the SLS abstraction have been proposed to demonstrate its applicability in MANETs.

The main focus in this work was on a lightweight implementation approach, which is based on a probabilistic model calculating the availability of a decision log for a recovering participant. I showed how this availability can be controlled by placing the log on a defined set of nodes within the network. Large parts of this chapter have been concerned with the description of this model. The model considers scenarios with and without multi-hop routing, individual node failure probabilities, and recovery probabilities of node failures. In multi-hop scenarios, a main idea is to precalculate the set of nodes the decision log is placed on in case of failures. Recovering participant can then directly contact these nodes, without being required to broadcast search requests within the network. In scenarios without multi-hop routing the decision log is found in close hop distance at a defined probability. Hence, only a limited part of the network has to be flooded with a search request by a blocked participant. Consideration of individual node failure probabilities allows to exploit individual stable nodes, which is of special interest in the real world. If recovery probabilities of node failures are known, the churn rate of nodes can be considered to achieve a high log availability with a small set of assistant nodes.

I showed that the calculation model of the lightweight approach accurately predicts log availabilities observed in simulations and that an implementation of this approach is straightforward. Additionally, I showed that the predicted reduction of blocking risks can be observed in simulations. I therefore conclude that the general SLS concept is applicable in MANETs and allows to reduce blocking risks in a controllable manner.

In fact, the observed reduction in blocking risks with the lightweight implementation are slightly higher than predicted, since calculations provide a safe lower bound for the probability of successful recovery from blocking. For more accurate predictions the distribution of participant recovery within the interval $[t_{ter,c}, t_m]$ has to be considered. I leave this enhancement of the prediction model to future research.

The overlay approach was presented only in brief and showed to be less message efficient than the lightweight approach, because cluster maintenance causes a high message overhead.

I argue that the general abstraction of the SLS, independently of its implementation is a useful contribution, since it allows to reason about the reduction of blocking risks in a probabilistic manner. Future implementation approaches, such as schemes based on cross-layer architectures are expected to integrate seamlessly into this abstraction. Additionally, the SLS concept is not limited to MANETs but can also be imagined in other dynamic environments where nodes frequently leave the system, such as in peer-to-peer systems in fixed networks.

However, the compensation for blocking achieved by the SLS requires that the *writeSLS* operation is executed. If the coordinator suffers a node failure before it is able to execute the *writeSLS* operation, the desired availability of the decision log is not reached and the predicted reduction of blocking is not met.

In the following chapter, I will present an approach on how to compensate for such situations using a backup coordinator.

Chapter 6

Backup Coordinator

The Shared Log Space (SLS) allows to guarantee successful recovery from blocking (i) and extended uncertainty (i) situations at probability P_{ret}. This guarantee is based on the *writeSLS* operation which has to be executed within the termination protocol of the coordinator. Hence, a main assumption in the discussion of the SLS and compensation for blocking (i) and extended uncertainty (i) in the previous chapter was that the execution of the *writeSLS* operation is not hindered by a node failure of the coordinator, i.e. the case of blocking or extended uncertainty caused by a node failure of the coordinator was not considered. This chapter is concerned with schemes to compensate for these situations.

In Chapter 4, I calculated the probability of blocking (ii) and extended uncertainty (ii) by Formulae (4.57) and (4.61). Thus, these formulae describe the probability of PA to suffer from blocking or extended uncertainty while no *writeSLS* operation is executed.

In this chapter, I will analyze approaches to compensate for the situation where a node failure of the coordinator completely removes the transaction decision from $\mathcal{A}$ and causes blocking (ii) or extended uncertainty (ii). This probability mainly depends on the probability of the coordinator to suffer a node failure. Hence, choosing a reliable node as commit coordinator is the obvious approach to decrease the probability of blocking (ii) and extended uncertainty (ii) situations. In fact, this is the approach followed by the XOpen standard [40]. Here, the commit coordinator is a designated site, exclusively concerned with commit processing, while the execution coordinator (the application) may reside on an arbitrary site. In fixed networks, this is a feasible approach, since communication between commit coordinator and execution coordinator is assumed to be reliable. However, in a MANET this is not the case and the probability of a communication failure between the commit coordinator and the execution coordinator would significantly increase the abort probability of a transaction.

In this work, I will not follow the approach to assign the execution role and commit role to different nodes. I will rather analyze the use of a second coordinator, a so-called backup coordinator (BC), in addition to the execution coordinator, henceforth called main coordinator (MC). If the MC suffers a node failure during commit, the BC would be able to terminate the transaction on behalf of the failed MC. This reduces the situations where no *writeSLS* operation is executed to the event where both coordinators suffer node failures simultaneously.

The main problem in delegating the commit decision to a second coordinator is that it induces an additional coordination problem. Agreement not only has to be reached among participants, but now also the BC must agree on the transaction outcome. Reaching agreement inevitably requires additional communication inducing a new source of defect. Hence, the main question to be answered is whether the increased reliability of coordinators actually results in a reduction of blocking situations or whether the increased risk for communication failures overcompensates the benefit of increased reliability of coordinators. General results being valid for all MANET scenarios cannot be found, given the the wide range of possible transaction and MANET scenarios. Again, a calculation model is the best contribution here to allow analysis of individual MANET and transaction scenarios. I will present such a model in this chapter.

While ACPs integrating a BC are known from literature, the main contribution of this chapter is: (i) the development of a calculation model estimating the reduction of blocking risk achieved by such protocols in a MANET, and (ii) enhancement of these protocols to use the SLS.

The chapter is structured as follows: as there exists some literature on backup coordinator schemes, I will first give a brief overview of related work in the field. Afterwards, I will describe a backup coordinator protocol which I will later enhance with an escalation scheme that executes the *writeSLS* operation in case of failures. The probability for the escalation scheme to be successful depends on when the BC is selected and integrated in commit processing. I show that in the strict transaction model, a BC is only beneficial if the BC is selected late, while early selection at time t_s may even increase the risk of blocking (ii) to occur in contrast to the situation where no BC is used. I show further that the greatest reduction in uncertainty is achieved in the strict model when the BC is selected late.

6.1 Multiple Coordinators

The approach to use multiple coordinators to increase the reliability of atomic transactions has been investigated by several scholars. The fundamental problem to consider is the trade-off between the increased reliability of coordination entities, i.e. the probability that a coordinator is reachable for a participant, and the induced message overhead to reach agreement among the group of coordinators.

Published approaches using multiple coordinators can be classified according to the point in time when additional coordinators are integrated in commit processing. They are either assigned to participants with the begin of the transaction and each participant executes a commit protocol with *its* assigned coordinator or additional coordinators are contacted as fallback only, if the main coordinator is unreachable.

Schemes belonging to the first class are for example [29, 6, 126], while in the protocols proposed in [82, 123, 124] participants execute the commit protocol with a single main coordinator, and additional backup coordinators are contacted only if a failure occurs. The approach of a reliable coordinator group proposed in [29] was already described in detail in Section 3.5.2.1.

In [6], a mechanism called *delegation of commitment* is proposed, which pairs each execution coordinator with a reliable commit coordinator. The execution coordinator prepares itself to commit the transaction and delegates the final commit decision to a more reliable commit coordinator similar to XOpen. A recovering participant can contact both coordinators to learn about the global decision.

A similar idea is proposed in [126], where the risk of coordinator failures causing permanent blocking of participants is reduced by using only designated reliable hosts, so-called *trusted hosts,* as commit coordinators. The protocol assures that every participant node participates in a commit protocol with a trusted coordinator, which is not necessarily the initiating coordinator.

Hence, these approaches are based on the idea to exploit especial reliable nodes as commit coordinator that are assumed not to fail. This is feasible in a traditional mobile environment where base stations can be used as commit coordinators. In MANETs it is a complex issue to discover stable nodes that can serve as BC as shown in [31].

The protocol proposed in [82, 123] uses multiple BCs as fallback coordinators if the MC fails and reduces to [124] when only a single BC is used.

However, none of the works cited above provides a calculation model that allows to estimate whether the scheme is beneficial in a certain transaction and MANET scenario or not. In the following, I will develop such a calculation model for the single backup coordinator with veto right protocol proposed in [124]. The protocol is then extended to use the SLS by proposing two so-called *escalation strategies* that trigger the *writeSLS* operation, while the probabilistic model is enhanced to calculate the probability that no *writeSLS* operation is executed.

In the following, the protocol proposed in [124] is described and compensation for blocking (ii) and extended uncertainty (ii) is analyzed in detail. Afterwards, the escalation strategies using the SLS are described.

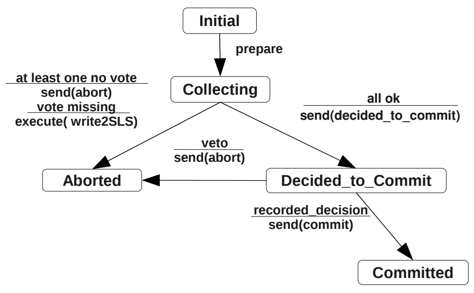
6.2 Single BC with Veto Right Protocol

When additional coordinators are used to terminate a transaction, agreement on the global outcome must not only be reached between participants but also among commit coordinators. Therefore, the atomic commit problem has also to be solved among commit coordinators, which requires the execution of an ACP among them.

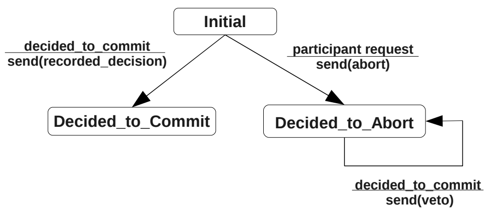
In the special case where only a single BC is used, 2PC allows for the last *agent optimization* [122], reducing 2PC to a single message exchange between the MC and the BC. This is the main idea of the single backup coordinator with veto right protocol proposed in [124].

The protocol induces a third phase required to coordinate the decision of the MC with the BC. The BC possesses a veto right and can therefore unilaterally decide to abort the transaction if there is a failure and will then veto a commit decision of the MC.

In the failure-free case, the protocol is processed as follows: similarly to the 2PC protocol, the coordinator initiates the protocol by issuing *prepare* messages to all participants and enters the *Collecting* state as depicted in Figure 6.1(a). Participants answer with *vote* messages and enter uncertainty. Note that in



(a) State diagram of MC (2PC).



(b) State diagram of BC.

Figure 6.1: Single backup coordinator protocol with veto right protocol [124].

the semantic transaction model, the acknowledgments of the last operations are interpreted as votes here. If all participants vote for commit, an additional phase comes into action. Based on the received commit votes, the MC decides on global commit. He immediately records its decision on its local stable log, sends a *decided_to_commit* message to the BC, and transits into the *decided_to_commit* state. The MC remains in the *Decided_to_Commit* state until a *recorded_decision* or *veto* message is received from the BC. When he receives a *decided_to_commit* message in *Initial* state, the BC saves this message to its local stable log and answers with a *recorded_decision* message. With reception of the *recorded_decision* message by the MC, both coordinators have agreed on global commit and the MC informs the participants about the decision for commit by issuing *commit* messages.

If at least one participant votes for abort, or more relevant in MANETs a vote is missing, the additional phase is not required. The MC decides on abort, preserves its decision on stable log, and sends *abort* messages to all participants. If a vote was missing, the MC also executes a *writeSLS* message to preserve its decision within the SLS as shown in Figure 6.1(a). An additional coordination phase is not required, because the BC will abort the transaction if he is contacted by a participant as shown in Figure 6.1(b).

From the participants view, the protocol behaves similar to 2PC, but the termination protocol of participants is enhanced in the following way: if the participant is uncertain and does not receive a final commit or abort message from the MC during a time-out period, participants can contact the BC by sending a *participant_request* message. Three situations can then occur:

- If the BC is in *Decided_to_Commit* state due to a *decided_to_commit* message from the MC, a commit message is sent to the uncertain participant.

- If the BC receives this message while in *Initial* state, the BC makes use of his veto right and aborts the transaction unilaterally. This is safe, since the MC cannot have informed participants to commit the transaction, as the *recorded_decision* message was not issued by the BC. The BC answers with an *abort* message and the uncertain participant can leave uncertainty.

- If the BC does not answer due to a node or communication failure, the participant remains blocked until the global decision can be obtained from the SLS, the MC, or by cooperative recovery.

Hence, the protocol compensates for the failure situation not considered in Chapter 5, where the MC suffers a node failure before the decision log can be written to the SLS. When the MC suffers a node failure during the collection phase, participants can contact the BC, who can then conclude the transaction by deciding on abort. The BC can now also preserve the global decision to the SLS. I will show in Section 6.3 how the *writeSLS* operation is integrated in the protocol.

However, while the protocol compensates for one blocking situation, it induces another one. It is therefore of interest whether the probability of blocking in this protocol is smaller than without a BC. To analyze these probabilities, the restart and termination protocol of the MC have to be considered. During restart or termination of the MC, the following situations can occur:

- Neither a *decided_to_commit* nor a *commit* or *abort* log is found. In this case, it is safe for the MC to abort the transaction, since the BC cannot have decided to commit but only on abort. If a *recorded_decision* log is found, agreement on commit was established between the MC and BC on the global commit decision.

- If a *decided_to_commit* message is found in the logs but no *recorded_decision* log, agreement on the global decision between the two coordinators was not reached. Depending on when the MC suffered a node failure, the BC either received a *decided_to_commit* message or not. Hence, the MC has to communicate with participants or the BC to learn about the state of the transaction, i.e. it is blocked. Note that participants are not affected by this uncertainty of the MC, as they can contact the BC right after the MC suffered from a node failure and do not have to await the restart of the MC.

From the perspective of a participant there are two situations, where a participant can suffer blocking (ii) or extended uncertainty (ii):

- If the MC and the BC suffer node failures simultaneously, i.e. no coordinator is available to derive and deliver the global decision. In this situation, participants remain blocked until the MC or BC becomes available again, or cooperative recovery is successful.

- If the MC issues a *decided_to_commit* message but does not receive the *recorded_decision* message from the BC, it cannot decide on the transaction and blocks. Participants that can only reach the MC and not the BC remain uncertain. The MC has to wait until communication with the BC is possible again.

The second situation is a new blocking situation introduced by the protocol. The probability of this situation to occur is mainly influenced by the probability for communication failures between the MC and BC.

To evaluate whether a BC is beneficial in a certain MANET scenario, it has to be computed whether the additional blocking state caused by a communication failure between the MC and BC overcompensates the decreased probability that both coordinators suffer a node failure simultaneously.

6.2.1 Integration of a BC in Strict Transactions

While the description above sketched only the basic structure of the BC protocol used, this section describes in more detail how the single backup coordinator with veto right protocol is integrated into the strict transaction model. It is described when and by which messages the BC is defined and made known to participants. These definitions are important, because the time the BC is selected significantly influences the probability of the two blocking situations described above.

In the strict model, the BC can be chosen and made known to participants at two times: (i) the BC can be selected right at the beginning of the transaction at time t_s; or (ii) with the initiation of the commit protocol at time t_p. I call (i) *early selection* of the BC, while (ii) is called *late selection* of the BC.

The use of a BC lowers the probability of the situation that no coordinator survives long enough to execute a *writeSLS* operation. The benefit in coordinator survivability is diminished by the need for a communication path between the MC and BC. If such a path is not available, the new blocking situation occurs and the benefit of a more reliable coordination entity is jeopardized. The risk of a communication failure between the MC and BC is especially high in the *early selection* scheme.

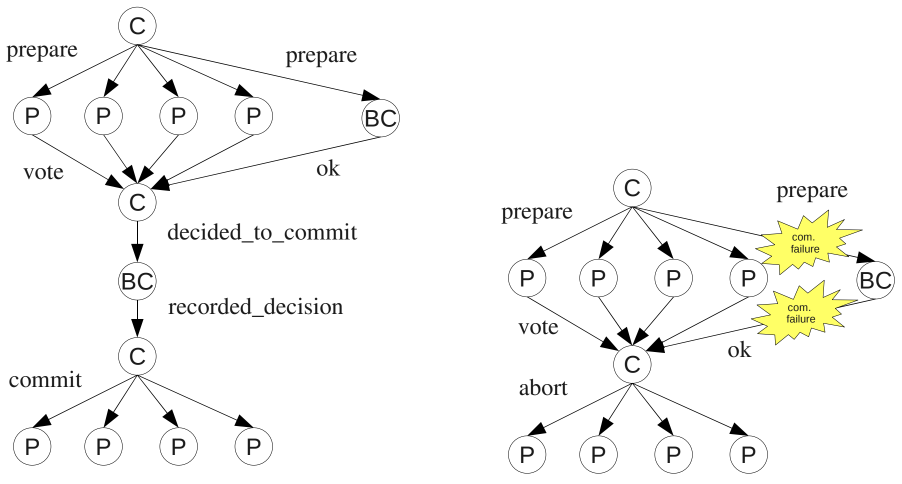
To reduce the risk of a *decided_ to_ commit* message to remain unanswered with *early selection*, I propose a reachability test to be integrated in the prepare phase. Hence, the MC also sends a prepare message to the BC. If the BC answers with an OK message, indicating its reachability, the probability of the *recorded_ decision* message not to be received by the MC, is given by $F_C(t_p..t_p + \Delta U)$. If the reachability test with the BC is omitted, the probability that the MC will not receive the *recorded_ decision* message and will therefore block, increases by the probability that a communication failure occurs that does not recover until t_p. This probability is calculated by $P_{c,nr}(t_p)$, and by $F_C(t_p)$ if link recovery is not considered.

Since $F_C(t_p..t_p+\Delta U) < F_C(t_p)$, the benefit of the reachability test is obvious. The reachability test within the prepare round avoids that every communication failure between the MC and BC that does not recover until t_p blocks the MC. The drawback of this approach is that every unsuccessful reachability test with the BC causes an abort decision of the complete transaction, although it can potentially be committed. Figure 6.2 depicts the integration of the proposed reachability check within the prepare phase.

If the BC is discovered and selected late, i.e. at t_p, the probability for a communication failure between the MC and BC is given by $F_C(\Delta U)$, which is considerably lower than in the early case. With late integration, the BC is made known to the participants by sending the BC identifier with the *prepare* message. Hence, a participant that moves into uncertainty is assured to know the address of the BC.

6.2.2 Integration of a BC in Semantic Transactions

In the semantic transaction model, the MC sends a *decided_ to_ commit* message to the BC after it has received an acknowledgment for the successful execution of the last operation of PA_{last} at time $t'_p + \Delta_{ex} + 2\delta_m$. When the MC receives a *recorded_ decision* message, participants are informed about the global decision.



(a) Failure-free case. (b) Failure case.

Figure 6.2: Reachability test in the *early selection* scheme.

Hence, the protocol is similar to the strict case with the difference that no explicit prepare messages are sent.

A late integration of the BC, as in the strict transaction model, is not possible in the semantic model. The reason is that participants move into uncertainty at individual times t_o in $[t_s, t'_p]$, and for meaningful BC integration, a participant must know the address of the BC before moving into uncertainty. As the BC must be defined and made known to participants at t_s, a reachability test as proposed for early integration in the strict model described above is mandatory. The probe message testing reachability of the BC is best sent in parallel with the last operation of PA_{last}.

The blocking situations are the same as described above, while the probabilities of the two blocking situations differ in the semantic model. The probability of the MC and BC to experience a simultaneous node failure while participants are uncertain is significantly higher than in the strict model. This is due to the fact that the uncertainty periods of participants are considerably longer in the semantic model as discussed in Section 4.6.

6.2.3 Blocking with BC in Strict Transactions

In the following, I will develop a calculation model to estimate the reduction of the probability for blocking (ii) if a BC is integrated into transaction processing for the strict model.

6.2.3.1 Early Integration of BC

With *early selection*, the BC is chosen at transaction initialization time t_s. I assume that a node in 1–2 hop distance to the coordinator and to all participants is chosen as BC. In [31], I proposed several schemes how to discover a BC with such properties.

To reduce the blocking risk of the MC, the reachability test is executed as described above. I distinguish between standard blocking situations due to a coordinator failure and the additional situation where the MC is alive but blocked. Again, I calculate the probability of blocking from the perspective of PA. Four cases may lead to a standard blocking situation:

(i) At least one unrecognized failure of a node in PA_{other} occurred until t_p. Then PA is blocked if the MC and BC are unreachable at recovery time. As the MC must have survived until t_p, only a node failure within $[t_p, t_p + \Delta U_{max}]$ needs to be considered. The link between the BC and PA may have been broken within the whole interval $[t_s, t_p + \Delta U_{max}]$. Then $BC1_e(t_p)$ describes the probability that PA blocks in case (i).

$$BC1_e(t_p) \quad = \quad P_{U_{max}}(t_p) \cdot CF_{Umax}(t_p) \cdot F(t_p + \Delta U_{max}) \tag{6.1}$$

where $P_{U_{max}}(t_p)$ is given by Formula (4.25), as introduced in Section 4.5.1, and $CF_{Umax}(t_p)$ is given by $F_N(t_p..t_p + \Delta U_{max})$, as described in Section 4.6.1.

(ii) No node in PA_{other} fails, but the BC suffers a communication failure with the MC until t_p. In this case, the reachability test fails and MC decides on abort at $t_p + \Delta U_{max}$. PA is blocked if the MC fails in $[t_p, t_p + \Delta U_{max}]$ and, additionally, the BC fails within $[t_p, t_p + \Delta U_{max}]$ or its link to PA breaks in $[t_s, t_p + \Delta U_{max}]$. This probability is given by $BC2_e(t_p)$.

$$\begin{aligned} BC2_e(t_p) \quad = \quad & \big[1 - F(t_p)\big]^{n-1} \cdot F_C(t_p) \cdot CF_{Umax}(t_p) \\ & \cdot \big[F_N(t_p..t_p + \Delta U_{max}) + F_C(t_p + \Delta U_{max})\big] \end{aligned} \tag{6.2}$$

(iii) No node in PA_{other} fails, but the BC encounters a node failure until t_p. Again, the reachability test fails and the transaction is aborted. A node failure of the MC leads to blocking of PA, as the BC has already failed and is not reachable. This probability is computed by $BC3_e(t_p)$.

$$BC3_e(t_p) \quad = \quad \big[1 - F(t_p)\big]^{n-1} \cdot F_N(t_p) \cdot CF_{Umax}(t_p) \tag{6.3}$$

(iv) Neither a node in PA_{other} nor the BC fails until t_p. A coordinator's node failure in $[t_p, t_p + 2\delta_m]$ leads to a blocking situation in the case that the BC has also suffered failure in this interval.

$$BC4_e(t_p) = \big[1 - F(t_p)\big]^{n-1} \cdot F_N(t_p..t_p + 2\delta_m) \cdot F(t_p..t_p + 2\delta_m) \tag{6.4}$$

The MC is blocked if the *recorded_ decision* message of the BC is awaited but not received. A presumption for this to happen is that no failure of any node has occurred until t_p. I distinguish two cases that actually lead to blocking of the MC.

In the first case called A, the BC encounters a node failure after sending the acknowledgment of the reachability test and before sending its *recorded_ decision* message (within interval $[t_p + \delta_m, t_p + 3\delta_m]$). The MC is then blocked and the BC is not reachable for PA.

In the second case called B, the BC suffers a communication failure with the MC in $[t_p + 2\delta_m, t_p + 4\delta_m]$. If additionally the communication link between the BC and PA breaks until $t_p + \Delta U_{max}$, the MC is blocked and PA cannot reach the BC. Since events A and B are not independent, the probability that either

A or B occurs has to be computed as $P(A) + P(B) - P(A) \cdot P(B)$. I therefore derive $BC5_e(t_p)$.

$$\begin{aligned} BC5_e(t_p) &= \left[1 - F(t_p)\right]^{n-1} \cdot \left[1 - F_N(t_p + \Delta U_{max})\right] \\ &\cdot \Big[F_N(t_p + \delta_m..t_p + 3\delta_m) + F_C(t_p + 2\delta_m..t_p + 4\delta_m) \\ &\cdot F_C(t_p + \Delta U_{max}) - F_N(t_p + \delta_m..t_p + 3\delta_m) \\ &\cdot F_C(t_p + 2\delta_m..t_p + 4\delta_m) \cdot F_C(t_p + \Delta U_{max})\Big] \end{aligned} \tag{6.5}$$

The resulting probability that a participant is blocked with early selection of a BC is then given by $P_{v,BCe}(t_p)$.

$$\begin{aligned} P_{v,BCe}(t_p) &= PA_{Umax}(t_p) \cdot \big[BC1_e(t_p) + BC2(_e t_p) + BC3(t_p) \\ &+ BC5_e(t_p)\big] + PA_{Umin}(t_p) \cdot BC4_e(t_p) \end{aligned} \tag{6.6}$$

where the probability that PA does not encounter any failure until $t_p + \Delta U_{min}$ is given by $1 - F(t_p + \Delta U_{min})$ and denoted by $PA_{Umin}(t_p)$. $PA_{Umax}(t_p)$ is defined analogously.

6.2.3.2 Late Selection of BC

With late selection, it is assured that the BC is in 1–2 hop distance of the MC and participants at t_p, because the BC is chosen at t_p. Hence, the probability that the BC is unreachable for PA in case the MC fails is lowered.

The computation of this probability is analogous to 6.2.3.1, with the difference that cases (ii) and (iii) from above do not need to be considered, because the BC is guaranteed to be available at time t_p. The increased probability that BC and PA can communicate has the greatest influence on the final result.

In case (i), only a failure of the BC in ΔU_{max} has to be considered:

$$BC1_l(t_p) = P_{Umax}(t_p) \cdot CF_{Umax}(t_p) \cdot F(\Delta U_{max}) \tag{6.7}$$

The probability of simultaneous node failures of the MC and BC if all nodes vote (case (iv)) is calculated by $BC4_l(t_p)$.

$$BC4_l(t_p) = \left[1 - F(t_p)^{n-1}\right] \cdot F_N(t_p..t_p + 2\delta_m) \cdot F(2\delta_m) \tag{6.8}$$

Blocking of the MC happens either if the BC suffers a node failure within $2\delta_m$ (situation A) or when a communication failure occurs until $4\delta_m$ between the MC and BC, while PA cannot reach the BC due to a communication failure in ΔU_{max} (situation B). Again, the events A and B are not independent, and therefore the probability of A or B to happen is given by $BC5_l(t_p)$.

$$\begin{aligned} BC5_l(t_p) &= \left[1 - F(t_p)^{n-1}\right] \cdot \left[1 - F_N(t_p + \Delta U_{max})\right] \\ &\cdot \big[F_N(2\delta_m) + F_C(\Delta U_{max}) \cdot F_C(4\delta_m) \\ &- F_N(2\delta_m) \cdot F_C(\Delta U_{max}) \cdot F_C(4\delta_m)\big] \end{aligned} \tag{6.9}$$

The probability of blocking (ii) with late selection of the BC is now given by:

$$P_{v,BCl}(t_p) = PA_{Umax} \cdot \left[BC1_l(t_p) + BC5_l(t_p)\right] + PA_{Umin} \cdot BC4_l(t_p) \tag{6.10}$$

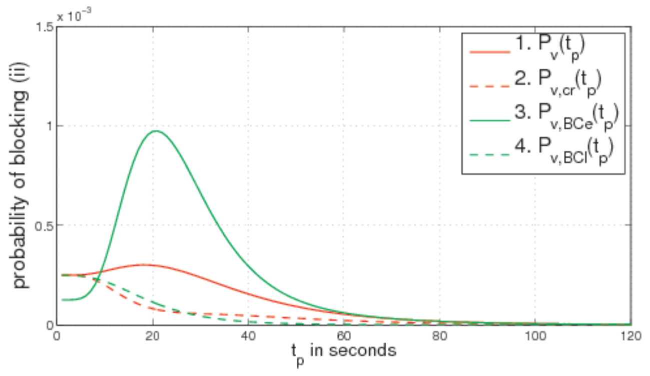


Figure 6.3: Blocking Probability with BC in the example MANET scenario with n=3 and Δ_{vo}=1.

6.2.3.3 Analytical Predictions

To demonstrate the reduction of the probability of blocking (ii) with a BC, I present blocking risks with early and late integration of a BC as well as risks without a BC for the example MANET scenario of this work in Figure 6.3. These probabilities are also compared to the situation where no BC but cooperative recovery is used to compensate for blocking (ii) as calculated by Formula (4.54) in Section 4.6.1.

The interesting observation in the example scenario is that early integration of a BC results in a higher blocking probability than predicted by $P_v(t_p)$, as shown by Curve 3 in Figure 6.3. The lower blocking risk for t_p<9 s with early selection is diminished by an increased abort rate caused by an abort decision if the reachability test is unsuccessful. Hence, early integration of a BC is not beneficial in the considered scenario, but even increases the probability of PA to suffer blocking.

In contrast, late selection of the BC depicted by Curve 4 of Figure 6.3, results in a reduced blocking probability compared to the situation where no BC is used. The BC scheme with late selection shows a similar reduction of blocking (ii) risks like cooperative recovery does, as shown by Curve 4.

Hence, for the example MANET scenario and similar scenarios it can be concluded that early integration is critical for most scenarios where no extra stable BC nodes can be assumed. Therefore late integration should be favored if possible. The results also underline the importance of calculation models as provided in this work, because each scenario shows different characteristics and the decision whether a scheme is beneficial is hard if not impossible without such a model. Such a model is particularly important if the behavior of the approaches should be analyzed when more reliable nodes, such as base stations, are available. I will analyze the use of an especial stable BC node below.

6.2.4 Blocking with BC in Semantic Transactions

For deriving the probability that PA suffers extended uncertainty with a BC in the semantic model, I again consider the intervals $[t_s, t'_p]$ and $[t'_p, t_u]$.

In the semantic model, only early integration has to be considered since late integration is not reasonable. Extended uncertainty caused in $[t_s, t'_p]$ can be resolved by the BC if it is reachable for a participant PA at $t_u + \delta_m$. Hence, for this interval I derive the probability that PA suffers extended uncertainty by $BC'1(t'_p)$.

$$BC'1(t'_p) = P^*_v(t'_p) \cdot F(t_u + \delta_m) \tag{6.11}$$

where $P^*_v(t'_p)$ is given by Formula (4.61) describing the probability of extended uncertainty (ii) without BC and without cooperative recovery.

In the second interval $[t'_p, t_u]$, the reachability test has to be taken into account. The MC checks reachability of the BC by issuing a probe message at time t'_p. An answer is awaited until $t'_p + 2\delta_m + \Delta_{ex}$, while I assume a time-out $\Delta_{to} = \Delta_{ex}$. Thus, at t_u the MC knows the global decision as well as whether the reachability test was successful. To derive the desired probability for the interval $[t'_p, t_u]$, multiple situations for both outcomes of the reachability test have to be considered. Like in Subsection 6.2.3, I distinguish between standard extended uncertainty and the situation where the MC is reachable but blocked. I first consider the three standard cases:

(i) If the reachability test fails because the BC suffers a node failure during $[t_s, t'_p + \delta_m]$, PA remains uncertain if the MC fails in the interval $[t'_p, t_u + \delta_m]$. The probability of this event is given by $BC'2(t'_p)$.

$$BC'2(t'_p) = \left[1 - P_{o>f}(t'_p)\right]^{n-1} \cdot F_N(t'_p + \delta_m) \cdot F_N(t'_p..t_u + \delta_m) \tag{6.12}$$

(ii) If the reachability test fails because a communication failure occurs between the MC and BC in $[t_s, t_p + 2\delta_m]$, PA remains uncertain if the MC suffers a node failure in the interval $[t_p, t_u + \delta_m]$ and the BC is not reachable for PA due to a node failure in $[t'_p, t_u + \delta_m]$ or a communication failure between the BC and PA in $[t_s, t_u + 2\delta_m]$. This probability is calculated by $BC'3(t'_p)$.

$$\begin{aligned} BC'3(t'_p) \quad = \quad & \left[1 - P_{o>f}(t'_p)\right]^{n-1} \cdot F_C(t'_p + 2\delta_m) \cdot F_N(t'_p..t_u + \delta_m) \\ & \cdot \left[F_N(t'_p..t_u + \delta_m) + F_c(t_u + 2\delta_m)\right] \end{aligned} \tag{6.13}$$

(iii) If no failure occurs until t'_p, PA remains uncertain if the MC suffers node failure in $[t'_p, t_u]$ and the BC is not reachable because of a node failure in $[t'_p, t_u]$ or a communication failure with PA in $[t_s, t_u + 2\delta_m]$. The probability for this situation to happen is calculated by $BC'4(t'_p)$.

$$\begin{aligned} BC'4(t'_p) \quad = \quad & \left[1 - P_{o>f}(t'_p)\right]^{n-1} \cdot F_N(t'_p..t_u) \\ & \cdot \left[F_N(t'_p..t_u) + F_C(t_u + \delta_m)\right] \end{aligned} \tag{6.14}$$

The situation when PA remains uncertain although it can reach the MC but the MC is blocked is derived by the same considerations as in Formula (6.5). The MC is blocked if the *recorded_ decision* message of the BC is awaited but not received. This situation can only occur if all participants votes have been received. This probability is given by $\left[1 - P_{o>f}(t'_p)\right]^{n-1}$. Again, two situations can lead to blocking of the MC. In the first situation, the BC suffers a node failure in interval $[t'_p + \delta_m, t_u]$ and, hence, answered the reachability test but could not send a *recorded_ decision* message. The second situation leading to blocking of the MC and PA occurs if the BC suffers a communication failure with the MC in the interval $[t'_p + 2\delta_m, t_u]$ and, additionally, with PA until t_u. Since probabilities of both events are not independent I derive $BC'5(t'_p)$ as follows.

$$\begin{aligned} BC'5(t'_p) \quad = \quad & \left[1 - P_{o>f}(t'_p)\right]^{n-1} \cdot \left[F_N(t_p + \delta_m..t_u)\right. \\ & +F_C(t'_p + 2\delta_m..t_u) \cdot F_C(t_u) - F_N(t'_p + \delta_m..t_u) \\ & \left.\cdot F_C(t_p + 2\delta_m..t_u) \cdot F_C(t_u)\right] \end{aligned} \tag{6.15}$$

The probability of PA to suffer extended uncertainty (ii) if a BC is used is now given by $P^*_{v,BC}(t'_p)$:

$$\begin{aligned} P^*_{v,BC}(t'_p) \quad = \quad & BC'1(t'_p) + \left[1 - F(t_u)\right] \cdot \left[BC'2(t'_p) + BC'3(t'_p)\right. \\ & \left.+BC'4(t'_p) + BC'5(t_p)\right] \end{aligned} \tag{6.16}$$

Analytical Predictions

Figure 6.4 depicts the probabilities of extended uncertainty (ii) with and without a BC as predicted by $P^*_u(t'_p)$ and $P^*_{u,BC}(t'_p)$. Additionally, the risk of extended uncertainty (ii) with cooperative recovery given by $P^*_{v,cr}(t'_p)$, which was derived in Formula (4.63), is presented.

Since only early integration of a BC is feasible in the semantic model, calculations as presented here are required to decide whether integration of BC is feasible at all.

Curve 2 in Figure 6.4 shows the probability of extended uncertainty (ii) if a BC is used. It can be observed that an early selection of the BC results only in a small reduction of extended uncertainty (ii) for some t'_p in the example MANET scenario of this work. Especially for $t'_p < 30$ s, a reduced uncertainty risk is observed, e.g. 0.2 % with BC, compared to 0.45 % without a BC at t'_p=16 s. However, for transaction size t'_p=28 s the blocking risks with BC is slightly higher than without. Hence, semantic transactions require special caution and a model as proposed here to decide whether it makes sense to integrate a BC for a given transaction, i.e. to identify the spectrum of t'_p where a BC is useful.

Generally, cooperative recovery shows to be more effective than a BC scheme as shown by Curve 3 in Figure 6.4. Especially for short transactions with $t'_p < 40$ s this scheme compensates for blocking more effectively than using a BC with veto right in the example scenario of this work.

6.2.5 Summary - Blocking Risks with a BC

In this section, I showed how a BC is integrated in the strict and in the semantic transaction model used in this thesis. I presented a calculation model to estimate the reduction of blocking (ii) and extended uncertainty (ii) situations if a BC is integrated in commit processing.

By applying the model to the example MANET scenario of this work, I showed that a BC is not necessarily beneficial if it is integrated early. For some t_p, early integration of a BC even increases the probability of blocking and extended uncertainty, because an additional blocking situation is induced by the protocol. The probability of this new blocking risk caused by a communication

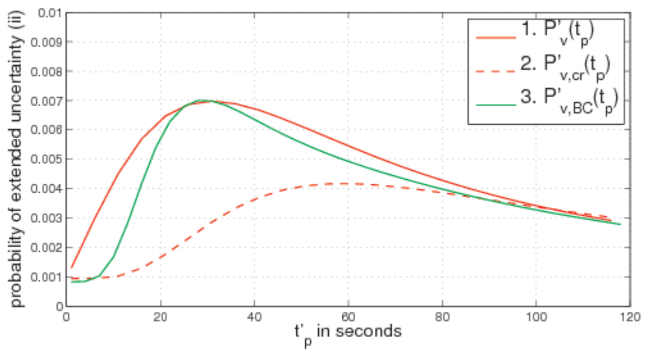


Figure 6.4: Probability of extended uncertainty (ii) with BC, n=3, and Δ_{vo}=1 s.

failure between MC and BC can overcompensate the increased reliability of coordinators with early integration. This is especially problematic in the semantic model where early integration of a BC is the only option. Here, a calculation model as presented in this work is crucial to identify the spectrum of t'_p where integration of a BC is beneficial.

In contrast, late integration reduces the risk of blocking (ii) situations at a similar rate as cooperative recovery.

However, this section dealt with the benefits of a BC for participants that have not suffered from a failure. To assure the availability of the transaction decision at P_{ret} as postulated in Chapter 5, the BC has to take over responsibility to execute the *writeSLS* operation to preserve the decision log at P_{ret} for possibly failed participants.

The following section will enhance the BC commit protocol presented here with an escalation strategy to integrate the SLS in BC commit processing.

6.3 SLS Escalation Strategy

While the BC protocol as discussed above compensates for the risk that a participant suffers blocking caused by a node failure of the coordinator, the protocol is not related to the SLS. To completely take over the duties of the MC in the context of this work, the BC also has to execute the *writeSLS* operation to ensure that availability of the decision log is maintained at P_{ret}.

In this section, I analyze the use of the BC to ensure the execution of the *writeSLS* operation in case of failures. Hence, this section closes the gap for a comprehensive probabilistic model by adding calculations to predict the probability that blocking (ii) or extended uncertainty (ii) occurs and no *writeSLS* operation is executed. The resulting model allows to predict the probability of the situation that an uncertain participant will not be able to retrieve the decision log from the SLS in general, i.e. both blocking situations are considered by the probabilistic model of this work.

Generally, execution of the *writeSLS* operation with an abort decision has to be considered by the BC if no *decided_to_commit* message from the MC is received. In this case, the MC has either suffered node failure or communication failure with the BC. In both situations, participants are uncertain and cannot

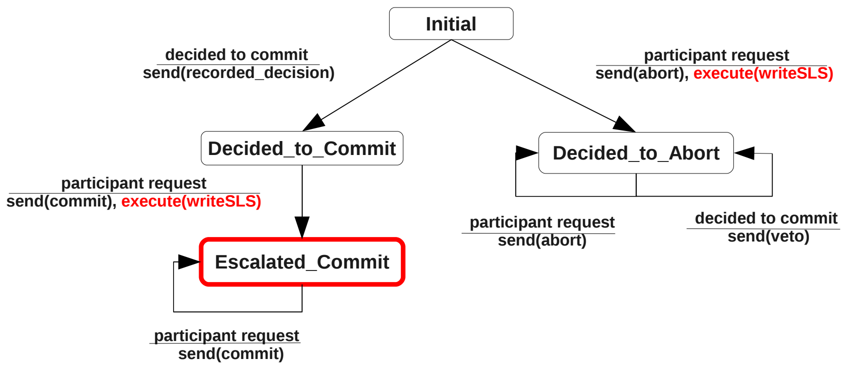


Figure 6.5: State diagram of BC in the participant-based escalation scheme.

learn the transaction decision from the MC and, more importantly, the MC cannot execute the *writeSLS* operation, because he is either uncertain about the decision (if commit was decided) or failed.

There are two options to trigger the execution of the *writeSLS* operation by the BC: (i) based on a time-out on reception of the *decided_ to_ commit* message and (ii) triggered by a request of an uncertain participant. I call the first approach *time-out-based* escalation, while the second approach is called *participant-based* escalation. In the following, I will describe both approaches and provide a probabilistic model for the time-out-based escalation scheme.

6.3.1 Participant-based Escalation

If a participant request is used as failure indicator to execute the *writeSLS* operation, the following modifications to the single backup coordinator with veto right protocol are required.

When the BC is contacted by a participant, it responds to the participant with its state information (commit or abort) and, additionally, executes the *writeSLS* operation. A new state called *Escalated_ Commit* is integrated, to prevent the BC from calling the *writeSLS* operation multiple times if additional participant requests arrive in the *decided_ to_ commit* state.

Figure 6.5 depicts the resulting protocol. I call this protocol *participant-based escalation strategy*. The state diagram of the MC remains unchanged and embeds the *writeSLS* operation as described in Section 6.2. The restart protocol of the MC is now changed in the way that the SLS is used to learn about undecided transactions, i.e. execution of the *readSLS* operation is embedded as depicted in Figure 6.6.

The main drawback of the participant-based escalation strategy is that participants must be able to reach the BC. This is not assured, as the following situation may occur: assume that the MC is blocked (it issued a *decided_ to_ commit* message but did not received a *recorded_ decision* message yet) or the MC suffered a node failure during the collection phase, while the BC is isolated in a partition. In this case, the BC cannot be reached by participants and will therefore not execute the *writeSLS* operation.

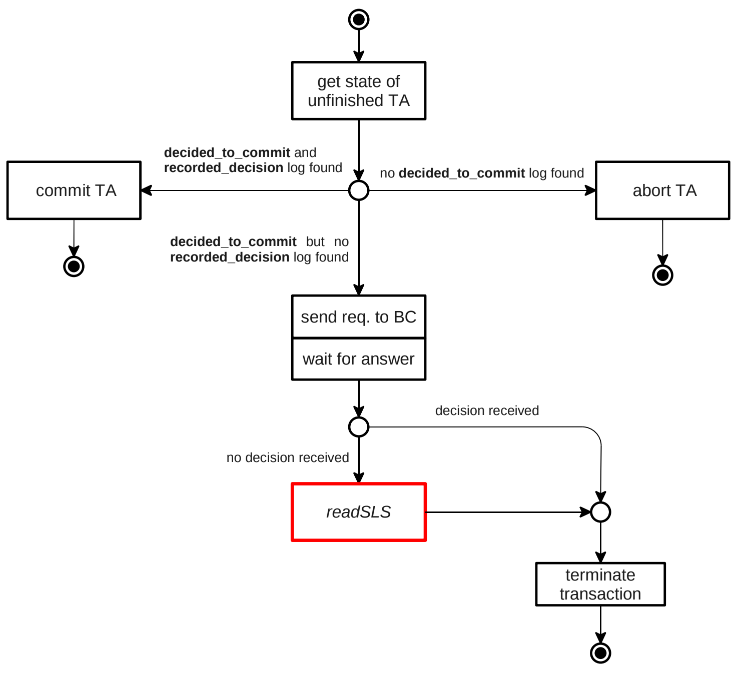

get state of unfinished TA
decided_to_commit and recorded_decision log found
commit TA
no decided_to_commit log found
abort TA
decided_to_commit but no recorded_decision log found
send req. to BC
wait for answer
decision received
no decision received
readSLS
terminate transaction

Figure 6.6: Restart protocol of MC considering the BC and the SLS.

Another drawback of the participant-based approach is that a participant that reaches the BC and learns about the abort decision does not need to query the SLS anymore. Hence, the *writeSLS* operation is of benefit only for other uncertain participants that either cannot reach the BC or are disconnected from $\mathcal{A}$. In scenarios with one to three participants, the probability of the existence of additional uncertain participants is low. Given these considerations, I will not provide a calculation model for the participant-based escalation strategy and refer to [24] for a simulative evaluation of this approach.

6.3.2 Time-out-based Escalation

If a time-out is used to trigger the *writeSLS* operation at the BC, I propose the so-called *time-out-based escalation* protocol. The main problem to solve is to initialize the time-out trigger securely. In the following, the *time-out-based escalation* scheme is analyzed for strict and semantic transactions. For both models, I will present a calculation model to estimate the probability of the event that no *writeSLS* operation is executed by any coordinator while PA is blocked.

6.3.2.1 Strict Atomicity

With late selection of the BC, the latest point in time to initialize the time-out trigger securely at the BC is when the MC enters the collection phase. Therefore, the prepare round is best used for initializing the time-out for the

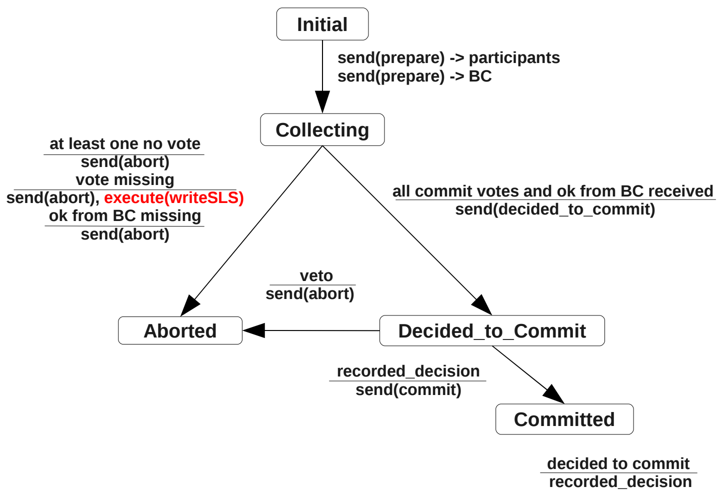


Figure 6.7: State diagram of MC with time-out-based escalation strategy.

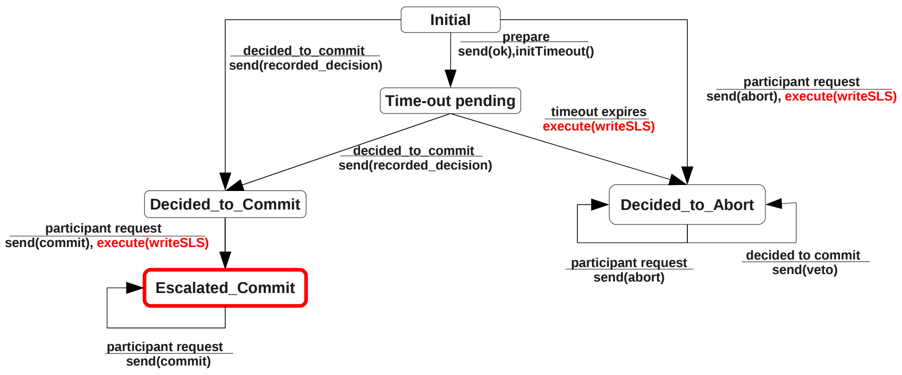


Figure 6.8: State diagram of BC with time-out-based escalation strategy.

writeSLS operation. Additionally, the *writeSLS* operation can be triggered by a participant request, if the prepare message of the MC did not arrive at the BC. However, this event is negligible, since the BC is selected at t_p in 1–2 hop distance. The resulting protocol is depicted in Figures 6.7 and Figure 6.8 while the restart protocol of the MC is the same as with participant-based escalation depicted in Figure 6.6.

The described protocol obviously reduces the probability that no *writeSLS* operation is executed if participants are uncertain. In the following, I will present calculations to predict this probability for arbitrary MANET scenarios and strict transactions, while the effect of more stable BC nodes is considered. Note that these calculations are based on similar considerations as the derivation of the blocking (ii) risks with late BC selection presented in Subsection 6.2.3.2.

To calculate the probability of PA to suffer blocking (ii) and that no *writeSLS* operation is executed, the two standard blocking situations where the MC and

BC are not reachable for PA as well as the situation where the MC is blocked have to be considered. The two standard blocking situations that may occur are:

(i) If at least one node of PA_{other} suffers an unrecognized failure within $[t_s, t_p]$, PA blocks at recovery time if the MC and BC are unreachable. No *writeSLS* operation is executed if both MC and BC suffer a node failure. Hence, the probability of the situation where PA is blocked due to a node failure of the MC and no *writeSLS* operation is executed is given by $BF1(t_p)$.

$$BF1(t_p) = P_{U_{max}}(t_p) \cdot CF_{Umax}(t_p) \cdot F_{N,BC}(\Delta U_{max}) \tag{6.17}$$

(ii) The same situation has to be considered in case that all participants vote and ΔU is of size ΔU_{min}, given by $BF2(t_p)$.

$$BF2(t_p) = P_{U_{min}}(t_p) \cdot CF_{Umin}(t_p) \cdot F_{N,BC}(\Delta U_{min}) \tag{6.18}$$

For the situation where the MC is alive but blocked, no *writeSLS* operation is executed: (i) if the BC suffers a node failure while the MC is blocked (I call this situation A), or (ii) if the *recorded_decision* message is lost, called situation B. The probability of a node failure of the BC is given by $F_N(2\delta_m)$, while the probability for the *recorded_decision* message to be lost is given by $F_C(2\delta_m..4\delta_m)$. In situation B, PA is only blocked if the BC is unreachable, which is given by $F_C(\Delta U_{max})$. Note that A and B are not independent. Hence, the probability of either A or B to happen is given by $BF3(t_p)$.

$$\begin{aligned} BF3(t_p) \quad = \quad & \left[1 - F(t_p)\right]^{n-1} \cdot \left[1 - F_N(t_p + \Delta U_{max})\right] \\ & \cdot \left[F_{N,BC}(2\delta_m) + F_C(2\delta_m..4\delta_m) \cdot F_C(\Delta U_{max})\right. \\ & \left. - F_{N,BC}(2\delta_m) \cdot F_C(2\delta_m..4\delta_m) \cdot F_C(\Delta U_{max})\right] \end{aligned} \tag{6.19}$$

The probability that PA suffers a blocking (ii) situation and no *writeSLS* operation is executed is now given by

$$P_{no\,writeSLS}(t_p) = PA_{Umax} \cdot \left[BF1(t_p) + BF3(t_p)\right] + PA_{Umin}(t_p) \cdot BF2(t_p) \tag{6.20}$$

Predictions

Figure 6.9 depicts the probability calculated by (6.20) for the example MANET scenario of this work.

While for Curve 1 in Figure 6.9, common node failure probabilities for all nodes in $\mathcal{A}$ and therefore also for the BC node are assumed (with $F_L(t)$ to be exponentially distributed with $\lambda = 1800^{-1}$), Curve 2 depicts the situation where a more stable node is chosen as BC with $F_L(t)$ exponentially distributed with parameter $\lambda = 5400^{-1}$. The expected positive effect of increased BC reliability is clearly reflected by the calculation results given in Figure 6.9. The decisive factor for the situation that PA suffers a blocking (ii) situation and no *writeSLS* operation is executed, is the situation where the MC is alive but blocked and the BC cannot execute the *writeSLS* operation due to a node failure. As with increasing t_p the probability that all participants vote becomes smaller, the probability that commit is decided and the MC blocks also decreases.

The probability of the situation to occur where PA suffers blocking caused by a coordinator failure and no *writeSLS* operation is executed is in the 10^{-4} domain for the example MANET scenario and, therefore, negligible. However, this probability might be different in other scenarios.

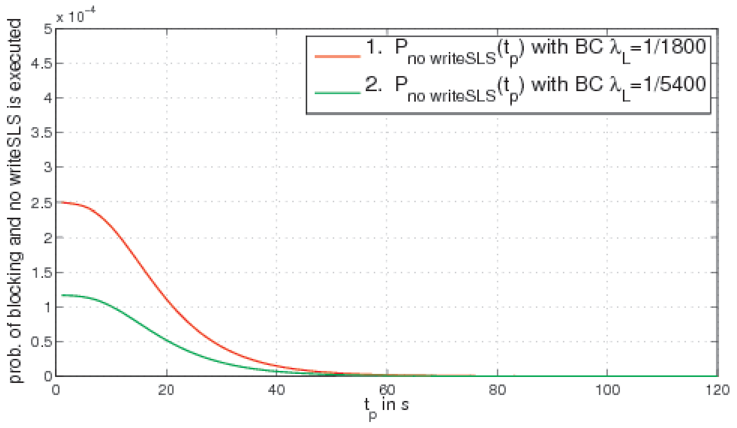


Figure 6.9: Probability that no *writeSLS* operation is executed with BC and time-out-based escalation in the example scenario (n=3, Δ_{vo}=1 s).

6.3.2.2 Semantic Transaction Model

In the semantic transaction model, the time-out trigger for the execution of the *writeSLS* operation by the BC has to be initialized at the beginning of the transaction at t_s, as participants enter uncertainty at a random time within $[t_s, t'_p]$. This requires precise knowledge of t'_p to prevent unnecessary transaction aborts, which is a major disadvantage of this approach. This problem can be approached by allowing a time-out extension scheme, for example as proposed in the TCOT protocol [83]. If the MC recognizes that the t'_p known by the BC does not hold, because some operations took longer to execute than expected, a new time-out is negotiated and the time-out is extended.

I will not consider such a time-out extension scheme here, but assume precise knowledge of t_u to be available. Figures 6.10 and 6.11 show the resulting protocol assumed in the following calculations.

I calculate the probability of PA to suffer extended uncertainty (ii) while no *writeSLS* operation is executed with the time-out-based escalation strategy. Calculations are structured by considering the relevant situations in interval $[t_s, t'_p]$ first and in $[t'_p, t_u]$ afterwards.

In the interval $[t_s, t'_p]$, PA experiences an extended uncertainty (ii) situation caused by a node failure of the MC if the MC suffers a node failure within $[t_s, t'_p]$, while PA has already entered uncertainty. This probability is given by $P'_u(t'_p)$ as developed in Section 4.6.2. If the BC suffers a node failure before the time-out for execution of the *writeSLS* operation fires and before PA contacts the BC at $t_u + 2\delta_m$, PA is uncertain and no *writeSLS* operation is executed. The probability for this event is given by $BF'1(t'_p)$.

$$BF'1(t'_p) = P'_u(t'_P) \cdot F_{N,BC}(t_u + \delta_m) \tag{6.21}$$

If the MC does not experience a node failure within the processing phase causing extended uncertainty, PA suffers extended uncertainty (ii) and no *writeSLS* operation is executed if the MC suffers a node failure within $[t'_p, t_u + \delta_m]$ and BC suffers a node failure in $[t_s, t'_p + \delta_m]$ and, therefore, will not answer the

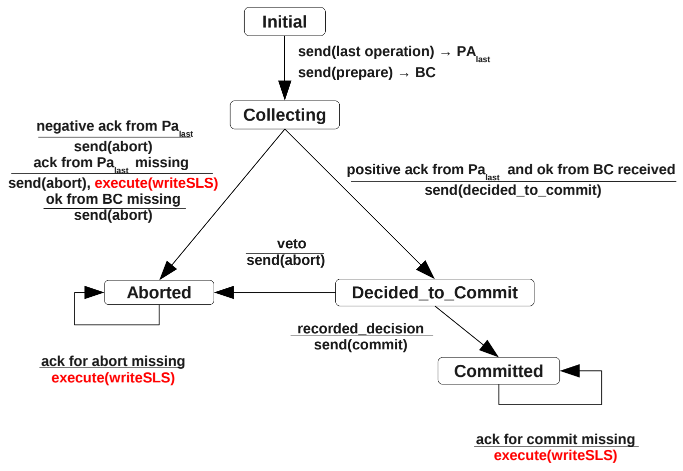


Figure 6.10: State diagram of MC with time-out-based escalation strategy.

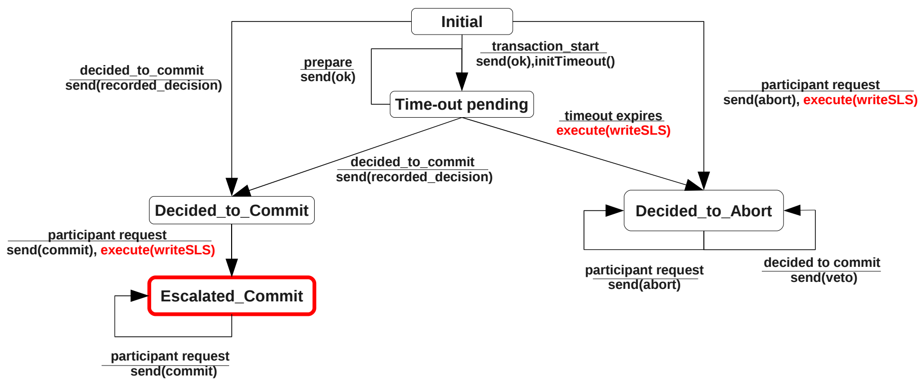


Figure 6.11: State diagram of BC with time-out-based escalation strategy.

reachability test. As both coordinators have failed, no *writeSLS* operation is executed. This probability is given by $BF'2(t'_p)$.

$$BF'2(t'_p) = \left[1-P_{o<f}(t'_p)\right]^{n-2} \cdot \left[1-F(t'_p)\right] \cdot F_{N,BC}(t'_p+\delta_m) \cdot F_{N,MC}(t'_p..t_u+\delta_m) \quad (6.22)$$

A similar situation as above can occur if the BC and MC do not suffer a node failure until t'_p but within $[t'_p, t_u]$. PA is uncertain and no *writeSLS* operation is executed at probability $BF'3(t'_p)$.

$$BF'3(t'_p) = \left[1-P_{o>f}(t'_p)\right]^{n-2} \cdot \left[1-F(t'_p)\right] \cdot F_{N,MC}(t'_p..t_u) \cdot F_{N,BC}(t'_p..t_u) \quad (6.23)$$

The event that the MC is alive but blocked and cannot execute the *writeSLS* operation occurs in two situations: (i) if the BC suffers a node failure after it has answered the reachability test and before it sends the *decided_to_commit* message (the probability of this event is given by $F_{N,BC}(t'_p + 2\delta_m..t'_p + 2\delta_m + \Delta_{ex})$); and (ii) if the BC is alive, but a communication failure between the MC and BC in the interval $[t'_p + 2\delta_m, t'_p + 4\delta_m + \Delta_{ex}]$ prevents the *recorded_decision* message to be delivered to the MC, which then blocks. $BF'4(t'_p)$ calculates the probability of situation (i) or situation (ii) to occur.

$$\begin{aligned} BF'4(t'_p) \quad = \quad & P_{U_{min}}(t'_p) \cdot \left[1 - F_N(t_p + \Delta U_{max})\right] \\ & \cdot\left[F_{N,BC}(t'_p + 2\delta_m..t'_p + 2\delta_m + \Delta_{ex})\right. \\ & + F_C(t'_p + 2\delta_m..t'_p + 4\delta_m + \Delta_{ex}) \cdot F_C(t'_p + 5\delta_m + \Delta_{ex}) \\ & - F_{N,BC}(t'_p + 2\delta_m..t'_p + 2\delta_m + \Delta_{ex}) \\ & \left.\cdot F_C(t'_p + 2\delta_m..t'_p + 4\delta_m + \Delta_{ex}) \cdot F_C(t'_p + 5\delta_m + \Delta_{ex})\right] \quad (6.24) \end{aligned}$$

The probability of PA to suffer extended uncertainty (ii) and no *writeSLS* operation is executed within the BC protocol is now given by the probability that either the MC suffers node failure within the processing phase given by $BF'1(t'_p)$ or that a relevant blocking situation is caused during the decision phase. This probability is now given by $P'_{no\,writeSLS}(t'_p)$.

$$\begin{aligned} P'_{no\,writeSLS}(t'_p) \quad = \quad & BF'1(t'_p) + \left[1 - F(t'_p + 2\delta_m + \Delta_{ex})\right] \\ & \cdot\left[BF'2(t'_p) + BF'3(t'_p) + BF'4(t'_p)\right] \quad (6.25) \end{aligned}$$

Predictions

Figure 6.12 depicts the risk of PA to suffer extended uncertainty and no *writeSLS* operation is executed for the example MANET scenario of this work.

As expected, this probability is higher than in the strict model, since with semantic transactions extended uncertainty situations occur more likely as described in Section 4.6.1 and, additionally, only early integration of the BC is possible. However, the risk that no *writeSLS* operations are executed, while PA is uncertain is still very low and does not leave the 10^{-3} domain in the example MANET scenario of this work. For example, at t'_p=20 s the probability that the decision log is not preserved at P_{ret} is 0.56 %. A more reliable BC, with an exponential cdf $F_L(t)$ with parameter λ_L=1/5400 reduces this risk only slightly, e.g. at t'_p=20 s this probability is reduced from 0.56 % to 0.53 % as shown

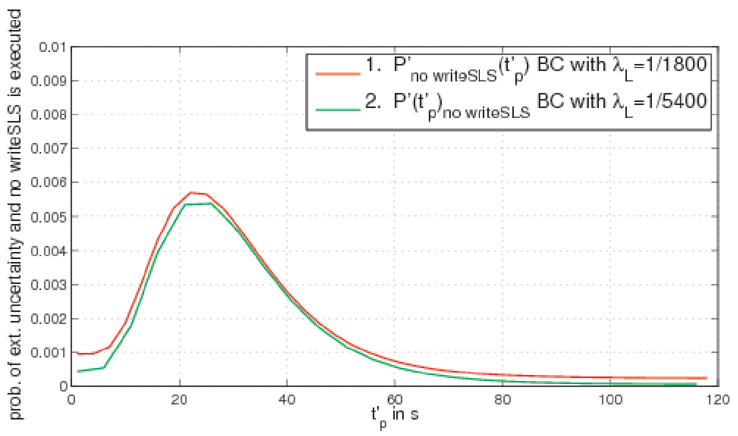


Figure 6.12: Probability of PA to suffer extended uncertainty and no *writeSLS* operation is executed with $n{=}3$ and $\Delta_{ex}{=}1\,$s.

in Figure 6.12. The maximum risk of PA to suffer extended uncertainty while no *writeSLS* operation is executed is observed at $t'_p{=}23\,$s with 0.58 %. Hence, from the perspective of PA the chance to not suffer from extended uncertainty or to suffer from extended uncertainty that can be recovered at P_{ret} by using the SLS is at least 99.42 % in the example scenario.

6.3.3 Summary - Escalation Strategies

In this section, I proposed the enhancement of the BC commit protocol with an escalation strategy, which assures that the decision log is preserved within the SLS at availability P_{ret} if the MC suffers a node failure. The two escalation strategies force the BC to execute a *writeSLS* operation either triggered by the request of an uncertain participant or by a time-out.

However, a small risk that no *writeSLS* operation is executed must remain. This situation occurs if both, the MC and BC suffer from a node failure simultaneously, or if the BC suffers a node failure while the MC is blocked. For the time-out based escalation strategy, I calculated these risks and showed that for the example MANET scenario of this work, the probabilities of these events are negligible in the strict transaction model and in the semantic transaction model. For strict transactions and *late integration* of the BC, I showed that the decision log can be provided at P_{ret} with a probability greater than 99.975 %. Risks are further reduced, if a special stable BC is available. Semantic transactions show a slightly lower probability of at least 99.42 % for the situation that PA is uncertain and cannot retrieve the decision log from the SLS at probability P_{ret} in the example scenario. However, these risks can be further reduced in the strict transaction model and in the semantic transaction model if cooperative recovery is used.

6.4 Summary and Conclusion

In this chapter, I analyzed the use of a backup coordinator (BC) to compensate for blocking (ii) and extended uncertainty (ii) situations. Additionally, I integrated the SLS into the BC protocol by proposing two so-called escalation strategies. These strategies trigger a *writeSLS* operation in case a failure is detected either due to a time-out or the request of an uncertain participant.

The main contribution of this chapter is a calculation model to derive the probability of blocking (ii) and extended uncertainty (ii) situations when a BC is integrated in commit processing. The calculation model also considers the time-out-based escalation strategy and therefore allows to calculate the risk of a participant to suffer a blocking (ii) or extended uncertainty (ii) situation that cannot be recovered by the SLS.

Together with the calculations presented in Chapters 4 and 5 it is now possible to calculate the total risk of a participant to suffer blocking or extended uncertainty caused by a participant or coordinator failure that cannot be recovered by the SLS. Hence, a comprehensive calculation model is given covering all blocking situations that can occur in a MANET.

A calculation model predicting the benefit of a BC is important as the use of a BC introduces the disadvantage of a new blocking situation. In this situation, the main coordinator (MC) is blocked due to a failure of the BC. This disadvantage can overcompensate the benefit of increased coordinator reliability, i.e. the integration of a BC can increase the blocking risks instead of reducing it.

I showed that the time when the BC is integrated in commit processing is critical for whether blocking risks are reduced or not. I distinguish two general schemes to integrate the BC in commit processing. In the early integration scheme, the BC is selected at t_s, while with the late integration scheme it is defined at t_p. In the strict transaction model, the late integration scheme is generally beneficial in the example scenario and results in a significant reduction of blocking risks. In contrast, an early integration of the BC is not necessarily helpful, since it causes higher blocking risks for certain transaction sizes instead of reduced risks. In the semantic model the risk for extended uncertainty is generally higher, as uncertainty periods of participants are larger. Here, a BC reduces risks for extended uncertainty only slightly for the example scenario.

It can be concluded that for strict transactions early selection of a BC is generally not feasible, while for semantic transactions such a general statement cannot be made.

In addition, the probabilistic model allows to consider individual failure probabilities of BCs, i.e. particular reliable BCs can be considered in calculations. In the strict model an especial reliable BC significantly reduces the probability of blocking (ii) that cannot be compensated by the SLS. For semantic transactions an especial reliable BC reduces such risks only slightly in the example scenario.

The presented calculation models assumed that a BC is discovered in 1–2 hop distance to the MC and all participants. While I did not cover the problem of discovering such a node here, I refer to [31] for a description on how discovery of a BC in 1–2 hop distance can be efficiently implemented in a MANET.

Chapter 7

Summary and Conclusion

In my thesis I presented several contributions in the area of distributed atomic transaction processing in MANETs focusing on the prediction and compensation of blocking situations.

The common approach to study the blocking problem is its analysis on an algorithmic level, e.g. by proving that a commit protocol is blocking in a defined system model or by deriving its message and time complexity etc. A protocol is considered to be susceptible to blocking even if blocking situations occur at negligible probabilities only. In contrast, this thesis analyzed the blocking problem from a probabilistic perspective. The presented probabilistic models allow to analytically derive quantitative statements about abort and blocking risks for basic transaction models. As a further step, recovery schemes to compensate for blocking have been developed and integrated into these models, which allows to quantify their benefit analytically. Being able to predict the blocking risks of a transaction and the benefit of compensation schemes is essential to evaluate whether the use of more reliable recovery schemes is indicated.

The three major contributions of this thesis are: (i) a comprehensive probabilistic model to analyze abort and blocking risks of atomic transactions in MANETs; (ii) the Shared Log Space (SLS) approach to compensate for blocking due to failures of participants; and (iii) a probabilistic model to calculate the benefit of a backup coordinator (BC) if blocking is caused by a node failure of the coordinator. In the following, conclusions of (i)–(iii) are summarized.

Prediction of abort and blocking probabilities is complex, since every transaction shows individual risks that are influenced by numerous parameters, such as number of participants, distribution of operations, and transaction size, just to name a few. These parameters have been varied to analyze the abort and blocking risks of different transaction scenarios. In addition, simulation results have been presented to support applicability of the calculation model. The major conclusions applying the model to an example MANET scenario are:

- Generally, the abort probability of strict transactions and semantic transactions is high. Only a small spectrum of transactions shows abort rates that are small enough to be tolerated. For such transactions, blocking has to be considered.

- In the strict model, low blocking probabilities are observed, since here blocking is a subsequent problem to abort. In fact for most strict transactions showing a tolerable abort rate, blocking risks are found to be negligible in the example scenario.

- Semantic transactions show considerably higher blocking risks than strict transactions. However, cooperative recovery can efficiently compensate for most blocking situations if multi-hop routing is available, reducing blocking risks to a negligible level. If multi-hop routing is not available, blocking occurs frequently.
- From the defined blocking situations, the situations caused by participant failures are much more likely than the situations caused by a node failure of the coordinator. In fact, the probability that a node failure of the coordinator causes blocking of a participant is negligible for the scenarios considered.

Hence, an important conclusion is that high blocking risks are not an inherent problem of transactions in MANETs. Furthermore, critical transactions are rare in the strict model and also in some semantic scenarios. Being able to identify these critical transactions analytically in a MANET scenario is crucial to manage blocking. Although numerous influences and relations make accurate prediction of abort and blocking risks a complex problem, I have shown that probabilistic models can be used to solve it. The probabilistic model proposed to analyze abort and blocking risks has been presented in Chapter 4. I have published parts of this model in [26, 25, 28].

As a next step, I proposed schemes to compensate for blocking in transactions. A major requirement was that such schemes allow to quantify their benefit analytically. The main intention for this requirement was that a coordinator predicting high blocking risks for a transaction is able to decide whether integration of a recovery scheme meets a desired level of blocking risks.

In Chapter 4, I distinguished between blocking situations caused by participant and coordinator failures. For both situations, a compensation scheme was proposed that allows to predict its benefit. Regarding participant failures, the major contribution was the Shared Log Space (SLS), while the backup coordinator (BC) scheme was proposed to reduce the risk of blocking caused by a node failure of the coordinator.

The SLS abstraction and its implementations described in Chapter 5 are considered as central contributions of this work. The idea behind the SLS is to establish a distributed shared storage that preserves decision logs at a defined availability for blocked participants. By controlling the availability of the decision log for recovering participants, their probability to leave blocking is increased to a desired level.

I presented two implementation approaches of the SLS to demonstrate its applicability in MANETs: (i) a lightweight approach that disseminates the decision log once and does not maintain its availability any further, and (ii) an approach based on a cluster-overlay, where the availability of the decision log is constantly maintained. The main focus of this work was on the lightweight approach and its underlying calculation model. This model considered MANET scenarios with and without multi-hop routing, individual node failure probabilities, and recovery from node failures to predict the availability of a decision log for recovering participants. From the findings presented in Chapter 5 the important conclusions are:

- The general concept of the SLS is applicable in MANETs. Not only its proposed lightweight implementation is straightforward to implement, it also has been verified that it allows to reduce blocking risks in a controllable manner.
- The log availability model used in the lightweight approach allows for an effective implementation of the SLS by considering individual node failure probabilities and recovery from node failures.
- Embedding the SLS in recovery protocols of strict and semantic transactions allows to provide a probabilistic guarantee for successful recovery of blocked transaction participants.

I consider the general abstraction of the SLS, independently of its implementation, to be a useful contribution as it allows to reason about the reduction of blocking risks in a probabilistic manner. While in this work the SLS was only considered in the context of atomic transaction processing, it is useful in numerous other settings as shown in [30, 65].

To compensate for the blocking situation caused by a node failure of the coordinator, I proposed to use a second coordinator acting as a backup coordinator (BC). While such schemes have been proposed by other scholars for fixed networks, my main contribution is a calculation model to estimate the benefit of a BC and to integrate the SLS into the BC protocol. Investigation of the BC protocol led to the following results:

- In the strict model, a BC compensates for blocking efficiently if chosen right before the commit protocol is initiated. An integration of the BC at transaction start may lead to increased blocking risks.
- In the semantic transaction model, where a BC has to be integrated at transaction start, only a small reduction in blocking risks can be observed in the example scenario of this work.
- Integration of the SLS into the BC protocol allows to reduce situations where blocking is caused by a node failure of the coordinator and cannot be compensated by the SLS to a negligible level.

The result that blocking is possibly increased by the use of a BC underlines the importance of a prediction model as proposed. I consider the calculation model as generic enough to also be applied to other dynamic environments such as peer-to-peer systems in fixed networks. The calculation model analyzing reduction in blocking risks due to a BC in strict transactions and semantic transactions has been published in [27].

This work analyzed blocking risks of distributed atomic transactions in MANETs as well as methods to compensate for these risks from a probabilistic perspective. I consider the models and methods developed as an important contribution towards a better understanding of atomic transaction processing in MANETs. Furthermore, I believe that the contributions of this work provide a basis for promising future research projects in adaptive risk management for MANET transaction processing.

Appendix A

Symbols and Abbreviations

List of Symbols

$\mathcal{A}$	$\mathcal{A}$ denotes a MANET formed within a certain area, while $\mathcal{A}$ represents one vertices within a graph defined by the Area Graph–Based Mobility Model (AGB).
b	Considering a constant energy consumption, b describes the maximum assumed battery life of a mobile node.
Δ_{ex}	Time the coordinator will wait until the last operation of PA_{last} is executed in the semantic transaction model.
Δ_{vo}	Time the coordinator will waits for vote messages in the 2PC protocol.
Δ_k	Time since plan k was derived; also called *age* of k.
$\Delta_{k,thr}$	Allowed age of k before a k has to be revised.
ΔU	Duration of the uncertainty window.
ΔU_{min}	Minimum duration of the uncertainty window in the 2PC protocol (failure-free case).
ΔU_{max}	Maximum duration of the uncertainty window in the 2PC protocol (failure case)
Δ_w	Time the writeSLS operation is allowed to execute.
Δ_r	Time the readSLS operation is allowed to execute until a decision log is returned at probability P_{ret}.
δ_m	Assumed message delay in $\mathcal{A}$.
δ_{to}	Time a node will wait for a protocol message in commit processing, e.g. in 3PC, Paxos Commit, etc.
$f_E(t)$,$F_E(t)$	Pdf and cdf of energy-related node failures.

$f_L(t)$,$F_L(t)$	Pdf and cdf of node failures caused by mobility between vertices of the AGB model.
$f_J(t)$,$F_J(t)$	Pdf and cdf of the time nodes remain disconnected from $\mathcal{A}$ due to mobility of nodes between the AGB clusters.
$f_C(t)$,$F_C(t)$	Pdf and cdf of communication failures.
$f_{CR}(t)$,$F_{CR}(t)$	Pdf and cdf of the time a failed communication path remains dysfunctional.
$f_T(t)$,$F_T(t)$	Pdf and cdf of technical failures causing a node failure.
$f_N(t)$,$F_N(t)$	Pdf and cdf of node failures.
$f(t)$,$F(t)$	Pdf and cdf of the general failure, i.e. the probability that either a node or a communication failure occurs.
k	The defined or undefined set of assistant nodes the decision log is disseminated to if a writeSLS operation is executed. Also called *dissemination plan.*
k_r	Realized dissemination plan, i.e. the assistant nodes that actually received and stored a decision log.
$\#k$	Number of assistant nodes defined in k.
$LA(k, t_m)$	Availability of the decision log for recovering participants provided by plan k until t_m if common node failures are assumed.
$LA_{in}(k, t_m)$	Availability of the decision log for recovering participants provided by plan k until t_m if individual node failure probabilities are considered.
$LA^{'}(k, t_m)$	Availability of the decision log for recovering participants given by plan k until t_m if common node failures and recovery from node failures are considered.
λ_T	Parameter of the exponential distribution describing node failures caused by technical failures. (Parameter of $f_T(t)$)
λ_L	Parameter of the exponential distribution describing the sojourn time of nodes in an area $\mathcal{A}$. (Parameter of $f_L(t)$).
λ_E	If the probability of energy-related node failures is assumed to be exponentially distributed, the parameter of $f_E(t)$ is denoted by λ_E.
λ_T	Parameter of the exponential distribution describing the probability of technical failures. (λ_T is a parameter of $f_T(t)$).
$n_{\mathcal{A}}$	Average number of nodes connected to $\mathcal{A}$.
n	Number of participants in a transaction.

$o(t)$,$O(t)$	Pdf and cdf of the event that the last operation of a participant's transaction branch is received at (until) time t.
$P_{ret}(t)$	Probability of successfully retrieving a decision log at time t within time Δ_r.
P_{path}	Probability that a communication path between two randomly chosen nodes in $\mathcal{A}$ exists. Also called path probability.
PA	The participant from which perspective blocking risks are calculated.
PA_{other}	The other participants of a transaction not including PA.
PA_{last}	The participant that receives the last operation of the transaction.
$P_{o<f}(t)$	Probability of an unrecognized failure, i.e. a participant suffers a failure after acknowledging the last operation of its transaction branch.
$P_{o>f}(t)$	Probability of a recognized failure, i.e. the failure of a participant is recognized because it happened before the last operation is acknowledged.
$P_{o<f_C}(t)$	Probability of an unrecognized communication failure.
$P_{o<f_N}(t)$	Probability of an unrecognized node failure.
$P_{o<f_C,r}(t)$	Probability of an unrecognized communication failure that recovers until t.
$P_{o<f_C,nr}(t)$	Probability of an unrecognized communication failure that does not recover until t.
$P_{a_p}(t_p)$	Probability of transaction abort during the processing phase of a strict transaction of size t_p.
$P_{a_d}(t_p)$	Probability of transaction abort during the decision phase of a strict transaction of size t_p.
$P_a(t_p)$,$P'_a(t_p)$	Overall probability of transaction abort in the strict model. $P'_a(t_p)$ additionally considers recovery from communication failures.
$P^*_a(t'_p)$	Overall probability of transaction abort in the semantic transaction model.
$P_u(t_p)$,$P'_u(t_p)$	Probability of blocking situation (i) in the strict model. $P'_u(t_p)$ also considers recovery from communication failures.
$P_{c,nr}(t)$	Probability of the event that a communication path between PA and one node of PA_{other} fails and does not recover until t.
$P'_{u,cr}(t_p)$	Probability of blocking situation (i) if cooperative-recovery and recovery of communication paths are considered.

$P_{u,cr}(t_p)$	Probability of blocking situation (i) if cooperative-recovery is considered.
$P_{u,cr}^{pp}(t_p)$	Probability of blocking situation (i) if cooperative-recovery is considered and P_{path} is used to calculate the probability of communication paths required for cooperative recovery.
$P_u^*(t_p)$,$P_u^{'*}(t_p)$	Probability of extended uncertainty situation (i) in the semantic transaction model. $P_u^{'*}(t_p)$ additionally considers recovery of failed communication paths.
$P_{u,cr}^{*'}(t'_p)$	Probability of extended uncertainty situation (i) if cooperative recovery and recovery of communication paths is considered.
$P_{u,cr}^*(t'_p)$	Probability of extended uncertainty situation (i) if cooperative recovery is considered.
$P_{u,cr}^{*pp}(t'_p)$	Probability of extended uncertainty situation (i) if cooperative recovery is considered and P_{path} is used to calculate the availability of communication paths for cooperative recovery.
$P_{u,SLS}(t_p)$	Probability of blocking or extended uncertainty if the SLS is used
$P'_{u,SLS}(t_p)$	Probability of blocking or extended uncertainty if the SLS is used and recovery of communication paths is considered.
$P_v(t_p)$,$P_{v,cr}(t_p)$	Probability of blocking situation (ii) (blocking caused by a node failure of the coordinator in the strict model). $P_{v,cr}(t_p)$ additionally considers cooperative recovery.
$P_v^*(t'_p)$	Probability of extended uncertainty situation (ii) (extended uncertainty caused by a node failure of the coordinator in the semantic model).
$P_{v,cr}^*(t'_P)$	Probability of extended uncertainty situation (ii) if cooperative recovery is assumed.
$P_{v,BCe}(t_p)$	Probability of blocking situation (ii) with *early-integration* of a BC.
$P_{v,BCl}(t_p)$	Probability of blocking situation (ii) with *late-integration* of a BC.
$P_{v,BC}^*(t'_p)$	Probability of extended uncertainty situation (ii) with a BC.
$P_{pr}(l,t)$	Probability of the event that l assistant nodes are present in $\mathcal{A}$ at time t when common node failure probabilities are assumed.
$P_{pr,in}(l,t)$	Probability of the event that l assistant nodes are present in $\mathcal{A}$ at time t if individual node failure probabilities are assumed.
$P'_{pr}(l,t)$	Probability of the event that l assistant nodes are present in $\mathcal{A}$ at time t if common node failure probabilities are assumed, as well as recovery from node failures.

σ	Parameter σ of log-normal distributed probabilities, e.g. $f_C(t)$.
t_f	Point in time a general failure occurs. t_f is interpreted as a random variable.
t_{rec}	Point in time the recovery protocol of a participant or coordinator is started.
t_m	Mission time; describes the duration until the availability of the decision log should be provided at P_{ret}.
t_p	Duration of the processing phase of a strict transaction.
t'_p	Duration of the processing phase of a semantic transaction.
t_s	Point in time a transaction is initiated and started.
t_o,$t_{i,o}$	Point in time the last operation of a local transaction branch is issued to participant i.
t_{cr}	Point in time cooperative recovery is started.
t_{tr}	Point in time the register operation terminates.
$t_{ter,p}$	Point in time the termination protocol of a participant is started.
$t_{ter,c}$	Point in time the termination protocol of the coordinator is started.
t_{res}	Point in time the restart protocol of uncertain participants is started.
μ	Parameter μ of log-normal distributed probabilities, e.g. $f_C(t)$.

Abbreviations

2PC	Two-Phase Commit Protocol.
AC	Atomic Commit.
ACID	Atomicity, Consistency, Isolation, Durability.
ACP	Atomic Commit Protocol.
AGB	Area Graph–Based Mobility Model (AGB).
AODV	Ad-hoc On-Demand Vector Routing.
ATM	Advanced Transaction Model.
BC	Backup Coordinator.
cdf	Cumulative Distribution Function.
DAC	Dictatorial Atomic Commit.
DSDV	Destination Sequence Distance Vector Routing.
DSR	Dynamic Source Routing.
EC-SAC	Eventually Certain Semantic Atomic Commit.
EC-WSAC	Eventually Certain Weak Semantic Atomic Commit.
MC	Main Coordinator.
NB-AC	Non-blocking Atomic Commit
NB-WAC	Non-blocking Weak Atomic Commit
pdf	Probability Density Function.
RWP	Random Way-Point Mobility Model.
RM	Resource Manager.
SAC	Semantic Atomic Commit.

Appendix B

Simulation Tools

The main simulation tool used in this work is the *ns2* network simulator. Another evaluation environment developed within the CoCoDa project is the MarNET emulator, that was used for evaluation of the cluster-based SLS implementation presented in Section 5.8. While both tools are completely independent, the simulation and emulation processes share some tools developed within this work. This Appendix gives a brief overview of the simulation and emulation tools as well as the simulation and emulation process.

B.1 The *NS2* Simulation Process

ns2 [2] is a discrete event simulator for network research. In MANET research, *ns2* is the by far most frequently used simulation tool [86]. The simulator implements a large number of protocols such as TCP, UDP, and various multi-hop routing schemes like AODV, DSDV, DSR, as well as tools to monitor and to analyze MANET simulations such as the Network Animator NAM[1].

Since *ns2* is written in *C++*, new functionality and protocols have to be implemented in *C++* and are accessed from *OTcl* scripts through a *C++*/OTcl linkage.

For simulation, *OTcl* scripts are used to initiate the event scheduler of *ns2*, to configure network components, and to define events such as initiation of a transaction between a defined set of nodes. As thousands of transactions have to be simulated with varying parameters, *OTcl* scripts have to be generated automatically. In the following, the toolchain depicted in Figure B.1 used in this work to generate *OTcl scripts* is briefly presented.

In the first step, a MANET scenario is generated which describes the movement for every node over the complete simulation time. Two different tools have been used: (i) the *setdest* tool from the *ns2* suite implementing the RWP mobility model and (ii) the AGB Model generator developed in [145] to create AGB Mobility scenarios. However, there are numerous other MANET scenario generators available that are compatible with the *ns2* simulator such as Bonn-

[1]Network Animator for ns2: Nam
http://www.isi.edu/nsnam/nam/.

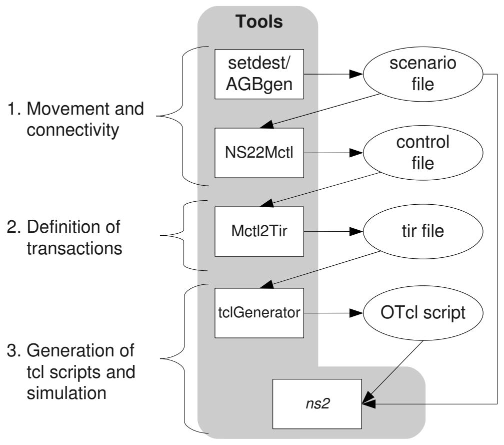

Tools
setdest/ AGBgen
scenario file
1. Movement and connectivity
NS22Mctl
control file
2. Definition of transactions
Mctl2Tir
tir file
tclGenerator
OTcl script
3. Generation of tcl scripts and simulation
ns2

Figure B.1: *ns2* simulation process.

Motion[2] or the Important Mobility generator[3] that can be used instead of (i) and (ii) to test other mobility models, while the remaining toolchain remains the same. The result of this step is a scenario file which is directly interpreted by the *ns2* simulator (see Figure B.1).

To define transactions processed in the simulation, an intermediate step is required. The scenario file is analyzed by a tool called *NS22Mctl*[4] which computes the connectivity among all nodes. The *NS22Mctl* tool creates a so-called *control file* that lists the loss rate for all node pairs for every point in time.

In the second step, the transactions to be simulated are defined using the *Mctl2Tir*[5] tool. The *Mctl2Tir* tool reads the control file and identifies groups of mobile nodes where one node (used as coordinator) has communication paths to all other nodes of the group not exceeding a given loss rate or hop distance. Hence, this tool defines the the transactions to be simulated. Besides the link quality between coordinator and participants, the frequency of transaction initiations can be customized to control the transaction load in the MANET. The result of the second step is a so-called *tir file* which lists all the transactions to be processed within the simulation.

In the third step, the *tir* file is used to generate the *OTcl* script. The transactions defined within the *tir* file are interpreted and the necessary events (*OTcl* commands) to initiate a transaction at the defined time are created. The resulting *OTcl* script is then interpreted together with the scenario file by the *ns2* simulator.

[2]BonnMotion: A mobility scenario generation and analysis tool. http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/.

[3]IMPORTANT: An evaluation framework to study the "Impact of Mobility Patterns On Routing in Ad-hoc Networks". http://nile.usc.edu/important/software.htm.

[4]The source code as well as documentation of the NS22Mctl tool can be found at http://www.cocoda.de/emulation/multiproject/ns22mctl/index.html.

[5]The source code as well as documentation of the Mctl2Tir tool is available at http://www.cocoda.de/emulation/multiproject/mctl2tir/index.html.

Listing B.1: Radio and protocol settings of mobile nodes for the example scenario.

```
set val(chan)          Channel/WirelessChannel   ;# channel type
set val(prop)          Propagation/TwoRayGround  ;# radio-propagation model
set val(netif)         Phy/WirelessPhy           ;# interface type
set val(mac)           Mac/802_11                ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)            LL                        ;# link layer type
set val(ant)           Antenna/OmniAntenna       ;# antenna model
set val(ifqlen)        50                        ;# max packet in ifq
set val(nn)            15                        ;# number of mobilenodes
set val(rp)            AODV                      ;# routing protocol
set val(lenx)          500.0                     ;# heigth of sim-area
set val(leny)          500.0                     ;# length of sim-area
set val(stop)          500000.0                  ;# cancellation time

set p_rx 0.3
set p_tx 0.4

set initialenergy 9999999

Antenna/OmniAntenna set Gt_ 1.5
Antenna/OmniAntenna set Gr_ 1.5

# wireless range (calc. RXThresh with indep-utils/propagation/thresh.cc)
Phy/WirelessPhy set RXThresh_ 1.42681e-08
Phy/WirelessPhy set freq_ 9.14e+08
Phy/WirelessPhy set Pt_ 0.281838
```

B.1.1 *NS2* Simulation Parameter

In the following, the radio and protocol settings of *ns2* simulations presented in Chapter 4 and Chapter 5 are listed.

B.1.1.1 *NS2* Settings of the Example Scenario

Most simulations in Chapters 4 and 5 have been done for the consistent example scenario introduced in Section 4.1. In these simulations, I used the settings listed in Listing B.1 for the MANET protocol stack and radio characteristics of mobile nodes. Note that in the listing AODV is used as a routing protocol. If no routing was simulated, “AODV” is replaced with “DumbAgent”.

B.1.1.2 *NS2* Settings of Case Study

For simulations of the Case Study presented in Section 4.7, different radio characteristics where used. In these simulations, radio and antenna characteristics of the *914 MHz Lucent WaveLAN DSSS* card have been used. Additionally, the number of nodes is significantly higher at 40 nodes and the dimensions of the MANET area are 2000 m * 2000 m.

B.2 MarNET Emulator

As an alternative to the *ns2* simulator, an emulation environment for MANETs was developed within the CoCoDa project [1] and used for evaluation of the overlay-based SLS implementation. While simulation accelerates the simulated

Listing B.2: Radio and protocol settings of mobile nodes in the case study of Chapter 4.

```
set val(chan)          Channel/WirelessChannel  ;# channel type
set val(prop)          Propagation/TwoRayGround ;# radio-propagation model
set val(netif)         Phy/WirelessPhy          ;# network interface type
set val(mac)           Mac/802_11               ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue  ;# interface queue type
set val(ll)            LL                       ;# link layer type
set val(ant)           Antenna/OmniAntenna      ;# antenna model
set val(ifqlen)        50                       ;# max packet in ifq
set val(nn)            40                       ;# number of mobilenodes
set val(rp)            AODV                     ;# routing protocol
set val(lenx)          2000.0                   ;# heigth of sim-area
set val(leny)          2000.0                   ;# length of sim-area
set val(stop)          500000.0                 ;# finishing time
set p_rx 0.3
set p_tx 0.4
set initialenergy 9999999
set energymodel "EnergyModel"

# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

# 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.28183815
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
```

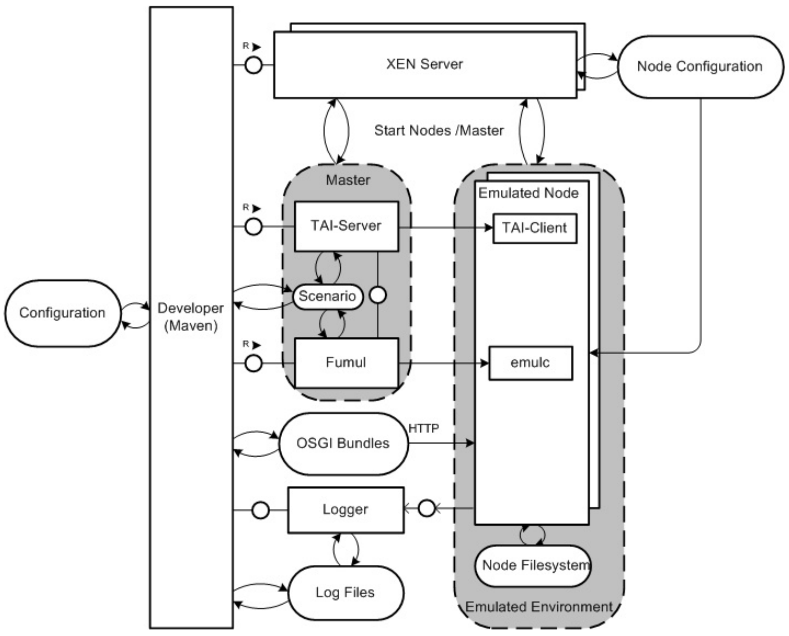


Figure B.2: CoCoDa Emulation Environment.

duration, emulation processes the behavior under observation in real-time, i.e. emulation of transaction processed within a time frame of 500000 s also requires an emulation time of 500000 s. The main advantage of emulation over simulation is that schemes to be evaluated do not have to be programmed against a special class hierarchy of a simulation suite, but the approaches to be evaluated can be directly implemented for the target environment, i.e. the operating system the software should later be deployed to. Hence, the development overhead with emulation is lower and convenient software testing is also possible.

The emulation system was developed in the CoCoDa project and is based on some components from the MarNET emulator [12, 98]. For a more detailed description of the emulation system, see [1].

The basic idea of the MarNET emulation system is to use paravirtualization to provide a complete independent virtual machine for each mobile node. Each virtual machine provides a complete Linux system (this work used Tiny Linux) which runs some additional components for emulation purposes.

In Figure B.2[6], the main components of the emulation system and their relations are depicted using a block diagram in FMC notation. The main components of the emulator are: (i) development environment (Developer), (ii) emulation server (Xen Server), (iii) control node (Master), and (iv) emulated nodes (Emulated Node). In the following, the different components are described briefly.

[6]This figure is taken from
http://www.cocoda.de/emulation/multiproject/xen/index.html

B.2.1 Emulation Server (Xen) and Emulated Nodes

For emulation servers Xen Linux Workstations were used, while each emulation server hosts a certain number of emulated nodes (Xen domU systems). The number of emulation nodes is mainly limited by the emulation server's main memory resources. Each emulated node requires 100 MB RAM. Within each emulated node, a small daemon (*emulc*) listens to time messages from a controlling master node. The master node is either the developer's machine or any dedicated virtual node.

At receipt of such a time stamp, directly reachable nodes are identified by *emulc* and a virtual network device is configured accordingly using the Net-Shaper [99] kernel module. Packets sent to unreachable destinations are then dropped by the network device. This device is used only for communication among mobile nodes. *Emulc* identifies reachable nodes by reading the *control file* of the MANET scenario (see Section B.1). Multi-hop routing on emulated nodes is provided by the *marnetd* daemon described in [12].

On the controlling master node, a component called *Fumul*[7] is running that synchronizes the emulation process by sending time events to all mobile nodes participating in the emulation. These messages are sent over a separate network device that is connected to a virtual private network, guaranteeing message delivery.

To control applications, e.g. to initiate transaction processing, a component called TAI-Client is running on each mobile node. The TAI-Client listens to messages from the TAI-Server that interprets a *tir* file (see Section B.1) and initiates transactions if a corresponding message is received. The TAI-Client is implemented as an OSGi bundle and provides an application initiation interface. This interface is used by most components of the CoCoDa project.

The OSGi framework allows for convenient deployment of the software under investigation to emulation nodes though the OSGi http admin service. The OSGi logging service and the log4j framework is used to send logs from emulated nodes to central log appender, e.g. a log file.

B.2.2 Emulation Process

A MANET scenario is described by a mobility definition given by a *scenario* file and by a *control file* giving the hop distance or loss rate for all node pairs at every time step of the simulation, and (iii) by a *tir* (transaction initiator resource) file. The *tir* and *control* files are derived as described in Section B.1.

The relation of the different components is presented in Figure B.3[8], which shows the interaction of the *Fumul*, *emulc*, *TAI-Server,* and the *TAI-Client* as a petri net in FMC notation. The *Fumul* server sends control messages used by *emulc* to configure the network device of the node using NetShaper. Before the emulation is started, a message containing the name of the *control* file used in this emulation is sent to each *emulc*.

The TAI-Server also receives the time messages from the Fumul server and checks whether a transaction has to be initiated. If an entry for the actual time is found in the *tir* file, a message is sent to the nodes listed in this *tir* entry.

[7]Further information about the Fumul can be found at
`http://www.cocoda.de/emulation/multiproject/fumul/index.html`.

[8]This figure is taken from [102].

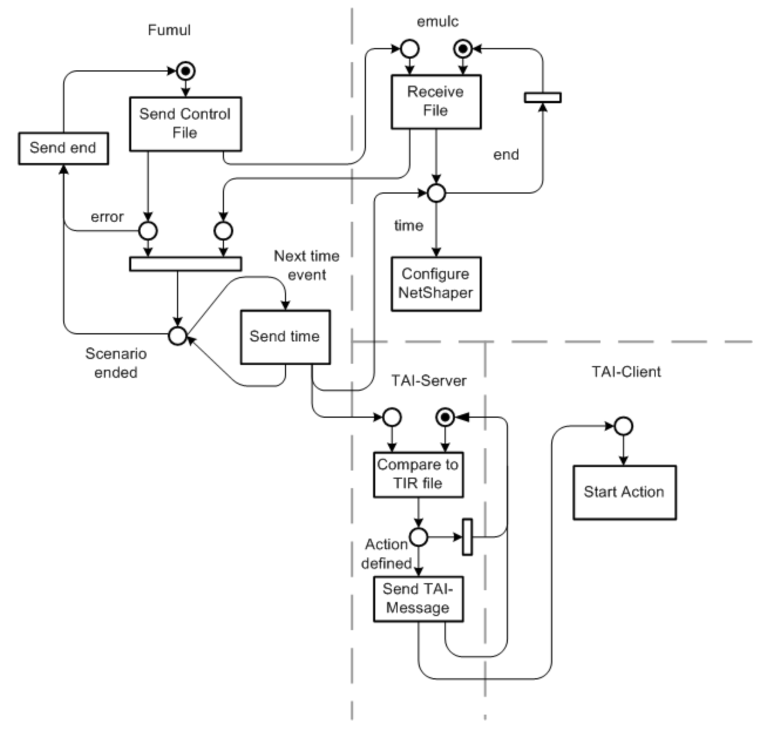


Figure B.3: Emulation Process.

Bibliography

[1] Cooperation communication data project. http://www.cocoda.de.

[2] Network simulator: Ns2. http://www.isi.edu/nsnam/ns.

[3] Osgi service platform release 3. http://www.osgi.org/Specifications.

[4] Maha Abdallah, Rachid Guerraoui, and Philippe Pucheral. One-phase commit: Does it make sense? In *Proceedings of International Conference on Parallel and Distributed Systems*, pages 182–192, 1998.

[5] Maha Abdallah, Rachid Guerraoui, and Philippe Pucheral. Dictatorial transaction processing: Atomic commitment without veto right. *Journal on Distributed and Parallel Databases*, 11(3):239–268, 2002.

[6] Y. Al-Houmaily and P. Chrysanthis. The implicit-yes vote commit protocol with delegation of commitment. In *Proc. of 9th Intl. Conf. on Parallel and Distributed Computing Systems*, 1996.

[7] Yousef J. Al-Houmaily and Panos K. Chrysanthis. Atomicity with incompatible presumptions. In *PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 306–315, New York, NY, USA, 1999. ACM Press.

[8] Todd R. Andel and Alec Yasinac. On the credibility of manet simulations. *IEEE Computer*, 39(7):48–54, 2006.

[9] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, pages 7–17, New York, NY, USA, 1997. ACM Press.

[10] F. Bai, N. Sadagopan, and A. Helmy. Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2003.

[11] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GloMoSim: A Scalable Network Simulation Environment. *UCLA Computer Science Department Technical Report*, 990027, 1999.

[12] O. Battenfeld, M. Smith, P. Reinhardt, T. Friese, and B. Freisleben. A Modular Architecture for Hot Swappable Mobile Ad hoc Routing Algorithms. *Embedded Software and Systems, 2005. Second International Conference on*, pages 359–366, 2005.

[13] Philip A. Bernstein, Vassco Hadzilacos, and Nathan Goodman. *Concurrency control and recovery in database systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[14] Philip A. Bernstein and Eric Newcomer. *Principles of Transaction Processing.* Morgan Kaufmann, 1997.

[15] Christian Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(3):55–66, 2001.

[16] Christian Bettstetter. On the connectivity of wireless multihop networks with homogeneous and inhomogeneous range assignment. In *Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, Vancouver, Canada, 2002.

[17] Christian Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 80–91, New York, NY, USA, 2002. ACM Press.

[18] Christian Bettstetter. *Mobility Modeling, Connectivity, and adaptive Clustering in Ad-hoc Networks.* Utz Verlag, 2004.

[19] Sven Bittner, Wolf-Ulrich Raffel, and Manuel Scholz. The area graph-based mobility model and its impact on data dissemination. In *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 268–272, Washington, DC, USA, 2005. IEEE Computer Society.

[20] C. Bobineau, P. Pucheral, and M. Abdallah. A unilateral commit protocol for mobile and disconnected computing. In *Int. Conf. On Parallel and Distributed Computing Systems (PDCS)*, 2000.

[21] Fatemeh Borran, Ravi Prakash, and Andre Schiper. Consensus problem in wireless ad hoc networks: Addressing the right issues. Technical report, Ecole Polytechnique Federale de Lausanne (EPFL), 2007.

[22] Y. Breitbart, A. Deacon, H.-J. Schek, A. Sheth, and G. Weikum. Merging application-centric and data-centric approaches to support transaction-oriented multi-system workflows. *ACM SIGMOD Record*, 22(3):23–30, 1993.

[23] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhn, Ya Xu, and Haobo Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, 2000.

[24] Jürgen Broß. An adaptive transaction manager for commit processing in mobile ad-hoc networks. Master's thesis, Freie Universität Berlin, May 2006.

[25] Joos-Hendrik Böse. Abort and blocking risk of atomic transactions in mobile ad-hoc networks. Technical Report B-08-07, Freie Universität Berlin, ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-08-07.pdf, June 2008.

[26] Joos-Hendrik Böse and Jürgen Broß. Predicting the blocking risk of atomic transactions in manets induced by coordinator failures. In *WAC'08 International Conference Wireless Applications and Computing*, 2008.

[27] Joos-Hendrik Böse and Jürgen Broß. Using a backup coordinator to compensate for blocking of atomic transactions in manets. In *International Workshop on Ad-hoc Ambient Computing (AdhocAmC)*. HAL - CCSD, 2008.

[28] Joos-Hendrik Böse, Jürgen Broß, and Heinz Schweppe. A probabilistic model for blocking risks of atomic transactions in p2p networks. In *International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P), In conjunction with the conference on very large database systeme.*, Auckland, 2008.

[29] Joos-Hendrik Böse, Stefan Böttcher, Le Gruenwald, Sebastian Obermeier, Heinz Schweppe, and Thorsten Steenweg. An integrated commit protocol for mobile network databases. In *IDEAS '05: Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS'05)*, pages 244–250, Washington, DC, USA, 2005. IEEE Computer Society.

[30] Joos-Hendrik Böse, Katharina Hahn, Lars-Christian Pelz, and Manuel Scholz. Optimistic fair transaction processing in mobile ad-hoc networks. Technical Report B-05-22, Freie Universität Berlin, 2005.

[31] Joos-Hendrik Böse and Andreas Thaler. Reliability evaluation of group service providers in mobile ad-hoc networks. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4278 of *LNCS*, pages 1540–1550. Springer, 2006.

[32] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[33] Julien Cartigny and David Simplot. Border node retransmission based probabilistic broadcast protocols in ad-hoc networks. *Telecommunication Systems*, 22(1-4):189–204, 2003.

[34] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: an engineering perspective. In *PODC '07: Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 398–407, New York, NY, USA, 2007. ACM.

[35] Tushar Deepak Chandra, Vassos Hadzilacos, and Sam Toueg. The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685–722, 1996.

[36] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.

[37] Panayiotis K. Chrysanthis and Krithi Ramamritham. Acta: A framework for specifying and reasoning about transaction structure and behavior. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, 1990.

[38] PK Chrysanthis. Transaction processing in mobile computing environment. *Advances in Parallel and Distributed Systems, 1993., Proceedings of the IEEE Workshop on*, pages 77–82, 1993.

[39] Brian A. Coan and Jennifer Lundelius Welch. Transaction commit in a realistic timing model. *Distributed Computing*, 4(2):87–103, June 1990.

[40] X/Open Consortium. Distributed transaction processing: The tx (transaction demarcation) specification. http://www.opengroup.org/, 1995.

[41] Eric C. Cooper. Analysis of distributed commit protocols. In *SIGMOD '82: Proceedings of the 1982 ACM SIGMOD international conference on Management of data*, pages 175–183, New York, NY, USA, 1982. ACM Press.

[42] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Sytems Concepts and Design*. Addison-Wesley, 3 edition, 2001.

[43] Ravi A. Dirckze and Le Gruenwald. A toggle transaction management technique for mobile multidatabases. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 371–377, New York, NY, USA, 1998. ACM.

[44] Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97, 1987.

[45] Margaret H. Dunham, Abdelsalam Helal, and Santosh Balakrishnan. A mobile transaction model that captures both the data and movement behavior. *Mobile Networks and Applications*, 2(2):149–162, 1997.

[46] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.

[47] A. Elmagarmid, Y. Leu, W. Litwin, and Marek Rusinkiewicz. A multidatabase transaction model for interbase. In *Proceedings of the sixteenth international conference on Very large databases*, pages 507–518, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[48] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[49] Hector Garcia-Molina. Using semantic knowledge for transaction processing in a distributed database. *ACM Transactions on Database Systems (TODS)*, 8(2):186–213, 1983.

[50] Hector Garcia-Molina and Kenneth Salem. Sagas. In *SIGMOD '87: Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, pages 249–259, New York, NY, USA, 1987. ACM Press.

[51] David Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[52] Dimitrios Georgakopoulos, Mark F. Hornick, and Amit P. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

[53] Jim Gray. Notes on data base operating systems. In *Operating Systems, An Advanced Course*, pages 393–481, London, UK, 1978. Springer-Verlag.

[54] Jim Gray. The transaction concept: Virtues and limitations. In *7th International Conference on Very Large Data Bases*, pages 144–154, Septemer 1981.

[55] Jim Gray. A comparison of the byzantine agreement problem and the transaction commit problem. In *Fault-tolerant distributed computing*, pages 10–17. Springer-Verlag, London, UK, 1987.

[56] Jim Gray and Leslie Lamport. Consensus on transaction commit. *ACM Transactions on Database Systems (TODS)*, 31(1):133–160, 2006.

[57] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Mateo, California, 1993.

[58] IEEE 802.11 Working Group. Ieee 802.11. http://www.ieee802.org/11, April 2008.

[59] IEEE 802.15 Working Group. Ieee 802.15. http://www.ieee802.org/15, April 2008.

[60] R. Guerraoui, M. Larrea, and A. Schiper. Non blocking atomic commitment with an unreliable failure detector. In *SRDS '95: Proceedings of the 14TH Symposium on Reliable Distributed Systems*, page 41, Washington, DC, USA, 1995. IEEE Computer Society.

[61] Rachid Guerraoui. Revisiting the relationship between non-blocking atomic commitment and consensus. In J.-M. Hélary and M. Raynal, editors, *Proceedings of the 9th International Workshop on Distributed Algorithms(WDAG95)*, volume 972, pages 87–100, Le Mont-Saint-Michel, France, 1995. Springer-Verlag.

[62] Zygmunt J. Haas and Marc R. Pearlman. The performance of query control schemes for the zone routing protocol. *IEEE/ACM Transactions on Networking (TON)*, 9(4):427–438, 2001.

[63] Vassos Hadzilacos. On the relationship between the atomic commitment and consensus problems. In *Proceedings of the Asilomar Workshop on Fault-Tolerant Distributed Computing*, pages 201–208, London, UK, 1990. Springer-Verlag.

[64] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys (CSUR)*, 15(4):287–317, 1983.

[65] K. Hahn, J. Böse, Birgitta König-Ries, and Philipp Obreiter. Robust and fair trading in volatile environments - overcoming technical problems and uncooperativeness. In *6th ACM international workshop on Data engineering for wireless and mobile access (MobiDE07) in Conjunction with 2007 ACM SIGMOD Intl. Conference on Management of Data (SIGMOD)*, pages 33–40, Bejing, China, June 2007. ACM.

[66] Jörg Hähner, Christian Becker, and Kurt Rothermel. A protocol for data dissemination in frequently partitioned mobile ad hoc networks. In Ahmed Tantawy and Kemal Inan, editors, *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC 2003)*, pages 633–640. Los Alamitos: IEEE Computer Society, Juni 2003.

[67] Yijie Han, Richard J. La, Armand M. Makowski, and Seungjoon Lee. Distribution of path durations in mobile ad-hoc networks: Palm's theorem to the rescue. *Computer Networks*, 50(12):1887–1900, 2006.

[68] Takahiro Hara and Sanjay K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 05(11):1515–1532, 2006.

[69] Takahiro Hara and Sanjay Kumar Madria. Dynamic data replication using aperiodic updates in mobile adhoc networks. *Lecture Notes in Computer Science*, 2973:869–881, 2004.

[70] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Mobile Computing and Networking*, pages 174–185, 1999.

[71] Dirk Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4):1067–1141, Dec 2001.

[72] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for ad hoc wireless networks. In *MSWiM '99: Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 53–60, New York, NY, USA, 1999. ACM Press.

[73] Mohammad Ilyas and Richard C. Dorf, editors. *The handbook of ad hoc wireless networks.* CRC Press, Inc., Boca Raton, FL, USA, 2003.

[74] D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[75] David B Johnson and David A Maltz. *Mobile Computing*, volume 353, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer Academic Publishers, 1996.

[76] Goutham Karumanchi, Srinivasan Muralidharan, and Ravi Prakash. Information dissemination in partitionable mobile ad hoc networks. In *SRDS '99: Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, page 4, Washington, DC, USA, 1999. IEEE Computer Society.

[77] Abdelmajid Khelil. *A Generalized Broadcasting Technique for Mobile Ad-Hoc Networks.* PhD thesis, Universität Stuttgart, 2007.

[78] Abdelmajid Khelil, Pedro José Marrón, Christian Becker, and Kurt Rothermel. Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. In *KiVS*, pages 142–153, 2005.

[79] Sandhya Khurana, Neelima Gupta, and Nagender Aneja. Reliable ad-hoc on-demand distance vector routing protocol. In *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*, volume 0, page 98, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[80] Michael Kifer and Arthur Bernstein. *Database Systems: An Application-Oriented Approach, Introductory Version.* Addison Wesley, 2004.

[81] Henry F. Korth, Eliezer Levy, and Abraham Silberschatz. A formal approach to recovery by compensating transactions. In Dennis McLeod, Ron Sacks-Davis, and Hans-Jörg Schek, editors, *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings*, pages 95–106. Morgan Kaufmann, 1990.

[82] P. Krishna Reddy and M. Kitsuregawa. Reducing the blocking in two-phase commit with backup sites. *Information Processing Letters*, 86(1):39–47, 2003.

[83] Vijay Kumar, Nitin Prabhu, Magaret H. Dunham, and Ayse Yasemin Seydim. Tcot-a timeout-based mobile transaction commitment protocol. *IEEE Trans. Comput.*, 51(10):1212–1218, 2002.

[84] Way Kuo and Zuo Ming. *Optimal Reliability Modeling: Principles and Applications.* Wiley, 2002.

[85] Way Kuo, V. Rajendra Prasad, Frank A. Tillman, and Ching-Lai Hwang. *Optimal Reliability Design, Fundamentals and applications.* Cambridge University Press, 2001.

[86] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. Manet simulation studies: the incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(4):50–61, 2005.

[87] Richard J. La and Yijie Han. Distribution of path durations in mobile ad hoc networks and path selection. *IEEE/ACM Transactions on Networking*, 15(5):993–1006, 2007.

[88] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, 16(2):133–169, 1998.

[89] Leslie Lamport. Paxos made simple. http://research.microsoft.com/users/lamport/pubs/paxos-simple.pdf, 2001.

[90] B. Lampson and D. Lomet. A new presumed commit optimization for two phase commit. In *Proceedings of the 19th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Dublin*. Morgan Kaufman pubs., 1993.

[91] Butler W. Lampson. How to build a highly available system using consensus. In *WDAG '96: Proceedings of the 10th International Workshop on Distributed Algorithms*, pages 1–17, London, UK, 1996. Springer-Verlag.

[92] W. H. O. Lau, M. Kumar, and Svetha Venkatesh. A cooperative cache architecture in support of caching multimedia objects in manets. In *WOWMOM '02: Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, pages 56–63, New York, NY, USA, 2002. ACM.

[93] Eliezer Levy, Henry F. Korth, and Abraham Silberschatz. An optimistic commit protocol for distributed transaction management. *ACM SIGMOD Record*, 20(2):88–97, 1991.

[94] Eliezer Levy, Henry F. Korth, and Abraham Silberschatz. A theory of relaxed atomicity. In *PODC '91: Proceedings of the tenth annual ACM symposium on Principles of distributed computing*, pages 95–110, New York, NY, USA, 1991. ACM Press.

[95] J. Liu, B. Li, Q. Zhang, and W. Zhu. Service Locating for Large-Scale Mobile Ad-Hoc Network. *International Journal of Wireless Information Networks*, 10(1):33–40, 2003.

[96] Yenliang Lu, Huier Lin, Yajuan Gu, and Helmy A. Towards mobility-rich analysis in ad hoc networks: using contraction, expansion and hybrid models. In *IEEE International Conference on Communications (ICC)*, volume 7, pages 4346– 4351, 2004.

[97] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[98] B. Freisleben M. Smith, S. Hanemann. Coupled simulation/emulation for cross-layer enabled mobile wireless computing. In *Proceedings of the Second International Conference on Embedded Software and Systems, Xian, China*, pages 375–383. Springer-Verlag, 2005.

[99] Steffen Maier, Daniel Herrscher, and Kurt Rothermel. On node virtualization for scalable network emulation. In *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS05)*, pages 917–928, Philadelphia, PA, July 2005. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Simulation Councils, Inc.

[100] A. MCDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1466–1487, August 1999.

[101] C. Mohan, B. Lindsay, and R. Obermarck. Transaction management in the r* distributed database management system. In *ACM Transactions on Database Systems*, volume 11, pages 378–396, 1986.

[102] Stefan Murawski. Reliable data dissemination using a cluster overlay in mobile ad-hoc networks. Master's thesis, Freie Universität Berlin, January 2007.

[103] A. Murphy, G. Picco, and G. Roman. Lime: A middleware for physical and logical mobility. In *ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems*, page 524, Washington, DC, USA, 2001. IEEE Computer Society.

[104] A. Murphy, G. Picco, and G. Roman. Lime: A coordination middleware supporting mobility of hosts and agents. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15:279–328, 2006.

[105] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM.

[106] Nadia Nouali, Anne Doucet, and Habiba Drias. A two-phase commit protocol for mobile wireless environment. In *ADC '05: Proceedings of the 16th Australasian database conference*, pages 135–143, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.

[107] J. Nuevo and J.C. Gregoire. Proposition of a Hierarchical Service Distribution Architecture for Ad Hoc Networks based on the Weighted Clustering Algorithm. *Proceedings of the 5th European Wireless Conference, Barcelona, Spain*, 2004.

[108] Sebastian Obermeier, Joos-Hendrik Böse, Stefan Böttcher, Panos Kypros Chrysanthis, Alex Delis, Le Gruenwald, Anirban Mondal, Aris Ouksel, George Samaras, and Stratis Viglas. 06431 working group summary: Atomicity in mobile networks. In Stefan Böttcher, Le Gruenwald, Pedro Jose Marrón, and Evaggelia Pitoura, editors, *Scalable Data Management in Evolving Networks*, number 06431 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[109] Katia Obraczka, Gene Tsudik, and Kumar Viswanath. Pushing the limits of multicast in ad hoc networks. In *Distributed Computing Systems, 2001. 21st International Conference on.*, pages 719–722, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[110] Object Management Group (OMG). Object transaction service. OMG Document, 2003. Version 1.4.

[111] Prasanna Padmanabhan and L. Gruenwald. Managing data replication in mobile ad-hoc network databases. volume 0, page 69, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[112] Marc R. Pearlman, Zygmunt J. Haas, Peter Sholander, and Siamak S. Tabrizi. On the impact of alternate path routing for load balancing in mobile ad hoc networks. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 3–10, Piscataway, NJ, USA, 2000. IEEE Press.

[113] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[114] Wei Peng and Xi-Cheng Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 129–130, Piscataway, NJ, USA, 2000. IEEE Press.

[115] Charles E. Perkins, editor. *Ad Hoc Networking*. Addison Wesley, 2001.

[116] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, 1994.

[117] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90, Washington, DC, USA, 1999. IEEE Computer Society.

[118] Petra Pfiedler. Simulative evaluation des shared log spaces (sls). Master's thesis, Freie Universitaet Berlin, December 2007.

[119] Roberto De Prisco, Butler Lampson, and Nancy Lynch. Revisiting the paxos algorithm. *Theoretical Computer Science*, 243(1-2):35–91, 2000.

[120] Krithi Ramamritham and Calton Pu. A formal characterization of epsilon serializability. *Knowledge and Data Engineering*, 7(6):997–1007, 1995.

[121] T. Rapport. *Wireless communications, principles and practice*. Pentrice Hall, 2nd edition, 2002.

[122] Yoav Raz. The dynamic two phase commitment (d2pc) protocol. In *Database Theory - ICDT '95*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1995.

[123] P. Krishna Reddy and Masaru Kitsuregawa. Blocking reduction in two-phase commit protocol with multiple backup sites. In *DNIS '00: Proceedings of the International Workshop on Databases in Networked Information Systems*, pages 200–215, London, UK, 2000. Springer-Verlag.

[124] P.Krishna Reddy and Masaru Kitsuregawa. Reducing the blocking in two-phase commit protocol employing backup sites. In *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems*, pages 406–416, Los Alamitos, CA, USA, 1998. IEEE Computer Society Washington, DC, USA.

[125] Kurt Rothermel, Christian Becker, and Jörg Hähner. Consistent update diffusion in mobile ad hoc networks. Technical Report 2002-04, Department of Computer Science, Department of Computer Science, July 2002.

[126] Kurt Rothermel and Stefan Pappe. Open commit protocols tolerating commission failures. *ACM Transactions on Database Systems (TODS)*, 18(2):289–332, 1993.

[127] E. Royer, P. Melliar-Smith, and L. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *IEEE Intern. Conference on Communications (iCC)*, Helsinki, Finnland, 2001. IEEE.

[128] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy. Paths: analysis of path duration statistics and their impact on reactive manet routing protocols. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 245–256. ACM Press New York, NY, USA, 2003.

[129] Amit Kumar Saha and David B. Johnson. Modeling mobility for vehicular ad-hoc networks. In *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 91–92, New York, NY, USA, 2004. ACM.

[130] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *Proceedings of the 4th International Conference on Mobile Data Management*, pages 13–28. Springer-Verlag London, UK, 2003.

[131] Y. Saito and M. Shapiro. Optimistic replication. *ACM Computing Surveys (CSUR)*, 37(1):42–81, 2005.

[132] P. Sens, L. Arantes, and M. Bouillaguet. Asynchronous implementation of failure detectors with partial connectivity and unknown participants. *Arxiv preprint cs.DC/0701015*, 2007.

[133] Patricia Serrano-Alvarado, Claudia Roncancio, and Michel Adiba. A survey of mobile transactions. *Distributed Parallel Databases*, 16(2):193–230, 2004.

[134] Amit P. Sheth and Marek Rusinkiewicz. On transactional workflows. *Data Engineering Bulletin*, 16(2):37–40, 1993.

[135] Dale Skeen. Nonblocking commit protocols. In *SIGMOD '81: Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, pages 133–142, New York, NY, USA, 1981. ACM Press.

[136] Dale Skeen. A quorum-based commit protocol. Technical report, Cornell University, 1982.

[137] Dale Skeen and Michael Stonebraker. A formal model of crash recovery in a distributed system. *IEEE Transactions on Software Engineering*, 9(3):219–228, 1983.

[138] James W. Stamos and Flaviu Cristian. Coordinator log transaction execution protocol. *Distributed Parallel Databases*, 1(4):383–408, 1993.

[139] William Su, Sung-Ju Lee, and Mario Gerla. Mobility prediction and routing in ad hoc wireless networks. *International Journal of Network Management*, 11(1):3–30, 2001.

[140] Masahiro Tamori, Susumu Ishihara, Takashi Watanabe, and Tadanori Mizuno. A replica distribution method with consideration of the positions of mobile hosts on wireless ad-hoc networks. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 331–335, Washington, DC, USA, 2002. IEEE Computer Society.

[141] Jian Tang and Jari Veijalainen. Transaction-oriented work-flow concepts in inter-organizational environments. In *CIKM '95: Proceedings of the fourth international conference on Information and knowledge management*, pages 250–259, New York, NY, USA, 1995. ACM Press.

[142] Andreas Thaler. Reliability evaluation of service providers in mobile ad-hoc networks. Master's thesis, Freie Universität Berlin, 2006.

[143] Robert H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems (TODS)*, 4(2):180–209, 1979.

[144] Jing Tian, Joerg Haehner, Christian Becker, Illya Stepanov, and Kurt Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *Simulation Symposium, 2002. Proceedings. 35th Annual*, pages 337–344, Los Alamitos, CA, USA, 2002.

[145] Danny Tschirner. Experimentelle und analytische untersuchung des area graph-based mobility models. Master's thesis, Freie Universität Berlin, January 2006.

[146] Yu-Chee Tseng, Sze-Yao Ni, and En-Yu Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *IEEE Transactions on Computers*, 52(5):545–557, 2003.

[147] Jeffrey D. Ullman, Jennifer Widom, and Hector Garcia-Molina. *Database Systems: The Complete Book.* Prentice Hall, 2001.

[148] Jari Veijalainen, Frank Eliassen, and Berhard Holtkamp. *Database transaction models for advanced applications*, chapter 12 The S-Transaction Mode, pages 467–513. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[149] Kumar Viswanath and Katia Obrazcka. An adaptive approach to group communications in multi hop ad hoc networks. In *ISCC '02: Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, page 559, Washington, DC, USA, 2002. IEEE Computer Society.

[150] Helmut Wächter and Andreas Reuter. *Database Transaction Models for Advanced Applications*, chapter 7 The ConTract model, pages 219–263. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[151] G.D. Walborn and P.K. Chrysanthis. Transaction processing in promotion. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 389–398. ACM New York, NY, USA, 1999.

[152] Gerhard Weikum and Hans-Jorg Schek. Concepts and applications of multilevel transactions and open nested transactions. In *Database Transaction Models for Advanced Applications*, pages 515–553. 1992.

[153] Gerhard Weikum and Gottfried Vossen. *Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[154] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194–205, 2002.

[155] W. Wu, J. Cao, and M. Raynal. The eventual clusterer oracle and its application to consensus in manets. *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, pages 23–32, 2007.

[156] K. Xu and M. Gerla. A heterogeneous routing protocol based on a new stable clustering scheme. *MILCOM 2002. Proceedings*, 2, 2002.

[157] Liangzhong Yin and Guohong Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.

[158] Aidong Zhang, Marian Nodine, Bharat Bhargava, and Omran Bukhres. Ensuring relaxed atomicity for flexible transactions in multidatabase systems. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 67–78, New York, NY, USA, 1994. ACM Press.

[159] J. Zheng, X.C. Lu, and Y.J. Wang. Clustering-based data replication algorithm in mobile ad hoc networks for improving data availability. In *Parallel And Distributed Processing And Applications: Second International Symposium, Ispa 2004, Hong Kong, China, December, 2004, Proceedings*. Springer, 2005.

[160] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1999.

Appendix C

Anhang gemäß Prüfungsordnung

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig angefertigt und alle verwendeten Hilfsmittel vollständig und genau angegeben habe. Die Arbeit wurde bisher in dieser oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 12. November 2008

(Joos-Hendrik Böse)

Zusammenfassung

Mobile Ad-Hoc Netze (MANETs) sind drahtlose Kommunikationsnetze, die von mobilen Geräten gebildet werden und ohne feste Infrastruktur wie z.B. Basisstationen etc. auskommen. Um die Kommunikation zwischen Knoten zu ermöglichen, die nicht in direkter Kommunikationsreichweite liegen, leitet jedes Gerät Nachrichten für andere weiter. Der ständige Topologiewechsel, die beschränkten Ressourcen (Systemleistung, Energie) der mobilen Netzknoten, mögliche Interferenzen, etc. bedingen eine höhere Wahrscheinlichkeit für Kommunikationsfehler als in festen Netzen. Die unzuverlässigere Kommunikation in MANETs wirft die Frage nach der Datenkonsistenz und Datenintegrität in verteilten Anwendungen in diesem Kommunikationsumfeld auf.

Konsistenz und Integrität verteilter Daten werden in festen Netzen mit Hilfe von atomaren verteilten Transaktionen gewährleistet. Die vorliegende Arbeit überträgt diese Methode auf MANET-Umgebungen, analysiert die sich daraus ergebenen Problemstellungen und diskutiert geeignete Lösungsansätze. Insbesondere werden die Abbruch- und Blockierungsrisiken von verteilten Transaktionen untersucht. Dafür werden spezifische Wahrscheinlichkeits- und Simulationsmodelle entwickelt.

Da es grundsätzlich nicht möglich ist, Blockierungssituationen vollständig zu verhindern, muss das Blockierungsrisiko überwacht und gegebenenfalls mit entsprechenden Methoden begegnet werden. Zur Lösung dieser Probleme werden folgende Beiträge präsentiert:

- Ein Berechnungsmodell zur Bestimmung der Abbruch- und Blockierungswahrscheinlichkeit von verteilten Transaktionen in MANETs für unterschiedliche Transaktionsmodelle.
- Ein Konzept - der Shared Log Space (SLS) - zur Aufhebung von Blockierungssituationen, die durch Fehler von Transaktionsteilnehmern ausgelöst werden. Der SLS ermöglicht es, den blockierten Knoten die Transaktionsentscheidung mit einer definierten Wahrscheinlichkeit zu kommunizieren und ihren Blockierungszustand zu verlassen. Zwei SLS Implementierungen werden vorgestellt und diskutiert.
- Einführung eines Backupkoordinators zur Reduzierung des Blockierungsrisikos durch Fehler des Transaktionskoordinators, wobei ein Wahrscheinlichkeitsmodell die Berechnung des Blockierungsrisikos bei dessen Verwendung erlaubt. Zusätzlich werden Strategien vorgestellt, um den Backupkoordinator mit dem SLS zu verknüpfen.

Diese Modelle und Methoden sind Vorraussetzung für ein adaptives Risikomanagement während der Transaktionsverarbeitung in MANETs. Die Einführung eines adaptiven Risikomanagements ist sinnvoll, da die Anwendung der in der Arbeit entwickelten Wahrscheinlichkeitsmodelle auf unterschiedliche Transaktions- und Kommunikationsszenarien gezeigt hat, dass nur für ein bestimmtes Spektrum von Transaktionen ein signifikantes Blockierungsrisiko existiert. Diese Transaktionen zu identifizieren und ihr Blockierungsrisiko zu kompensieren ist der grundlegende Beitrag dieser Arbeit.