Transactions in Mobile Systems: a Systematic Literature Review

Simon Matejetz
2978716
simon.mat@t-online.de
University of Stuttgart

Tobias Mathony
2956055
mathony.tobias@gmail.com
University of Stuttgart

Adrian Wersching
2947909
adrian.wersching@web.de
University of Stuttgart

***Abstract*—The importance of mobile transactions grows as more mobile devices are used in general or in trending paradigms such as Industry 4.0 and the Internet of Things. Even though research on traditional transaction protocols is far advanced, mobile transaction protocols face different challenges and thus, traditional transaction protocols may not be applied to the mobile context directly. Mobile devices are limited by battery and processing power and are also prone to regular disconnections. These issues have to be reconsidered when executing transactions in distributed systems including mobile devices. In this paper, we describe the execution of a systematic literature review on transactions in mobile systems and present and discuss our findings.**

***Index Terms*—Mobile transactions, Databases, Transaction protocols, ACID, BASE**

I. Introduction

Due to emerging paradigms such as Industry 4.0, Internet of Things (IoT) and connected vehicles, the number of embedded and mobile devices has increased constantly over recent years [1]–[3]. This holds especially true for mobile devices such as smartphones, as they are ubiquitous in everyday life. Many devices have to be able to handle transactions of some sort. This does not only include financial transactions but transactions involving data of any form which have to adhere to certain rules. Transaction properties such as Atomicity, Consistency, Isolation and Durability (ACID) are well known in traditional database systems and are considered essential to guarantee correct data management [4].

However, these properties cannot be translated to the mobile context directly [4]. Transactions in mobile systems pose a variety of challenges and limitations which require revisiting traditional transaction properties [4]–[6]. Mobile devices typically have limited resources, e.g. battery and processing power, memory/storage capabilities, or screen size [4]. Furthermore, as mobile devices by definition are not bound to a certain location like other devices, they can suffer from frequent disconnections and network changes [7], [8]. Therefore, the movement of mobile devices and interchanging, heterogeneous networks of different kinds (local, wireless, ad-hoc, etc.) need to be considered with respect to the transaction properties or even when designing a transaction protocol [4], [8]. Moreover, wireless networks tend to have a variable and low bandwidth, which, in addition to usually high transmission costs, makes bandwidth consumption an important factor [4].

These restrictions and limitations result in different drawbacks for mobile devices. While a transaction may succeed, execution times can vary highly due to bandwith capacity and communication costs Furthermore, more battery is consumed when being affected by a low bandwidth, and, in worst case, transactions may fail due to unexpected disconnections or battery breakdown [4].

To provide an overview of current research, we conducted a systematic literature review on transactions in mobile systems and environments. The basis of our work is "A Survey of Mobile Transactions" from Serrano-Alvarado et al.[4], an extensive survey and comparation of mobile transaction models and protocols, which was initially released in 2004 and then extended in 2009. Therefore, we decided to focus exclusively on results published later than 2009. This also helped to collect relevant results on trends developed since 2009, e.g. Industry 4.0 and IoT.

The rest of the paper is structured as follows: Section II explains the foundations and background on transactions in mobile systems based on the work from Serrano-Alvarado et al. [4], Section III describes the execution of our systematic literature review, while Section IV presents the most promising and interesting mobile transaction proposals found by the execution of our systematic literature review. Section V contains a conclusion about our paper and an outlook on further research directions.

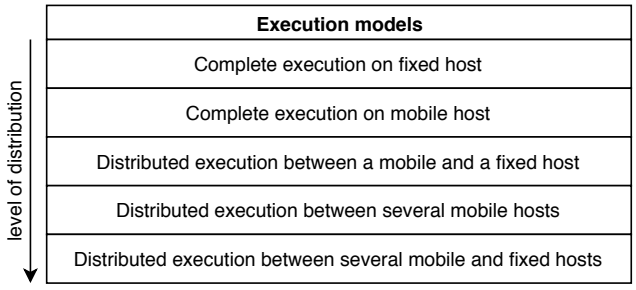

Fig. 1. Different execution models as proposed by Serrano et al. [4]

II. Background

The goal of this paper is to revisit the current state of mobile transaction protocols. We based our works on the investigation of the state-of-the-art in mobile transaction protocols from Serrano-Alvarado et al. [4], [9]. They initially released their survey in the year 2004 and revised their work in 2009. While mobile transactions were already important during the publication of said paper, their importantance has increased even further since then with emerging paradigms such as the IoT and more mobile phones existing than ever [1].

Serrano-Alvarado et al. generally differentiate between mobile hosts (MHs), e.g. a mobile device or a car, and fixed hosts (FHs), e.g. a server. Furthermore, there is no specific architecture which protocols have to adhere to. Both MHs and FHs can be either server or client [4]. While Serrano-Alvarado et al. do not require MHs to be a specific kind of device, the following properties are assumed for MHs: (i) has storage capabilities, (ii) can run Database Management System (DBMS) and (iii) has wireless network capabilities [4].

MHs are limited by several restrictions imposed by the nature of the network and hardware of the devices itself. Data transmission to MHs can be extensive and can suffer from low bandwidth. Furthermore, as MHs move around, they do not stay connected to the same access points. In order to achieve a certain degree of portability, MHs are restricted in battery power, processing and storage capabilities. Therefore, disconnections and other failures have to be considered normal for the communicaton with MHs, this holds especially true for transactions executed between FHs and MHs. Serrano-Alvarado et al. define a mobile transactions as a transaction which involves at least one MH [4].

A. Mobile Transaction Context

Serrano-Alvarado et al. define different characteristics of mobile transactions to categorize the various protocols and approaches. In general, different kinds of client-server architectures have to be considered. MHs are considered *thin clients* if the main operations are executed on the FH servers. MHs are considered *full clients* if they can emulate server functionality. Full clients can be used to enable the MHs to work offline. A *flexible client-server architecture* allows for dynamic changes of clients and servers. *Client-agent-server architectures* use a third instance to manage the communication between the clients and the servers [4].

Furthermore, several operation modes have to be considered. If a MH moves within the same network, it is considered *micro* mobility. If a MH moves between different networks, it is considered *macro* mobility [4].

Serrano-Alvarado et al. also define different execution models for mobile transactions, which are depicted in Figure 1, ordered by their level of distribution. The following kinds of execution are considered: (i) complete execution on FH, (ii) complete execution on MH, (iii) distributed execution between a MH and FH, (iv) distributed execution between several MHs, and (v) distributed execution between several MHs and several FHs [4]. For a complete execution on FH, a mobile transaction is initiated by a MH but executed completey on a FH [4]. When a mobile transaction is completely executed on an MH, the transaction is both initiated and executed on the MH, which enables an isolated execution of a mobile transaction, even if there is no connection to a server [4]. One of the constraints of this execution model is the neccesity of having server capabilities and all relevant data on the MH to execute the transaction locally [4]. A more flexible, but also more complex operation mode, is the distributed execution between a MH and FH, where the transaction execution is distributed based on optimization reasons or resource availability [4]. To enable a distributed execution between several MHs, MHs embody a server for other MHs [4]. This could be useful if another MH is closer than a database server, regarding the communication [4]. For a fully distributed execution between several MHs and several FHs, the peer-to-peer approach of MHs is extended by multidatabases of FHs [4].

B. ACID, BASE, SACReD and CAP

Atomicity, Consistency, Isolation and Durability (ACID) [10], Basically Available, Soft state and Eventual consistency (BASE) [11] and Semantic Atomicity, Consistency, Resiliency and Durability (SACReD) [5] are transaction correctness criteria.

1) ACID: ACID is considered to be essential to achieve transaction correctness [4]. *Atomicity* describes that either all operations of a transaction must be executed, or none of them [4], [10]. *Consistency* means, that the state of the database after the execution of a transaction remains consistent [10]. Each transaction execution must be isolated from other, concurrently running transactions, which is represented by the *Isolation* property of ACID. Furthermore, *Durability* implies that, once a transaction commits, its effects on the database are durable and must survive any possible malfunctions [4], [10].

2) BASE: In contrast to ACID, BASE focuses on higher availability at the expense of hard consistency. *Basically available* means, that the database will be available most of the time. *Soft state* refers to the possibility of inconsistent states, while *Eventual consistency* implies that the state of the database will be consistent at a later point in time [11].

3) SACReD: SACReD is a transaction correctness criteria for mobile services transactions introduced by Younas et al. [5], [12]–[14]. *Semantic Atomicity* enables that component transactions are allowed to commit independently from their parent mobile services transaction. However, after execution of a transaction, sources of information need to achieve *Consistency* [13]. *Durability* is defined the same way as in traditional ACID transaction criteria. *Resilience* allows committing even if failure or service unavailability occurs, which is achieved by associating alternative transactions with each sub-transaction [13].

4) CAP: The Consistency, Availability and Partition Tolerance (CAP) theorem [15] states, that it is impossible to achieve the three desirable properties *Consistency*, *Availability* and *Partition tolerance* at the same time in a distributed system which uses data from different locations [7], [15].

III. Systematic Literature Review

The following section will explain our way of proceeding during the systematic literature review. To start off, our research questions will be explained in Section III-A. Afterwards, we will present our inclusion and exclusion criteria for the search results in Section III-B, before we proceed with the search keywords in Section III-C. The search process itself and the data collection will be presented in Section III-D and Section III-E respectively. Finally, our search results and their evaluation will be described in Section III-G. Concludingly, in Section III-H, limitations of our systematic literature review will be discussed.

A GitHub repository[1] containing our research questions, keywords, queries, search script and results was set up aditionally to the following explanations of our systematic literature review.

A. Research Questions

Initially, we defined nine research questions for our systematic literature review. While formulating the research questions, we focussed on current and past research directions of mobile transactions and what progress has been made, especially in the apparent most-challenging fields. In the first steps, we wanted to evaluate how mobile transaction protocols differ from traditional transaction protocols and what mobile transaction protocols exist in general. Furthermore, we aimed to investigate mobile transaction protocols in the context of ACID, BASE and CAP. Since transactions in mobile context have to handle frequent disconnections, we also wanted to inspect compensation techniques of mobile transaction protocols. Moreover, we looked to address current limitations and open research questions of mobile transaction protocols. Finally, we decided on following research question for our systematic literature review:

1. How do mobile transaction protocols differ from traditional transaction protocols?
2. What types of mobile protocols exist?
3. What progress has been made regarding distributed execution models (MH to MH, multiple MH to multiple FH)?
4. What progress has been made regarding movement and disconnection management in the past years?
5. How are properties of ACID and BASE fulfilled by mobile transaction protocols?
6. How are mobile transaction protocols regarding the CAP theorem?
7. What type of scenarios require what type of mobile transaction protocols?
8. What are the compensation techniques in mobile transactions?
9. What are the main limitations and open research questions of mobile transaction protocols at the moment?

[1] https://github.com/mathonto/MobileTransactionsSLR

B. Inclusion and Exclusion Criteria

After defining our research questions, we formulated inclusion and exclusion criteria for our search results. Since we based our systematic literature review on works from Serrano et al. [4], which was revised again in 2009 after it was initially released in 2004, we decided to only include results published after 2009. As our research questions were broadly diversified, we initially decided not to use another exclusion criteria. However, we refined our exclusion criteria after noticing a multitude of untechnical papers related to financial transactions within the first results. Furthermore, many results included caching, routing and energy saving for mobile devices or networks, which we wanted to exclude as they do not address our research questions.

C. Search Keywords and Queries

Based on our research questions, we decided on the search keywords for our systematic literature review depicted in Figure 2. Every node in the figure is a search keyword. Nodes which are connected via an edge build a compound search keyword, e.g. we had many search keywords starting with *Mobile*, for example *Mobile Transactions*, *Mobile Transaction Models* or *Mobile Environment*. Search keywords which are related strongly are grouped in the figure via a border, for example the search keywords *Resilient Connectivity*, *Disconnection Management*, *Disconnection Handling*, *Seamless connectivity* and *Compensation* as they all cover connection-related topics of mobile transactions.

To cover all research questions, the search keywords had to be connected via operators. We used a set of four search operators for our search process. This included *AND*, *OR*, *()* and *. The *AND* operator, as it implies, was used to connect two or more search keywords with a logical AND, i.e. that a paper has to contain all logically AND connected search keywords to appear as a search result to such query. Same applies to the *OR* operator, which connects two or more search keywords with a logical OR, which was helpful e.g. to include synonyms of our search keywords in a search query. The *()* operator was used to group search keywords within a search query and thus, control the logic of a query, while the * operator acts as a wild-card and matches on any word. This operator was used to cover alternative formulations following a certain keyword.

Using the search keywords and operators, we formulated a variety of search queries. Only search queries without a operator were: (i) *Mobile transaction protocols* and (ii) *Mobile transaction models*. For all further search queries, we used our set of operators. For the majority of search queries we used the *AND* operator to connect two or more search keywords from Figure 2. With the search keyword *Mobile transaction*, we built following search queries: (i) *Mobile transactions AND IoT*, (ii) *Mobile transactions AND Embedded*, (iii) *Mobile transactions AND (BASE OR ACID)*, (iv) *Mobile transactions AND CAP*, (v) *Mobile transactions AND consistency*, (vi) *Mobile transactions AND compensation*, (vii) *Mobile*

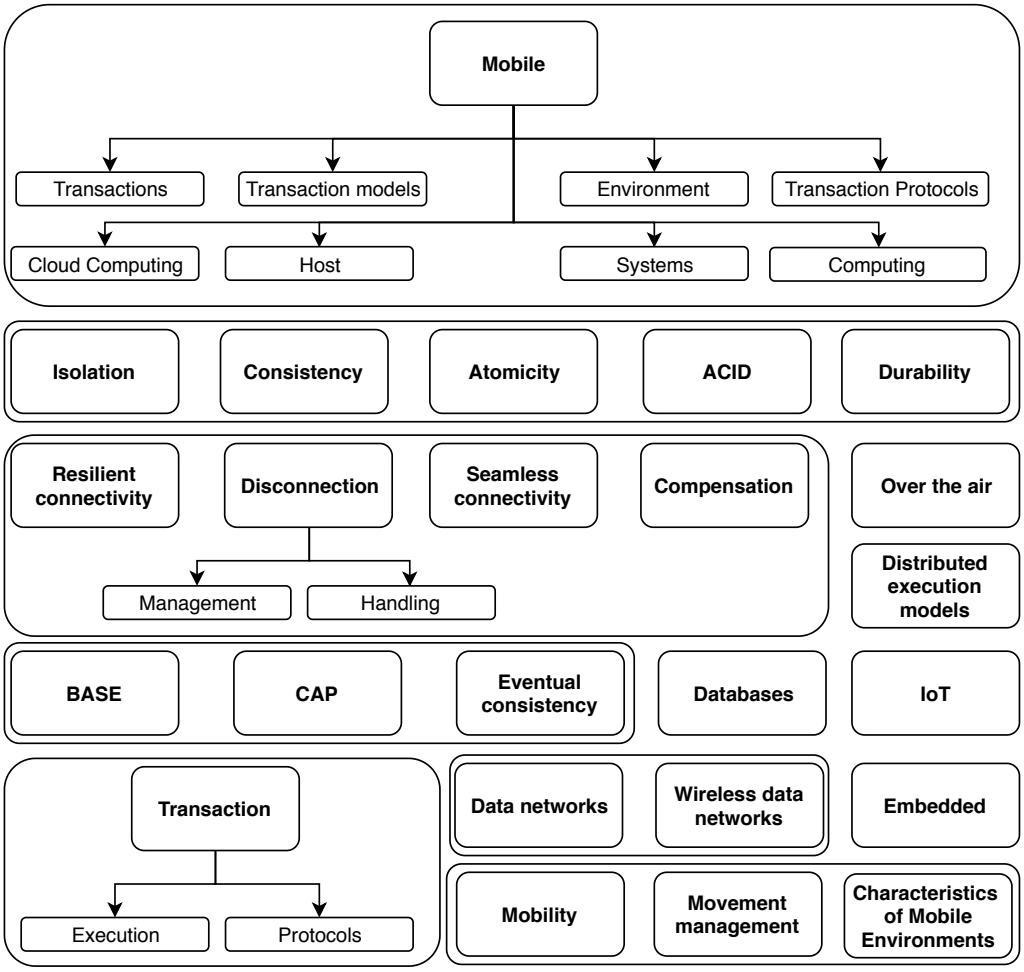

Fig. 2. Aggregation of search keywords

transactions AND (atomicity OR consistency OR isolation OR durability), (viii) *Mobile transactions AND Disconnection handling*, (ix) *Mobile transactions AND Resilient connectivity*, (x) *Disconnection management AND (mobile environments OR mobile systems OR mobile transactions)*, (xi) *Compensation AND (mobile environments OR mobile systems OR mobile transactions)*, (xii) *Mobile transactions AND Embedded systems*, and (xiv) *Mobile transactions AND execution*.

Further search queries covering our research questions about disconnection management and mobility were (i) *Disconnection handling AND mobility*, (ii) *Resilient connectivity AND mobility*, (iii) *Movement management AND transactions*, (iv) *Embedded systems AND disconnection AND (management OR handling)*, (v) *Wireless data networks AND disconnection AND (management OR handling)*, and (vi) *Movement AND compensation AND (IoT OR Embedded OR mobile)*.

The search queries (i) *Portable device* AND transaction* AND mobile*, (ii) *Mobile AND cloud computing AND transactions*, and (iii) *Seamless connectivity AND mobile* completed our set of search queries.

D. Search Process

To maximize the reproducibility of the search process, it was conducted programatically. A custom Python script was developed to crawl the scientific search engines Google Scholar[2], Springer Link[3] and IEEE Xplore[4] using the defined queries. While Springer Link and IEEE Xplore offer access to their own curated databases, Google Scholar enables more general searches not specific to one provider. The result of our search process therefore was a broad spectrum of literature.

E. Data Collection

Each query was executed seperately on each of the selected search engines. The results of each query were then collected in a comma-separated values (CSV) file. The following metadata was collected for each result:

- Title
- Authors
- Date

[2]https://scholar.google.de/
[3]https://link.springer.com/
[4]https://ieeexplore.ieee.org/Xplore/home.jsp

- Link to result
- Link to the PDF of the result
- The query which produced the corresponding result
- The search engine that was used
- The number of times the result has been cited
- The Digital Object Identifier (DOI) of the result

The results from all queries and search engines were then collected into a single CSV file. At this point, roughly 245,000 possible results remained. As all queries were executed on all search engines, duplicate results could be possible. These results were therefore removed from the aggregated CSV file. Two results were considered equal if (i) the link of both results was equal or (ii) the DOI of both results was equal or (iii) the titles of both results were at least 75% equal according to the Levenshtein distance. To provide an additional metric to quantify the relevance of a result, the duplicates of each result were counted. After all duplicates had been removed from the aggregated CSV file, roughly 94,000 results remained.

F. Data Analysis

In order to extract the relevant results from our current data set, we created the following subsets from the aggregated CSV file (i) all results from Google Scholar, (ii) all results from IEEE Xplore, (iii) all results from Springer Link with at least ten duplicates found during the search process and (iv) all results from Springer Link which had been cited at least fifty times. As the Google Scholar and IEEE Xplore results were already limited to a manageable amount, they were not further divided. This left us with roughly 1500 results for the first data set, 200 results for the second data set, 1900 results for the third data set and 1900 results for the fourth data set.

As this data set still contained a lot of literature not relevant to our work, we began screening the remaining results manually. A result was included or excluded based on its title. If the title did not enable us to make a clear decision, we also read the abstract of the paper and then revisited our choice of exclusion or inclusion. There now remained approximately 570 potential results.

The base of this paper is the work of Serrano-Alvarado et al. [4]. Serrano-Alvarado et al. conducted a comparative survey of mobile transactions in the year 2009. To build upon this survey, we decided to only consider literature published after 2009. This left us with approximately 160 results.

For the final step of our data analysis, we read the abstracts of all of the remaining results. We then assigned at least one category to each result. Furthermore, if the abstract of the paper suggested that it would not be relevant for our work, we discarded it. After we had categorized and filtered the remaining results, our final data set consisted of approximately 50 results.

As this paper is based on the work of Alvarado-Serrano et al., we also extended the proposed categories [9]. The specific categories used in this paper are depicted in Figure 3.

We adopted the categories *architecture*, *operation modi*, *mobility modi* and *execution* as described by Alvarado-Serrano et al. [4]. Additionally, we introduced the transaction categories ACID, BASE and SACReD to categorize the proposed protocols based on transaction properties, especially since BASE and SACReD represent more relaxed transaction properties that are more suitable for mobile transaction protocols and models. Furthermore, we introduced several miscellaneous categories. In order to classify other survey papers conducting a comparative analysis of multiple protocols, we used the *survey* category. If we found that a result was not relevant to our work at this stage, we excluded it from our further result set. As our results included multiple papers on mobile ad-hoc networks (MANET), we added an additional category for that.

Architecture	
Thin client	Full client
Flexible client-server	Client-agent-server
Operation modi	
Connected	Disconnected
Strong	Weak
Doze	
Mobility modi	
Micro	Macro
Execution	
On MH	On FH
Distributed between MH and FH	Distributed between MHs
Distributed between MHs and FHs	
Transaction properties	
ACID	BASE
SACReD	
Other	
MANET	Survey
Not relevant	

Fig. 3. Extended categories based on [4]

G. Search Results

It was noticeable, that many of our initial results included papers related to financial transactions, routing in networks and caching in mobile context, which were not target of our research questions. This eventually led to a refinement of our exclusion criteria, as described in Section III-B. After investigating further, we analyzed which queries were responsible for these results. Most of the results related to financial transaction resulted from the query *portable + device* + AND + transaction* + AND + mobile*, which was possibly formulated too generally to produce useful results. A majority of results similar to routing and caching were caused by the query *Mobile + transactions + AND + (atomicity + OR + consistency + OR + isolation + OR + durability)*. One reason for this could be that the many OR links enabled a very wide variation of search results. Furthermore, many results were not

related to mobile transactions or mobile databases at all, which were excluded from our result set manually, based on the title.

Our final data set included around 50 results, as described in Section III-F. The same query *Mobile + transactions + AND + (atomicity + OR + consistency + OR + isolation + OR + durability)* which caused a multitude of unwanted results was responsible for almost 20% of our final results. Another query that contributed significantly to the results was *portable + device* + AND + transaction* + AND + mobile*, which is another query that also led to unsuitable results as mentioned before. Especially these two queries were responsible for very good results, as well as unsuitable results, which is probably due to the rather general formulation of these queries, which led to a multitude of unsuitable results, but also to good results which were not covered by other queries. The rest of the results was obtained realtively evenly distributed through the rest of our queries.

H. Limitations

As most search engines do not provide Application Programming Interfaces (APIs), the programmatic search process had to be performed by imitating a regular web browser and parsing the resulting HTML pages. Only IEEE Xplore provides an API. However, it has a strict limit to the number of queries that can be performed each day. Therefore, only the first 35 results of a given query were considered for IEEE Xplore.

Google Scholar also does not provide an official API. Furthermore, it employs a strict protection against automated crawling. After querying about forty pages, a temporary ban of 24 hours was imposed. We therefore limited the Google Scholar results to ten pages per query. One page of Google Scholar results contains ten results.

Springer Link does not provide an official API either but does not implement any kind of restrictions against automated crawling. However, Springer Link cannot display results after page 1000. We therefore limited the Springer Link queries to 1000 pages.

It should also be noted that not all results provide the same metadata. The only metadata associated with every result was title, authors and link of the result. More specifically, the metadata properties date, PDF link, citation cound and DOI were not always present.

IV. Mobile Transaction Proposals

Due to our base survey paper [4] being from 2009, our main focus for this paper was to find technologies that weren't proposed in 2009 when the survey was originally published. Thus, the following section presents some of the most interesting proposals since 2009 that were found within our previously described systematic literature review (see Section III). Furthermore, our findings are categorized using the categories from Serrano et al. [4], which were explained and extended in Section II. As *connected* and *strong* are considered to be default operation modi, they will not be mentioned explicitly when categorizing the protocols. Additionally, *micro* mobility is supposed to be supported per default by mobile transaction models and will also not be mentioned explicitly.

A. Surrogate Object Model (Refined Kangaroo Transaction Model)

In their paper "An Improved Kangaroo Transaction Model Using Surrogate Objects for Distributed Mobile System", released in 2011, Ravimaran et al. [16] propose an improvement to the Kangaroo model [17], that was previously discussed in Serrano-Alvarado et al.'s [4] work. The Kangaroo transaction model takes into consideration the movement of a MH and delegates the transactions between MHs and Base Stations (BSs) to the BS that is responsible for the current, possibly changing, location of a MH. This is achieved by adding a Data Access Agent (DAA) to each of the BSs, to which the MHs connect. This DAA then handles the transition between different BSs by initiating a Kangaroo Transaction on each transaction requested by a MH. A Kangaroo Transaction splits a transaction that occurs during a transition into several sub transaction, called *Joey Transactions*, that are then delegated to the BSs between which the hand-off is done [17].

Ravimaran et al. [16] improved this transaction model by introducing surrogate objects that are attached to a BS and act on behalf of a MH. A surrogate object maintains current state information of the MH it belongs to. According to Ravimaran et al. this has several advantages over the previous approach like easier location management, improved data access speed by caching and ensured Quality of Service (QoS) by delivering data even when the requested MH is not currently connected to the same BS [16]. The proposal paper brings forward the example of communication between various MHs that are connected to different BSs. Instead of only communicating via the Database (DB)-Server, the proposed middleware looks up the surrogate object for MH_y in MH_xs currently connected BS for each Kangaroo substransaction. If the surrogate object and all requested data is found this cache hit is returned [16]. Else, the request is raised to adjacent and nearby BSs surrogate objects and finally to MH_y before the traditional DB lookup is done. In conclusion, the surrogate object model adds data replication as well as caching to the existing Kangaroo transaction model which, according to Ravimaran et al. decreases the overall amount of network traffic as well as providing a lower abort rate [16].

With the surrogate object, the proposed model has a *client-agent-server* architecture. It supports *disconnected* operation modi, as well as *macro* mobility. The mobile transaction itself is executed *on FHs*

B. A lazy commit protocol for mobile transactions

The Lazy Commit Protocol proposed by Pathak et al. [18] aims at reducing the bandwith used in a system supporting mobile transactions. This is achieved by employing a central entity managing locks on the available resources. If a device wants to execute a transaction, it has to acquire locks for the resources involved in the transaction from the central manager. The transactions are then executed locally on the device. Locks

are only released if another device wants to acquire them or the user specifically wants to do so [18]. Until this time, the result of the executed transactions remains on the device. Pathak et al. propose the Lazy Commit protocol under the assumptions that disconnections in the system are unlikely and devices remain connected to a single network [18].

As transactions are executed locally on the MH, the lazy commit protocol features a *full-client* architecture. Pathak et al. feature a *doze* operation mode with their lazy commit protocol, which is achieved by deferring the commit of transactions.

C. Perturbation-Resilient Atomic Commit Protocols for Mobile Environments

In his disseration from 2010, Brahim Ayari [19] designed a framework to support communication of applications in different types of mobile environments. For this framework, multiple atomic commit protocols with special use cases were designed. One year later a refined and condensed paper was published on the dissertation's basis [20] containing the three most interesting and promising proposals, which are presented in the following. The authors take into consideration several models of mobile environments regarding different environmental constraints as well as failures. The environmental constrains include heterogeneity of nodes and links, unstable storage and energy. For the failures, mainly network disconnections, message loss and node failures are considered.

1) Pre-Phase Transaction Commit (PPTC): The PPTC was designed with failure-free environments in mind. Despite being failure-free, these environments may suffer from environmental constrains. A proposed example for such a network is the industrial plant scenario with very heterogeneous and reliable nodes and stable wireless networks [20]. PPTC is based on a decoupling strategie so that when a transaction is initialized the commit is split in two phases, namely the pre-commit and core phase [20]. In the pre-commit phase, PPTC collects the vote of all mobile participants before a set timeout. If all mobile participants voted "Yes" in that time, the core phase is initialized. If one or more mobile participants do not answer in time or vote "No", the Coordinator aborts the transaction [20]. For the following core phase, any atomic commit protocol for wired networks, e.g. Two-Phase-Commit (2PC), can be used to find the final decision between the coordinator, that now holds the MHs decision, and the FHs. The decision then is sent to all MHs and FHs that were involved in the commit [20].

2) Fault-Tolerant PPTC (FT-PPTC): The FT-PPTC is an extension of the PPTC and additionally to environmental constraints tolerates network disconnections as well as message losses. This is achieved by adding an agent for every MH taking part in the commit [20]. A MH can inform its agent before disconnecting from the network, who will then communicate with the coordinator to delay the pre-commit phase timeout. After a decision on the commit has been made, the coordinator sends the decision to the agents to inform their possibly disconnected MH about it [20].

3) Fault-Tolerant and Recovery PPTC (FT-PPTC-Rec): FT-PPTC-Rec is a further extension of FT-PPTC and adds support of tolerating node failures [19], [20]. Thereby, the agent and the coordinator mentioned in Section IV-C2 log and checkpoint during the execution of a mobile transaction in order to be able to recover if a failure occurs. When a mobile transaction is initiated, the coordinator creates a token including one entry for every execution fragment of the mobile transaction, where each entry contains the current state of processing and the timeout values of execution and shipping time of each execution fragment [19], [20]. Thus, this token stores constantly the current state of each execution fragment of a mobile transaction and its state and, hence, allows recovery to each checkpoint, i.e. each execution fragment [19], [20]. FT-PPTC-Rec is especially suitable to mobile systems and environments which are estimated to suffer from a huge amount of message loss and frequent disruptions regarding the connection [19], [20].

As PPTC and FT-PPTC are only intermediate steps to the FT-PPTC-Rec protocol, only the FT-PPTC-Rec will be categorized. With its agents and coordinator, FT-PPTC-Rec represents a *client-agent-server* architecture. The approach supports both *disconnected* and *weak* operation modes. Furthermore, *macro* mobility is considered. The execution of a mobile transaction is *distributed between MHs and FHs*. *ACID* properties are enforced.

D. A model for transaction management in mobile databases

In their paper from 2010, Abdul-mehdi et al. [21] propose a model for transaction management in mobile environments that supports the disconnection of MHs while allowing asynchronous, parallel transactions between MHs and a BS, keeping accessed data eventually consistent in a certain timeframe. All MHs get assigned a fraction δ of the value of each data item in a distributed reservation system, e.g. number of flight seats available. δ is an integer that's larger than zero but smaller than the data item it belongs to. When a data item is requested by MHs, the BS replies with a time in which the δ is valid. The δ depends on the data items value, i.e. the sum of MHs currently disconnected and the amount of MHs requesting the data item, so that the sum of all δ plus the remaining data item's value is equal to the data items value before assigning the δ [21]. δ represents the range in which a MH may change the data items value, e.g. if the value is 300 and the δ a MH got assigned for that date item is 50, the MH may commit changes to the data item in the range of -50 to 50. A BS only considers transactions made before the timeout and in the specified δ for each MH, other transactions will be sent directly to the server [21]. Before the timeout, MHs may send their changes to a BS, which then, at the end of the timeout, calculates the total change that will be written to the master database. When the timeout occurs, the BS updates its data item value by $X_{BS} = X_{BS} + (\delta - C)$ for each MH, with X_{BS} being the current data item value in the BS and C being the changes towards the data item's value made by the MH in its δ range [21].

The model proposed by Abdul-medhi et al. implies a *full-client* architecture, where a MH can emulate server functionalities. The model is valid for a *disconnected* operation mode and features a *distributed execution between MH and FH*. Abdul-mehdi et al. do not mention *mobility* explicitly.

E. Generalized Mobile Transaction Commit (GMTC)

Ayari et al. [22] describe GMTC as a pertubation-resilient atomic commit transaction protocol for hybrid mobile environments. A hybrid mobile environment exists, if a MH supports both following communication modes: (i) infrastructure-based mobile environment and (ii) ad-hoc mobile environment [22]. Infrastructure-based communication is present if MH communications is infrastructure-based, e.g. through base stations, while ad-hoc communication is present if MH communicate only via ad-hoc mode [22]. With pre-commit and core phase, GMTC has two phases. In its pre-commit phase, it resembles FT-PPTC, but with multiple coordinators instead of a single one, which converge at the end of the mobile transaction to one coordinator. MHs which have an infrastructure-based connection are favoured when converging to a single coordinator. During the core phase and after converging to one single coordinator, the coordinator checks whether it has a complete list of all participating MHs required for the execution of the transaction. If this is the case, the state of the mobile tranaction is then set to "pre-committed" and a 2PC session is started [22].

As Ayari et al. assume a hybrid mobile environment, they follow a *flexible client-server* architecture. A *disconnected* operation mode is supported. The execution of a mobile transaction can be *distributed between MHs and FHs*. A *MANET* architecture is also considered and supported, where MHs establish an ad-hoc communication.

F. Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks

In their paper "Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks", Hara et al. [23] focus on data replication strategies in MANET to improve data availability in case of disconnections. Hara et al. [23] define five different levels of consistency according to demand, and design protocols to achieve these consistency levels.

1) Global Consistency Protocol: Global consistency requires traditionally that every write-operation is performed on all data replicas, which works badly in MANET due to frequent disconnections and network partitioning [23]. Hara et al. present a three-phase protocol to obtain global consistency in MANET, which consists of following phases: (i) a lock request, to globally lock read operations of replicas, (ii) write operations on replicas with locks and (iii) a message for commit and release the lock [23].

2) Local Consistency Protocol: Local consistency allows versions of replicas to differ between regions. The protocol to achieve this is basically the same as in Section IV-F1, but only applied to a certain region with data replicas [23].

3) Time-Based Consistency Protocol: For time-based consistency, peers do not have to read the latest version, but the latest version of replicas within a certain valid period. When a peer in a MANET aims to perform a write operation, and a peer has a replica of the targeted data, the write operation is performed on the replica. If the peer does not have the required data replica, it searchs for the closest peer in the MANET with the replica and performs the operation there [23].

4) Peer-Based Consistency Protocol: For peer-based consistency, each peer only has to read the version it last wrote, which means, it is only applicable to replicas a peer holds [23]. Furthermore, it requires every peer to have enough memory space to hold a replica of all data it accesses [23]. In the system model proposed by Hara et al., every transaction is considered as a singel operation, which implies that every peer can perform operations anytime on its own replica it holds, i.e. no further modification is neccessary [23].

5) Application-Based Consistency Protocol: Since application-based consistency provides all described consistency levels required from different applications, write operations are performed analogous to the global consistency protocol from Section IV-F1. The execution of read operations depends on the required consistency level of the peer issuing the request [23].

Following categorization is valid for all proposed consistency protocols desribed in this subsection. Due to its quorum-based approach MHs in *disconnected* mode in the MANET is supported if there are other MHs holding replicas of the required data item. The execution of the mobile transaction is *distributed between MHs*. The global consistency protocol only focuses on *MANETs*. *Macro* mobility is considered, however only under the assumption that there is no region without a certain required data replica.

G. Managing concurrent execution of transactions in mobile ad-hoc network database systems: an energy-efficient approach

Xing et al. [24] propose a protocol for MANETs which aims at decreasing transaction abort rate as well as energy consumption and increasing the general balance of energy consumption. In order to achieve these improvements, the network has to be split into different clusters, where each cluster has a head node managaging it. Xing et al. also propose an algorithm called Mobility, Energy and Workload (MEW) for the automatic clustering. This algorithm considers the respective mobility, energy and workload of each node during the creation of the clusters. Furthermore, a primary leader node is elected, which coordinates global communication and transactions [24]. After the clustering of the network, Xing et al. propose a new algorithm for the execution of mobile transactions called Sequential Order with Dynamic Adjustment (SODA). SODA allows for the dynamic reordering of transactions in two ways: simple and complex. In the simple case, a transaction can be inserted anywhere in the transaction history if no adjustment is needed in order for the

transaction to commit successfully. In the complex case, the transaction history is rearranged, the transaction is inserted and if it can commit successfully, the rearranged transaction history replaces the old one. This approach increases concurrency and reduces block time. Furthermore the abort rate is reduced [24].

The proposed approach by Xing et al. is based on a *flexible client-server* architecture due to the *MANET* nature. *Disconnected* operation mode is supported and mobile transactions are executed *distributed between MHs*. Xing et al. state that *ACID* properties are ensured by this approach.

H. Improving the success rate of concurrent Mobile Transactions by predicting time for execution

Moiz et al. [25] propose a transaction model for reducing the rollback rate in concurrent mobile transactions by predicting the time a MH will need for a certain transaction and thus decrease the amount of locked data items due to MHs that disconnected while keeping a lock. More specifically, the strategy is to default to traditional pessimistic locking until a meaningful expected time needed for calculation of a certain transaction can be calculated. For this to be possible, K-means and K-neighborhood is used on, at least ten, existing logs of similar transactions in MHs with similar performance to compute a time t in which a MH will most likely finish a transaction. To this t, a constant, representing the tolerance of delay in the transaction, is added to receive the final time T. When a MH requests a data item, a lock will be placed on the data item for the time T. The MH then also has T time to finish the transaction. If it is not finished in time, the transaction will be aborted and the lock on the data item is removed [25].

This approach by Moiz et al. cannot be classified in our categorization as it is a generally applicable model to improve pessimistic concurreny control techniques.

I. Context-aware Mobile Services Transactions

In their papers "Context-aware Mobile Services Transactions" [5], "A new model for context-aware transactions in mobile services" [12], "Failure Resilient Criteria for Mobile Web Services Transactions" [13] and "Mobility Management Scheme for Context-Aware Transactions in Pervasive and Mobile Cyberspace" [14], Younas et al. cover challenges of transactions in mobile services and propose protocols to overcome these challenges. In the context of these papers, they propose two new, more relaxed sets of transaction correctness criteria: (i) SACReD (see Section II-B3) and (ii) Relaxed atomicity, Consistency, Context, and Durability (RACCD) [5], [12]–[14].

RACCD introduces context as a main property of transactions and is considered by Younas et al. to be more suitable to the nature of mobile services transactions [5]. The protocol proposed by Younas et al. to enforce RACCD criteria is structured into several phases, where they at first acquire the context of the mobile service, then reserve the required services to execute, and eventually collect the required services to successfully commit the transaction [5].

To enforce SACReD, Younas et al. propose an approach using a transaction coordinator and several representative systems, which reflect services to be collected from a mobile service transaction [13]. The transaction coordinator, as well as the representative systems maintain log files about the execution of the mobile service transaction. When a mobile service transaction is issued, the transaction coordinator sends a message to all representative systems to begin the component service transaction and waits then for responses regarding the completion of the transaction [13]. The coordinator writes then abort or commit in the log file, depending on the response it receives from the representative system. If the coordinator receives an abort messsage, it initiates an alternative component service transaction. Eventually, the coordinator writes the global commit decision, sends it to all representative systems and terminates the mobile service transaction [13].

Furthermore, Younas et al. introduce a mobility management scheme for context-aware mobile services transactions, based on various queuing models, enabling an execution environment allowing free movement between cells while executing mobile services transactions [14].

With its transaction coordinator and the representative systems, Younas et al. use a *client-agent-server* architecture. Mobility or specific operation modes are not mentioned. The transaction is executed on the representative system in behalf of the *MH*. With *SACReD*, Younas et al. introduce a new set of transaction properties suitable to mobile services transactions.

J. A Priority Heuristic Policy in Mobile Distributed Real-Time Database System

In their paper "A Priority Heuristic Policy in Mobile Distributed Real-Time Database System", Singh et al. [26] cover scheduling of different running transactions in mobile distributed real-time database systems. They propose a priority heuristic approach to schedule transaction execution, especially addressing parallel execution of sub-transactions [26]. Their proposed heuristic concentrates mainly on the number of write locks of sub-transactions and total size of data items on which write locks are required [26]. Based on this, they schedule and prioritize transactions in mobile distributed real-time database systems [26].

Since the approach proposes a way to schedule and prioritize transactions in mobile systems and does not enforce a specific protocol or architecture, it cannot be classified by our categories.

K. MiCATS: Middleware for Context-Aware Transactional Services

Ettazi et al. [27] propose a model called Context-Aware Transactional Service Model (CATSM) for context-aware transactional services, enabling an implementation of context-aware services based on nested transaction models. A transaction service in CATSM is structured hierarchically, with a global transaction which can be decomposed into several sub-transactions [27]. The global transaction has a context

descriptor which contains the state of resources and conditions of the execution environment [27]. The context descriptor includes following sub-contexts: (i) transactional service, (ii) user (requirements, purpose, ...), (iii) device (operating system, battery level, ...), (iv) environment (location, time, ...), and (v) network contexts (connectivity, bandwith, ...) [27]. For failure resiliency, sub-transactions may refer to alternative transactions [27]. Their approach is split into four phases. In the first phase, the requirement of the user is captured. It is assumed that the user submits a request via a Graphical User Interface (GUI) provided by the context-aware Middleware for Context-Aware Transactional Service (MiCATS) - also introduced by Ettazi et al. within this work - which is installed on the user's device. A request contains functional requirements (e.g. criticality of tasks) and preferences of the user (e.g. recommendations), where the required degree of atomicity can be derived from [27]. Once the requirements are captured, the specification of transactional service rules is built in the second phase, which include the required degree of atomicity, the CATSM structure, i.e. the sub-transactions, their compensating and alternative transactions and the associated context descriptors [27]. The third phase mainly contains the selection of the services based on the transactional service rules. For each task, several services are selected, which can be used according to the context descriptor and transactional properties [27]. In the final phase, concrete services are chosen for each task, according to the current context, and eventually executed.

As the MiCATS is deployed onto MHs, a *full client* architecture is ensured. Due to associating alternative transactions with each sub-transaction, the proposed model can cope with the *disconnected* operation mode. Supported mobility modes are not mentioned explicitly. With MiCATS, the transaction execution is performed on *MHs*.

L. Context-adaptive and energy-efficient mobile transaction management in pervasive environments

Similar to Section IV-K, Tang et al. [28] introduce a context model, a context-aware transaction model and a context-adaptive and energy-efficient transaction management mechanism (CETM) in their work "Context-adaptive and energy-efficient mobile transaction management in pervasive environments". Their context model contains information about individuals, networks and devices, i.e.: (i) person (profile, requirements, ...), (ii) wireless network (connectivity, performance of networks), (iii) mobile device (computing and storage capacity), and (iv) location (longitude and latitude of mobile device) [28]. Using this context model, the proposed context-aware transaction model is able to adapt to changing transaction context. A transaction as proposed by Tang et al. is a 6-tupel, which contains: (i) a set of sub-transactions, (ii) a set of compensating transactions for these sub-transactions, (iii) a list with event, context and action for each sub-transaction, (iv) a set of relationships between sub-transactions, (v) a set of all sub-transactions' states, and (vi) a set of acceptable final states [28]. Finally, CETM coordinates a transaction, i.e. distributes its sub-transactions to neighboring MH or FH. Thereby, CETM selects an neighbor which is qualified in terms of having enough computing and storage capacity. If multiple neighbors are qualified to execute a sub-transaction, the closest is chosen, enabling a reduction of energy consumption [28].

Tang et al. propose a *flexible client-server* architecture, where a transaction can be executed *distributed between MHs and FHs*, but also entirely on a *MH* or a *FH*, depending on availability. *Micro* mobility is considered in their model to select the closest MH to execute a sub-transaction. They assume that a local database is reponsible to ensure ACID properties on local sub-transactions.

M. A cross-layer atomic commit protocol implementation for transaction processing in mobile ad-hoc networks

Obermeier et al. [29] introduce the Cross-Layer Atomic Commit Protocol (CLCP) in their paper "A cross-layer atomic commit protocol implementation for transaction processing in mobile ad-hoc networks". CLCP focuses on MANET and uses multiple coordinators to achieve a robust und failure resilient protocol [29]. It consists of following two phases: (i) decentralized commit phase, where participants vote and try to achieve a dezentralized decision on commit or abort, and (ii) termination phase, in the rare case the protocol cannot progress [29]. Participants exchange their votes and the votes of other participants in the decentralized commit phase via a commit matrix. Participants learn from each other's commit matrix, then merge and broadcast them [29]. A transaction is committed, if a participant P_i knows that for each participant P_j a majority of participants P_k know that participant P_j has voted for commit [29].

CLCP can be categorized as *MANET*. Additionally, the execution is *distributed between MHs* during the voting phase. Other specifics of the protocol such as mobility or operation modes are not mentioned explicitly. However, because of the similarities to FT-PPTC-Rec, an execution *distributed between MHs and FHs* can be assumed.

N. Blocking reduction for distributed transaction processing within MANETs

Besides their contribution described in Section IV-M, Obermeier et al. [30] propose an approach to reduce the amount of blocking within execution of mobile transactions in MANET. Blocking is usually caused by wrongly estimating the time-out of a transaction execution [30]. Obermeier et al. introduce therefore a distributed transaction model and a non-blocking "Adjourn-state", allowing processing of concurrent transactions while waiting for vote requests of coordinators of distributed transactions [30]. Contrary to a blocking, respectively waiting state, the "Adjourn-state" does not block concurrent transactions, nor require the setup of a transaction time-out [30]. This is achieved by recognizing and differentiating failures that: (i) require a complete transaction abort, (ii) require only a repitition of a sub-transaction, and (iii) allow reusing sub-transactions [30]. Using a commit tree, Obermeier

et al. keep track of the status of all sub-transactions and additionally coordinate the transaction [30].

Obermeier et al. assume a *flexible client-server* architecture for their approach. A *disconnected* operation mode is supported and, as an advantage, does not block concurrent transactions. They do not mention mobility explicitly. The execution of the mobile transaction itself is *distributed between MHs* since the approach focuses on *MANETs*

O. Agent-Based Infrastructure for Data and Transaction Management in Mobile Heterogeneous Environment

Ongtang et al. [31] propose Agent- based Transaction Management scheme for Mobile Multidatabase (AT3M) to allow fully distributed transaction management in mobile heterogeneous environments. At first, a global transaction is split into global sub-transactions, both global transactions and sub-transactions are represented by autonomous mobile agents [31]. Each mobile agent is able to decide locally on commit or abort, furthermore all mobile agents involved in a global transaction cooperate to eliminate global conflicts. To successfully execute a global transaction, the mobile agents create collaboratively a global schedule before executing sub-transactions locally [31]. AT3M enables a fully distributed transaction management, plus parallel processing of global sub-transactions [31].

Since the approach proposed by Ongtang et al. includes a mobile agent, the architecture is *client-agent-server*-based. AT3M supports *macro* mobility, as well as a *disconnected* operation mode. The execution of a mobile transaction can be *distributed between MHs*.

P. Connection Fault-Tolerant Model for distributed transaction processing in mobile computing environment

Dimovski et al. [32] consider two scenarios to introduce a distributed transaction processing model for mobile environments, i.e. one scenario where MH can connect to a FH, and one scenario where they cannot [32]. Their proposed connection fault-tolerant model is supposed to minimize aborts of mobile transactions caused by network disconnections and, in addition reduce the blocking time of FHs [32]. Furthermore, their model ensures atomicity [32]. To achieve this, Dimovski et al. assign a MH-agent to each MH, which represents the MH and serves as an intermediator between a MH and a transaction coordinator, which stores the state of the transaction execution [32]. The MH-agent stores all relevant data including the state of all mobile transactions in which the MH is involved [32]. If a MH, respectively their MH-agent is not able to connect to a FH, an attempt is made to establish an ad-hoc connection to any neighboring MH [32]. If no connection can be established during an execution of a transaction, neither through ad-hoc nor through a FH, the MH-agent sends fragments of the processing of the transaction to a decision algorithm, which checks if the fragment is writing or reading data. If the fragment is writing data, the decision algorithm stores the fragment in a First In, First Out (FIFO) queue and sends all fragments in the queue out, as soon as a connections is re-established [32]. If the fragment is reading data, the decision algorithm waits for re-establishment of a connection for a defined time frame. If a time-out occurs while waiting for the re-establishment of a connection, the MH-agent sends a "No" vote to the transaction coordinator [32].

As Dimovski et al. propose a concept with MH-agents, they follow a *client-agent-server* approach. This also enables working in *disconnected* operation mode and supports thereby handling of *macro* mobility. The execution itself can be *distributed between MHs*, as this approach supports *MANETs*.

V. Conclusion and Outlook

Within this paper, a systematic literature review on transactions in mobile systems was performed to gather the current state of research. As a basis, we used works from Serrano-Alvarado et al. [4], who had conducted an extensive survey of mobile transactions in 2009, which was described in Section II. We extended the categorization of mobile transactions introduced by Serrano-Alvarado et al. to include additional categories specifically related to technologies which were either introduced, or increased in relevance after 2009. Our performed systematic literature review was described in detail in Section III. In Section IV, the concepts of all relevant mobile transaction models and protocols that arose from our systematic literature review were briefly presented. Furthermore, a classification of each mobile transaction model, based on our extended categories, was performed.

It is likely that current trends like Industry 4.0 and IoT will shape the development of upcoming mobile transaction models further, increasing the popularity of MANET and Peer-to-Peer (P2P) approaches even more. With the crawling tool we developed and the precise description of the execution of our systematic literature review, it is possible to reproduce the whole process in the future, possibly with different keywords adapted to changes caused by emerging trends in mobile transactions.

References

[1] Statista. (2017). Number of smartphones sold to end users worldwide from 2007 to 2017 (in million units), [Online]. Available: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/.

[2] S. Vashi, J. Ram, J. Modi, S. Verma, and C. Prakash, "Internet of things (iot): A vision, architectural elements, and security issues," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 492–496. DOI: 10.1109/I-SMAC.2017.8058399.

[3] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 2014, ISSN: 1867-0202. DOI: 10.1007/s12599-014-0334-4. [Online]. Available: https://doi.org/10.1007/s12599-014-0334-4.

[4] P. Serrano-Alvarado, C. Roncancio, and M. Adiba, "A survey of mobile transactions," version 1-11, Sep. 2009. [Online]. Available: https://link.springer.com/content/pdf/10.1023%2FB%3ADAPD.0000028552.69032.f9.pdf.

[5] M. Younas, "Context-aware Mobile Services Transactions," pp. 705–712, 2010. DOI: 10.1109/AINA.2010.157.

[6] A. O. A. Salem and H Ahmad, "Classification of transaction models in mobile database system," *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, pp. 1–6, 2015. DOI: 10.1109/WSWAN.2015.7209086.

[7] L. Frank, R. U. Pedersen, C. H. Frank, and N. J. Larsson, "The cap theorem versus databases with relaxed acid properties," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '14, Siem Reap, Cambodia: ACM, 2014, 78:1–78:7, ISBN: 978-1-4503-2644-5. DOI: 10.1145/2557977.2557981. [Online]. Available: http://doi.acm.org/10.1145/2557977.2557981.

[8] R. A. Haraty, "A Comparative Study of Mobile Database Transaction Models," *2015 IEEE Symposium on Computers and Communication (ISCC)*, pp. 134–139, 2015. DOI: 10.1109/ISCC.2015.7405506.

[9] P. Serrano-Alvarado, C. Roncancio, and M. Adiba, "A survey of mobile transactions," *Distributed and Parallel Databases*, vol. 16, no. 2, pp. 193–230, 2004, ISSN: 1573-7578. DOI: 10.1023/B:DAPD.0000028552.69032.f9. [Online]. Available: https://doi.org/10.1023/B:DAPD.0000028552.69032.f9.

[10] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Comput. Surv.*, vol. 15, no. 4, pp. 287–317, 1983, ISSN: 0360-0300. DOI: 10.1145/289.291. [Online]. Available: http://doi.acm.org/10.1145/289.291.

[11] E. Brewer, *Towards Robust Towards Robust Distributed Systems*, 2000.

[12] M. Younas and S. K. Moste, "A new model for context-aware transactions in mobile services," pp. 821–831, 2011. DOI: 10.1007/s00779-011-0369-1.

[13] M. Younas and L. Barolli, "Failure Resilient Criteria for Mobile Web Services Transactions," *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, pp. 97–103, 2012. DOI: 10.1109/EIDWT.2012.54.

[14] M. Younas and I. Awan, "Mobility Management Scheme for Context-Aware Transactions in Pervasive and Mobile Cyberspace," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 3, pp. 1108–1115, 2013. DOI: 10.1109/TIE.2012.2198032.

[15] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, no. 2, pp. 51–59, Jun. 2002, ISSN: 0163-5700. DOI: 10.1145/564585.564601. [Online]. Available: http://doi.acm.org/10.1145/564585.564601.

[16] S. Ravimaran and M. A. M. Mohamed, "An improved kangaroo transaction model using surrogate objects for distributed mobile system," in *Proceedings of the 10th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, ser. MobiDE '11, Athens, Greece: ACM, 2011, pp. 42–49, ISBN: 978-1-4503-0656-0. DOI: 10.1145/1999309.1999319. [Online]. Available: http://doi.acm.org/10.1145/1999309.1999319.

[17] M. H. Dunham, A. Helal, and S. Balakrishnan, "A mobile transaction model that captures both the data and movement behavior," *Mobile Networks and Applications*, vol. 2, no. 2, pp. 149–162, 1997, ISSN: 1572-8153. DOI: 10.1023/A:1013672431080. [Online]. Available: https://doi.org/10.1023/A:1013672431080.

[18] S. Pathak, "A Lazy Commit Protocol for Mobile Transactions," *2014 IEEE International Advance Computing Conference (IACC)*, pp. 432–437, 2014. DOI: 10.1109/IAdCC.2014.6779363.

[19] B. Ayari, P. N. Suri, and P. P. Koopman, "Perturbation-Resilient Atomic Commit Protocols for Mobile Environments," 2010.

[20] B. Ayari, A. Khelil, and N. Suri, "On the design of perturbation-resilient atomic commit protocols for mobile transactions," *ACM Trans. Comput. Syst.*, vol. 29, no. 3, 7:1–7:36, Aug. 2011, ISSN: 0734-2071. DOI: 10.1145/2003690.2003691. [Online]. Available: http://doi.acm.org/10.1145/2003690.2003691.

[21] Z. T. Abdul-Mehdi, A. Bin Mamat, H. Ibrahim, and M. M. Deris, "A model for transaction management in mobile databases," *IEEE Potentials*, vol. 29, no. 3, pp. 32–39, 2010, ISSN: 0278-6648. DOI: 10.1109/MPOT.2010.936929.

[22] B. Ayari, A. Khelil, and N. Suri, "GMTC : A Generalized Commit Approach for Hybrid Mobile Environments," vol. 12, no. 12, pp. 2399–2411, 2013.

[23] T. Hara, T. Hara, S. Member, S. K. Madria, and S. Member, "Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks," 2009.

[24] Z. Xing and L. Gruenwald, "Managing concurrent execution of transactions in mobile ad-hoc network database systems: An energy-efficient approach," *Distributed and Parallel Databases*, vol. 31, no. 2, pp. 183–230, 2013, ISSN: 1573-7578. DOI: 10.1007/s10619-012-7114-2. [Online]. Available: https://doi.org/10.1007/s10619-012-7114-2.

[25] S. A. Moiz, "Improving the success rate of concurrent Mobile Transactions by predicting time for execution," pp. 7–10, 2014. DOI: 10.1109/UNESST.2014.13.

[26] P. K. Singh and U. Shanker, "A priority heuristic policy in mobile distributed real-time database system," in *Advances in Data and Information Sciences*, M. L.

Kolhe, M. C. Trivedi, S. Tiwari, and V. K. Singh, Eds., Singapore: Springer Singapore, 2018, pp. 211–221, ISBN: 978-981-10-8360-0.

[27] W. Ettazi, H. Hafiddi, M. Nassar, and S. Ebersold, "Mi-CATS : Middleware for Context-Aware Transactional Services," vol. 3, pp. 496–512, 2015. DOI: 10.1007/978-3-319-29133-8.

[28] F. Tang and M. Li, "Context-adaptive and energy-efficient mobile transaction management in pervasive environments," pp. 62–86, 2012. DOI: 10.1007/s11227-009-0277-6.

[29] S. Obermeier, S. Böttcher, and D. Kleine, "A cross-layer atomic commit protocol implementation for transaction processing in mobile ad-hoc networks," pp. 319–351, 2009. DOI: 10.1007/s10619-009-7051-x.

[30] S. Obermeier, S. Böttcher, M. Hett, and P. K. Chrysanthis, "Blocking reduction for distributed transaction processing within MANETs," pp. 165–192, 2009. DOI: 10.1007/s10619-009-7033-z.

[31] M. Ongtang and A. R. Hurson, "Agent-based Infrastructure for Data and Transaction Management in Mobile Heterogeneous Environment *," *2009 WRI International Conference on Communications and Mobile Computing*, vol. 3, pp. 314–318, 2009. DOI: 10.1109/CMC.2009.321.

[32] T. Dimovski and P. Mitrevski, "Connection Fault-Tolerant Model for Distributed Transaction Processing in Mobile Computing Environment," *Proceedings of the ITI 2011, 33rd International Conference on Information Technology Interfaces*, pp. 145–150, 2011.