Issues in Designing Concurrency Control Techniques for Mobile Ad-hoc Network Databases

Zhaowen Xing and Le Gruenwald
School of Computer Science
The University of Oklahoma, Norman, OK 73019, USA
{zhaowenxing, ggruenwald}@ou.edu

Abstract

With the rapid growth of wireless networking technology and mobile computing devices, a highly increasing demand arises for processing mobile transactions in Mobile Ad-hoc NETwork (MANET) databases. This allows mobile users to access and manipulate data anytime and anywhere. However, to guarantee timely access and correct results for concurrent mobile transactions, concurrency control (CC) techniques become critical. Due to the characteristics of MANET databases, existing mobile database CC techniques cannot work effectively. This paper discusses the issues that need to be addressed when designing a CC technique for MANET databases, and reviews related techniques with respect to those issues.

Keywords: Concurrency Control & Recovery, Distributed Databases, Mobile Databases, Mobile Transactions, Mobile Ad-hoc Networks

1. Introduction

A Mobile Ad-hoc NETwork (MANET) is a collection of self-organized mobile nodes connected by relatively low bandwidth wireless links. Each node has an area of influence called cell, only within which others can receive its transmissions [10]. Due to no fixed infrastructures, all nodes are free to roam, the network topology may change rapidly and unpredictably over time, and nodes automatically form their own cooperative infrastructures. Each node operates as an autonomous end system and a router for others in the network. Thus, nodes can communicate directly with each other either through one hop or multiple hops. In a distributed client-server MANET database system, not only clients but also servers are mobile, wireless and battery-powered.

MANET database systems are of interest because there is no prior investment for fixed infrastructures, it can be easily deployed in a short time, and end users can access and manipulate data anytime and anywhere. Examples of MANET database applications [7] include law enforcement operations, automated battlefield applications, natural disaster recovery situations where the communication infrastructures have been destroyed, self-organizing sensor networks for data collecting, and interactive lectures or conferences for data exchange without pre-installed infrastructures. However, the flexibility and convenience in MANETs raise new issues when performing concurrency control (CC). CC is the activity of preventing transactions from destroying the consistency of databases while allowing them to run concurrently, so that the throughput and resource utilization of database systems are improved and waiting time of concurrent transactions is reduced [22].

The following inherent characteristics of MANET databases have to be taken into considerations when designing a CC algorithm.

- **Mobility**: The ability to change nodes' locations while retaining network connection is the key motivation of MANET computing. However, a node's network and physical location change dynamically, and at the same time, the states of transactions and accessed data items have to move along with the node. These changes affect the system configuration parameters of the node and those of other nodes like the routing information in the routing tables and affect those transactions executed on the node. Mobility can also lead to multi-hop communication, high communication latency, frequent disconnections and long-lived transactions.
- **Low bandwidth**: Wireless network bandwidth is much lower than its wired counterpart. For example, the widely used 802.11b wireless card has a maximum data rate of 11 Mbit/s [12] and the latest 802.11g wireless card supports data transmission up to 54 Mbit/s [1]; however, currently an affordable Gigabit Ethernet card realizes a maximum data rate of 1000 Mbit/s [24]. Thus, within the cell of one node, inside neighbors have to share and compete for the same channel. If someone fails, it may keep sending requests until timeout. This low bandwidth can result in communication delays, a high risk of disconnections and long-lived transactions as well.
- **Multi-hop communication**: In traditional mobile networks, mobile nodes that communicate with each other must go through a fixed infrastructure. While in MANETs, nodes can communicate with each other either directly or via other nodes that function as routers. When communication goes more hops, more power

and bandwidth are consumed, and more execution time is needed to complete transactions.

- **Limited battery power**: Because of the mobility and portability, clients and servers have severe resource constraints in terms of capacity of battery and sizes of memory and hard drive. In addition, the battery technology is not developed as rapidly as the mobile devices and wireless technologies. For instance, a fully-charged Dell Latitude C600 laptop can run about 3.5 hours, which is estimated by well-known industry battery life benchmarks [23]. When processing power is limited, it compromises the ability of each mobile node to support services and applications [7]. Once a node runs out of power or has insufficient power to function, communication fails, disconnections happen, execution of transactions is prolonged, and some transactions may have to be aborted.
- **Limited storage**: Due to mobility and portability, the sizes of memory and hard drive are smaller than the ones in the wired network. The consequences of this are less stored/cached/replicated data, fewer installed applications, and more communication.
- **Frequent disconnections**: A node is disconnected when it roams freely and is out of the transmission range of all its neighbors, it fails to compete for the channels of popular neighbors, its battery runs out, or it runs into some failures. It is normal for a node to become disconnected in MANETs, and such disconnections do not always imply the failure of transactions initiated or executed by the disconnected node because the disconnected node may reconnect after some time. While disconnected, the disconnected node may still be able to process part of already

started transactions. However, when disconnections happen frequently, more transactions may be delayed or blocked, and even aborted if they are real-time and missed their deadlines.

- **Long-lived transactions**: Due to mobility, wireless communication delay, less processing power, frequent disconnections and unbounded disconnection time, transactions in MANETs databases tend to be long-lived. When the execution is prolonged, the probability of conflicts with other transactions becomes higher and, consequently, transactions are likely blocked if a pessimistic CC method applies or restarted if an optimistic CC method is in use.

MANET database CC research is still in the early stage. To the best of our knowledge, so far only one MANET CC algorithm has been proposed [6]; however, it does not take into account all the identified characteristics of MANET databases. The existing CC techniques for traditional mobile network databases, in which only clients are mobile and battery-powered, cannot be directly applied because of these characteristics either. In this paper, we identify the design issues inherent to CC techniques for client-server MANET databases, review how current existing mobile database CC techniques handle these issues and address MANET characteristics at the same time.

The rest of this paper is organized as follows. Section 2 discusses the general issues that need to be addressed in the design of CC algorithms for MANET databases and reviews the existing mobile CC techniques with respect to these issues. Section 3 presents the application-dependent issues for MANET database CC algorithm design and reviews the related work. Section 4 summarizes the techniques reviewed in Sections 2 and 3. Finally, Section 5 concludes the paper.

2. General Issues for MANET Database Concurrency Control Algorithm Design

General issues are those that every CC algorithm for MANET databases needs to address.

2.1 Types of Concurrency Control Algorithms

To guarantee the correct results and consistency of databases, the conflicts between transactions can be either avoided, or detected and then resolved. Most of the existing mobile database CC techniques use the (conflict) serializability as the correctness criterion. They are either pessimistic if they avoid conflicts at the beginning of transactions, or optimistic if they detect and resolve conflicts right before the commit time, or hybrid if they are mixed. To fulfill this goal, locking, timestamp ordering (TO) and serialization graph testing can be used as either a pessimistic or optimistic algorithm.

2.1.1 Pessimistic CC Methods

In a pessimistic CC method, many transactions conflict with each other, concurrent transactions are serialized at the beginning of their execution, and conflicting transactions are blocked, restarted or aborted. A lot of recent CC research in mobile databases has used this method and applied a locking scheme to produce serializability. However, this method is not suitable for high volumes of transactions, and it is an unnecessary overhead when transactions are read-only. Also because of this early prevention, available limited system resources cannot be fully utilized. As the transaction execution is prolonged and the disconnection time is unbounded in MANET databases, the possibility of conflicts among concurrent transactions increases as well. When there are more conflicts, more transactions will be blocked, restarted, or directly aborted.

The Kangaroo Transaction (KT) model [9] is the first model that captures both the data and movement behavior in a traditional mobile distributed environment. However, there is no isolation rule for global transactions. The isolation of local transactions is guaranteed by the locking scheme. A KT, which is initiated by a mobile node, consists of a set of Joey Transaction (JT). Part of KT status information moves to the new Mobile Support Station (MSS) and a handoff record is written into the old MSS log when a mobile node migrates from one cell to another. At the new MSS, a new JT as part of the handoff process is subsequently created. Under the compensating mode, a previously committed JT can be rolled back and a portion of a JT can commit early to release valuable resources. However, in MANETs, no one has unlimited wired power and higher bandwidth like a MSS does, thus, how to elect an appropriate replacement for the MSS without overloading it would be a problem. Also some of MANET applications may not have the compensable transactions, thus, it is impossible for sub-transactions to commit early. Because many of MANET applications are mission-critical, they require consistency and accuracy. For instance, to query the location of enemies before firing a missile, the accurate data has to be returned.

Distributed High Priority 2-Phase Locking (DHP-2PL) [15] is proposed for traditional mobile distributed real-time database system. DHP-2PL extends the well-known High Priority 2PL, which solves conflicts in favor of the higher priority transactions, but, a high-priority transaction can be restarted by a conflicting low-priority one to reduce the unnecessary abort when it is suspected in a disconnection state. Lam et al [15] overcome the wireless communication limitation by using the shipping approach, which ships the entire transaction to the wired base station for processing, and propose similarity-based DHP-2PL to reduce the number of restart transactions and blocking time, where the similarity means the values of a data item are slightly different and interchangeable. Unfortunately, in MANETs, because of disconnections, locked data may be unavailable for unbounded time. Also, since no node has unlimited wired power like a wired base station, the shipping approach would drain some nodes' power quickly.

To reduce the impacts of unpredictable and unreliable mobile networks, a Multi-Version (MV) data model is presented in [16], and the relative consistency is adopted as the correctness criterion for the processing of real-time transactions in a traditional mobile distributed environment. In the MV model, each data item consists of a sequence of versions, which are defined by validity intervals. The associated validity interval indicates at which time the version is valid. Given a transaction, accessed data items are relatively consistent if they are temporally correlated with each other and valid at the same time point other than the commit time. An image transaction model, in which fixed nodes (like proxies) pre-fetch required data from other fixed hosts and process mobile transactions for mobile nodes, is proposed to reduce the data access delay, save wireless bandwidth, and minimize restart overhead. Each data item stored in mobile nodes is also replicated at a fixed node in case of disconnections. Unfortunately, there are no fixed nodes in MANETs, thus the image transaction model would overload some nodes, and it is a difficult task to guarantee the consistency of replicated data when disconnections are common. Also, there are more overheads to maintain multiple versions when more transactions run concurrently

[17]. The MV method is not scalable because of the limited storage in term of memory and hard drive in MANET databases.

In [2], the Broadcast and Updated Cycles (BUC) technique is proposed for a traditional mobile data broadcast environment. Broadcast data are grouped into cycles. BUC uses the cycle number to check the conflict between the cached data in the read set of clients and data at the wired Information Server (IS), which provides the data and validates all write transactions. While at IS, the 2-Phase Locking (2PL) is running to verify the serializability. Wired broadcast servers use their abundant bandwidth to broadcast the validation information and data periodically, and mobile clients fetch the required data according to the broadcast indices during a part of the broadcast cycle. Since the data indices are broadcast at the start of each cycle, mobile clients can disconnect for a while and even switch to the doze mode to save battery power. However, in MANETs, when the broadcast servers and centralized IS get disconnected, mobile clients cannot fetch any data to process transactions. Because of limited battery power, broadcasting data and invalidation information periodically may drain broadcast servers' batteries very fast, and consequently, the support of disconnection and power does not work any more.

Based on the 2PL and Summary Schema Model (SSM), a semantic hierarchical concurrency control algorithm, V-Lock, is proposed in [18] for a traditional mobile heterogeneous database system. V-Lock uses semantic information in the summary schema to maintain global locking tables, which are used to guarantee the consistency of global transactions, detect and remove global deadlocks. The local sites use the strict 2PL for correctness. Because of the limitation of bandwidth, V-Lock can dynamically adjust the frequency of the communication between the global transaction manager and the local sites. However, the consequence is that less communication results in more false aborts. To reduce wireless communication and deal with disconnections, read-only transactions and their associated data are cached at mobile nodes. Unfortunately, global deadlocks are detected by collecting local wait-for-graphs to build global wait-for-graphs and checking for cycles; thus, more communication and power are needed. In addition, how to keep detecting global locks is a problem when some local wait-for-graphs are not available due to disconnections.

In order to conserve the limited resources and avoid intensive messages, a weak form of Concurrency Control Mechanism (CCM) is proposed in [20] for the centralized replicated database, which stores only data items that have numerical values. CCM is based on the epsilon-serializability, which tolerates a limited amount of inconsistency and can be preserved by divergence control and consistency restoration. CCM is different from others because locking is not applied to produce serializability. That is, transactions are executed serially and are free to commit at mobile clients if the updated value and total change of replica are not more than the maximum change allowed, which is calculated by a predefined function and valid within a predefined timeout; otherwise, transactions are blocked until timeout or submitted to the centralized server for processing. The centralized server periodically broadcasts all data and associated maximum change allowed, updates the master copy of data by the arrival order of transactions, and sends acknowledgements of commit or abort to clients. If mobile clients disconnect for a period longer than the specified timeout, the cached data have to be re-read and pre-commit transactions have to be re-executed. Since some transactions are free to commit at mobile clients, the abort rate and transaction execution time are reduced. However, CCM cannot work effectively with

MANET databases because the centralized database server is not available during disconnections and not all MANET applications contain only numerical data items. Due to the limited battery power and storage in MANETs, broadcasting all data periodically and caching them locally are not feasible either.

Semantic Serializability Applied to Mobility (SESAMO) [6] is proposed for MANET databases. SESAMO is based on the semantic serializability, which assumes that databases disjoint, updates on a database only depend on values of data of the same database, and the global serializability is automatically guaranteed if each site maintains its own serializability by applying strict 2PL. The limited bandwidth is saved and transaction execution time is reduced because global serializability is automatically guaranteed without coordination among servers, and the locks held by sub-transactions of a global transaction can be released once they finish at local sites. However, SESAMO may fail in MANET applications, in which global transactions do conflict with each other because it assumes "*Any two given mobile multi-database transaction schedulers do not schedule any transaction in common*". Due to no coordination among servers, the database consistency is not preserved when some sub-transactions cannot commit along with others due to disconnections.

2.1.2 Optimistic CC Methods

In contrast, an optimistic CC method assumes that not too many transactions conflict with each other, and thus allows transactions to be executed simultaneously, and delays the serializability check of these transactions until the commit time. This delay provides CC mangers more information to determine the fate of committing transactions.

However, the tradeoff is the overhead of late restart and waste of limited resources. This tradeoff becomes worse when the probability of conflicts among concurrently executing transactions is high due to the prolonged execution of transactions and unbounded disconnection time in MANET databases.

Partial Validation with Timestamp Ordering (PVTO) [17] is proposed to process transactions in a mobile centralized broadcast environment, in which the static database server uses its abundant bandwidth to broadcast the data and validation information periodically, so that mobile clients can fetch required data during a part of the broadcast cycle and validate executed transactions partially. In PVTO, every transaction is associated with a timestamp interval, which is dynamically adjusted when the transaction reads some data or after another transaction successfully commits at the static server. If the interval shuts out, then the associated transaction has to be aborted because a non-serializable schedule is detected. In case of a disconnection, the upper the bound of timestamp interval is set to the time of disconnection or the current upper bound, whichever is smaller, to avoid shutting out. However, in MANETs, when the centralized database server disconnects, mobile clients cannot receive data and validation information to process and partially validate transactions, and partially validated transactions cannot be finalized at the server; because of the limited power and bandwidth, no node can play the role as the centralized static database server.

2.1.3 Hybrid CC Methods

A hybrid method is a combination of optimistic and pessimistic CC methods. For instance, an optimistic method may be used to validate global transactions and pessimistic one is applied to verify local transactions in a distributed database system. Thus, all the problems existing in both methods have to be addressed.

Partial Global Serialization Graph (PGSG) [8] is introduced to enforce a range of consistency and isolation in the traditional mobile multi-databases environment. In PGSG, before committing a global transaction, a partial global serialization graph is built to check for cycles. The local sites serialize transactions by applying the ticket method proposed in [11]. A global data structure moves along with a mobile node when it migrates from one cell to another. When a mobile node disconnects, its status is marked as disconnected, and its mobile support station (MSS) saves all the responses in a structure called ResponseList, and delivers them to this mobile node upon reconnection. If a catastrophic failure occurs during the disconnection, then the status is changed to suspended. In order to minimize erroneous aborts, suspended transactions are not aborted until they obstruct other executing transactions. In order to release resources in a timely manner and tolerate long-lived transactions, compensable sub-transactions can commit before the global ones commit. Unfortunately, in MANETs, since everyone is a self-organized node, during disconnections, electing which neighboring node to backup the information will be a challenge. The strategy of supporting mobility and long-lived transactions does not work either because PGSG has the same problems as the KT model reviewed in Section 2.1.1. In case of disconnections, some predecessor graphs may not be collected, and then the PGSG algorithm has to stop and wait.

In [13], to speed up transaction processing and to reduce wireless communication, mobile clients execute transactions against the local cache and use strict 2PL to serialize transactions. To ensure that cached data are up-to-date, an invalidation report is periodically broadcast by the centralized static database server. Before transactions commit, the commit request must be validated at the centralized database server by applying one of the three algorithms: Mobile Transaction Commit using Serialization Graph (MTC-SG), SeQuential order (MTC-SQ) or MTC-Hybrid. MTC-SG maintains only committed transactions, and builds the serialization graph to guarantee no cycle is involved. In MTC-SQ, every validating transaction is checked if it can be inserted somewhere into the maintained sequential order of committed transactions, and the final order must comply with the serialization order. In MTC-Hybrid, MTC-SQ is applied first, and if it fails, then MTC-SG is employed. When mobile clients migrate from one cell to another, their last invalidation report must be checked to ensure that they receive the latest report. Unfortunately, in MANETs, because of the limited power and bandwidth, no node can process all commit requests and, at the same time keep broadcasting periodically like the static database server. In MTC-SG/Hybrid, building the serialization graph and checking for cycles require more time and power to accomplish. When the centralized database server disconnects, not only mobile clients cannot receive broadcast data and invalidation report, but also commit requests cannot be processed at the server.

2.2 Rules of Producing Serializability

Serializability requires that the effects of executing a set of transactions concurrently be equivalent to the effects of executing the same set transactions in some serial order [4]. There are three general rules to produce serializability: locking, timestamp ordering and serialization graph testing.

2.2.1 Locking

When two transactions access the same data items, one transaction must wait until the other releases the data items common to both. This can be done by locking and unlocking the data items. The well-known protocol 2-phase locking (2PL) requires that all lock operations (growing phase) precede all unlock operations (shrinking phase) in any transaction [5].

To improve transaction throughput and response time and to minimize the abort rate, the inherent deadlock and starvation problems must be resolved in a locking scheme. Deadlock occurs when two transactions try to lock several common data items simultaneously, and both wait for the other to release the locks that they have successfully requested. Starvation happens when some transaction keeps failing to obtain the requested locks. Due to long-lived transactions and the unbounded disconnection time in MANET databases, locked data items can be unavailable for a long time [3], and thus the deadlock and starvation problems become worse than in traditional mobile databases.

The KT model [9] applies some locking scheme, and allows locks to be released early, so that valuable resources can be reused in a timely manner. DHP-2PL [15] and BUC [2] adopt the 2PL to serialize transactions, but DHP-2PL resolves lock conflicts between non-committing transactions by using the transaction restart mechanism other than putting transactions into the waiting list. MTC-SG/SQ [13], V-Lock [18] and SESAMO [6] apply the strict 2PL to guarantee the consistency at local sites, where all locks are released only after the transaction commits, so that the cascading abort problem can be avoided.

2.2.2 Timestamp Ordering

Each transaction is assigned a globally unique timestamp by the system, the timestamp is attached to its all operations, and conflict operations of two transactions are processed in the assigned timestamp order (also called serialization order). The timestamp may be assigned in the beginning, middle or end of the execution of transactions. To achieve the uniqueness of timestamps in a distributed database system, all clocks at all sites have to be synchronized; otherwise, two identical timestamps must be resolved [5]. In MANETs, consuming the limited resources makes unnecessary aborts more expensive, and the synchronization task is harder to achieve because of the frequent disconnections.

In PGSG [8], at each site, each sub-transaction of a global transaction must read and increment the ticket, and the ticket number determines the serialization order between two sub-transactions, that is, the one with a lower number must precede the one with a higher number. However, these read and write ticket operations may result in a forced conflict between two global transactions that do not conflict with each other originally, and consequently, one of them is aborted unnecessarily and the abort wastes the limited resources in MANETs. In MTC-SG/SQ [13], a sequential order of committed transactions is maintained. When a validating transaction *T* arrives, the timestamp is assigned and *T* is checked if it can be inserted somewhere into the sequential order, and the final order must comply with the serialization order. If *T* can, it is committed. In PVTO [17], a timestamp interval is assigned to each active transaction *T* with the initial value $[0,\infty)$, and the read timestamp and write timestamp of each data item are maintained. *TI(T)* reflects the dependencies of *T* on the committed transactions, and it is dynamically adjusted. If *TI(T)* becomes an empty set after the adjustments, *T* has to be aborted because a non-serializable schedule is detected. Because of the mobility and limited storage in MANETs, in order to avoid overflowing the storage and reduce communication overhead, the list of committed transactions and timestamp intervals of committed transactions have to be trimmed periodically.

2.2.3 Serialization Graph Testing

A serialization graph is built explicitly to check for cycles when a transaction get started or is ready for validation, where the graph is a directed graph whose nodes represent transactions and whose edges represent the conflict relationships among them. If a cycle is detected, this transaction has to be restarted or aborted to keep the graph acyclic. The minimization of the serialization graph information and communication cost has to be addressed due to the limited resources in MANET databases.

In PGSG [8], when a global transaction *T* is ready for verification of atomicity and isolation, the *T*'s coordinator collects the predecessor graphs from all participating sites of *T*, and builds a partial global serialization graph to check for cycles. If there is no cycle or a cycle is detected and it can broken by aborting some suspended transaction, then *T* passes the isolation verification; otherwise *T* has to be aborted. In [13], MTC-SG maintains only committed transactions in the serialization graph, and the serialization order between two conflicting transactions is determined by the types of conflicting operations (read-write or write-write) and their execution locations. When a transaction *T* is ready for commit at the centralized server, MTC-SG adds *T* to the maintained graph and checks for cycles. If a cycle is detected, the commit request is rejected and all edges induced by *T* are removed; otherwise *T* can go ahead to commit. Since every node is mobile and frequent disconnections are common in MANETs, how to construct the serialization graph during disconnection is a challenge.

2.3 Concurrency Control Granularity

The granularity of a data item is the size of the data contained in the data item [4]. The CC granularity, which is the size of data items used to prevent/detect transaction conflicts, can be a database row, a page, a table or a database. If a typical transaction accesses a small number of rows, then it is advantageous to have row granularity for higher concurrency and fewer aborts. If a transaction typically accesses many rows of the same table, then it is better to have table granularity so that the resources, which are required to prevent conflicts in a pessimistic method or detect conflicts in an optimistic method, can be saved. In other words, higher concurrency and fewer aborts but more resources are required for fine granularity. In contrast, lower concurrency and more aborts but fewer resources are required for coarse granularity; however, more aborts consequently waste the limited resources in MANETs. All reviewed techniques utilize the term “data items” but they do not specify it.

2.4 Mobile System Architecture

A mobile system architecture can be classified as either a traditional mobile network (or cellular mobile network) architecture or a MANET architecture. A traditional mobile network architecture consists of fixed nodes and mobile nodes, where only mobile nodes are mobile and battery-powered. Mobile nodes retain their network connections through a wireless interface supported by some fixed nodes known as mobile support station (MSS) [21]. But in a MANET architecture, every node is mobile, wireless and battery-powered, and can communicate each other directly either through one hop or multiple hops. All reviewed techniques except SESAMO [6] are developed for traditional mobile network architectures, in which MSSs either periodically broadcast data items and validation information for mobile nodes or process transactions on behalf of mobile nodes by using their unlimited power and high bandwidth. SESAMO is proposed for MANETs, but it only addresses two out of seven MANET database characteristics: low bandwidth and long-lived transactions.

2.5 Location of Concurrency Control Manager(s)

A CC manager (or scheduler) is the heart of a CC algorithm because every data request or final validation must go through it. Depending on the architecture of a database system (centralized or distributed) and the design of a CC algorithm, a CC manager can be either centralized or distributed as well. For instance, in a distributed database system, if there is only one CC manager that is located at one site to schedule all transactions, then this CC manager is centralized. However, besides the bottleneck problem, a centralized CC manager may not always be available due to frequent disconnections. Also the computation overhead gets worse because of limited battery power, memory and disk space. To resolve these problems and process transactions in a timely manner, there should be more than one CC manager located at different sites, where each CC manager has autonomous processing capability on local transactions, and may coordinate with each other to execute global transactions. But, the tradeoff of distributed CC managers is the communication overhead in terms of handshaking among them for the cooperation or broadcasting data and invalidation information.

CC managers in all the techniques reviewed in this paper are distributed, but some of them can process only local transactions with cached/replicated data such as MTC-SG/SQ [13], BUC [2], PVTO [17] and CCM [20]. In these techniques, static primary-copy database servers periodically broadcast the data and invalidation information; while mobile nodes cache/replicate necessary data, process transactions locally and do some possible early validation. But, finally the primary-copy database servers still need to validate uncertain transactions with latest data information. Although a caching/replication technique can improve data access time and data availability, the primary-copy database server still has the same problems as a centralized CC manager. Also, because of the MANET database characteristics, no one can replace the role of the static primary-copy database server, and it is impossible for any node to keep broadcasting data periodically when the size of a database is large.

2.6 Improving System Performance

In order to improve transaction throughput and response time and to effectively utilize system resources, it is natural to allow multiple concurrent transactions to be executed simultaneously. However, when a transaction does not complete its execution successfully because of failure or inconsistency, it is aborted or restarted, and consequently, the time and system resources are wasted. It is expensive to abort or restart transactions in MANET databases because this would consume the limited bandwidth, battery power and storage. Therefore, a good CC algorithm for MANET databases should offer high concurrency and avoid any unnecessary aborts, so that more transactions will have chances to complete and commit in a timely manner and nodes will not waste its scarce resources, especially power, which may subsequently cause disconnection. In the following we discuss how the reviewed CC techniques improve system performance by either improving transaction response time or reducing transaction abort rate.

To improve the response time: In KT [9], the isolation property of global transactions is not preserved, thus, no global transactions are blocked/aborted at the global level. In DHP-2PL [15], to improve concurrency, all similar data can be used interchangeably in a schedule without adverse results, where a data item is similar if its values are slightly different and are interchangeable as read data for transactions. In [2], BUC allows mobile clients to update data locally. In V-Lock [18], the read-only transactions and their associated data are cached to lessen the effects of weak communications and disconnections. In CCM [20], mobile clients have the ability to commit transactions locally in a timely manner instead of waiting for the centralized database to do all the work. In SESAMO [6], the transaction execution time is reduced because the semantic serializability is applied and the global serializability is automatically guaranteed.

To reduce the abort rate: In PGSG [8], in order to minimize erroneous aborts, suspended transactions are not aborted until they obstruct other executing transactions. In MTC-SG/SQ [13], the sequential order of committed transactions is adjusted to allow more transactions to pass the commit request. In MV [16], the authors use multiple versions (each data item consists of a sequence of versions), relative consistency (data are temporally correlated with each other and valid at the same time point other than the commit time) and an image transaction model (fixed nodes process mobile transactions for mobile nodes), so that read transactions are never rejected and update transactions have more chances to commit. In [17], since PVTO dynamically adjusts the timestamp interval (a transaction has to abort if its interval becomes empty) of transactions, more transactions may commit.

2.7 Cascading Rollback

When a transaction aborts, the recovery scheme must restore the database to the consistent state that existed before the transaction started. It is necessary to ensure that any transaction that has read data written by the aborted transaction is also aborted. This phenomenon is called cascading rollback. Cascading rollback usually can be avoided by not allowing transactions to read un-committed data items in the CC design; otherwise the consistency property is not preserved and it also results in a significant amount of undo work, which is expensive in MANET databases because of wasting low bandwidth, limited battery power and storage.

In KT [9] and PGSG [8], because compensable sub-transactions can commit before global transactions commit, cascading rollback can still exist when some committed sub-transactions are rolled back by executing corresponding compensable transactions. This makes them infeasible to MANET applications because many of them are mission critical and they have no compensable transactions. All other existing techniques reviewed in this paper implicitly resolve this issue because they do not allow transactions to read un-committed data items.

2.8 Insert and Delete Operations

Besides read and write (update) operations, an insert operation inserts a new data item with an initial value into the database; and a delete operation deletes a data item from the database. Read, write, insert and delete can be conflict operations and result in the phantom phenomena or logic errors when any two of them execute in different orders. For instance, a logic error will occur when a data item is read after it is deleted or before it is inserted. To avoid logic errors, insert and delete operations have to be treated like write operations; otherwise, corresponding transactions have to be aborted or restarted, and aborts are expensive in MANET because they waste limited system resources.

Only in DHP-2PL [15], insert operations are identified as write operations, but delete operations are left out. All other existing techniques reviewed in this paper assume that transactions contain only read and write operations. Although insert and delete operations are not applied as frequently as read and write operations, they are still necessary for database management and need solutions in case of failures.

2.9 Level of Consistency

In some applications (e.g. statistical analysis or traffic information), in order to increase the degree of concurrency and reduce abort rate, it is acceptable to relax the transaction consistency requirement by reading stale data; however, this loose consistency may compromise the accuracy of the database.

In [9], the KT model completely relaxes the consistency and isolation level of global transactions, and the authors believe "*Returning 'dirty' data tagged with appropriate warnings is much more useful than returning an ABORT message*." In PGSG

[8], transactions are classified as either vital or non-vital, and the isolation property is not forced on non-vital transactions. All other existing techniques reviewed in this paper require strict consistency instead. However, since many MANET applications are mission critical and require strict consistency of the database, KT cannot be directly applied.

2.10 Transaction Model

Depending on applications, a transaction can be flat or nested [19]. The flat model is simpler to implement, but rolling back the entire transaction and starting from the very beginning is the only option when some part of the transaction fails. This would definitely waste lot of the limited resources of MANET. While in a nested model, a transaction can be dynamically decomposed into a hierarchy of sub-transactions (child transactions), and this decomposition grants the opportunity to rollback/restart only the failed sub-transaction rather than the entire transaction. This of course reduces the amount of completed work that is wasted by a flat model, but the tradeoff is that a nested model complicates the ACID properties in MANETs. For instance, some sub-transactions committed early to improve concurrency, but their parent transactions cannot complete and rollback them due to frequents disconnections. Then how to guarantee the atomicity and consistency has to be addressed.

In [9], based on an open nested and split transaction model, the KT model is proposed and associated with two processing modes compensating mode and split mode, which determine whether committed transactions can be compensated or not. In PGSG [8], a global transaction can be decomposed into a set of compensable sub-transactions, which can be rolled back even after they already committed by executing corresponding compensable transactions. However, when some MANET applications do not provide compensable transactions, KT and PGSG lose the benefits of compensable transactions. All other reviewed techniques adopt the flat model for simplicity.

3. Application Dependent Issues for MANET Database Concurrency Control Algorithm Design

Not all MANET applications may have the same requirements for CC algorithm design. In this section, we discuss the issues that are application-dependent.

3.1 Global serializability

The consistency of a global transaction may not be guaranteed although its sub-transactions are serializable at each local site. Thus, in a distributed client-server database system, serializability has to be maintained at not only the local level but also the global level. To achieve global serializability, due to MANET characteristics, it is crucial to address how to require less communication, utilize battery power efficiently and allow nodes to effectively cooperate with each other.

In KT [9], the isolation property of global transactions is not preserved at all. In PGSG [8], a partial global serialization graph is constructed by the global transaction coordinator to check for cycles. In [15], DHP-2PL adopts distributed 2PL to serialize global transactions. In MV [16], to guarantee the consistency of global transactions, the authors use multiple versions (each data item has a sequence of versions), relative consistency (data are temporally correlated and valid at the same time point other than the commit time) and an image transaction model (a fixed host works as a proxy for mobile hosts). In [18], the V-Lock model uses the global locking table to verify global transactions, detect and remove global deadlocks. In SESAMO [6], based on the semantic serializability, the global serializability is automatically preserved when all local sites guarantee local serializability. Since taking care of the global serializability is part of a CC algorithm if applicable, whether these techniques handling global serializability are good or bad for MANET databases is already discussed in Section 2.1—Types of CC algorithms. All other reviewed techniques process only local transactions with cached/replicated data, thus, they do not need to address this issue.

3.2 Degree of Local Autonomy

In a heterogeneous database (or multi-database) system, each local database system has the right to share internal data or not, and choose its own mechanisms for data and transaction management. This is called local autonomy. Local autonomy is preserved if the local site is not modified to coordinate global serializability. When the local site has more autonomy, it can effectively utilize the local system resources. In MANETs, it is important to effectively utilize limited resources in term of wireless bandwidth, battery, memory, hard disks and so on.

In KT [9], the local system has the complete autonomy because the global consistency is not maintained. In PGSG [8], local autonomy is preserved by adopting semantic atomicity and compensable transactions, that is, all sub-transactions are committed or each sub-transaction is aborted/compensated. In V-Lock [18], the local autonomy is preserved as much as possible except constraining local systems not to abort any sub-transactions that already response YES votes to the “Prepare to Commit” requests from the 2-phase commit protocol. In SESAMO [6], local autonomy is fully preserved because authors assume distributed databases disjoin, so that the global serializability is automatically preserved when all local sites guarantee local serializability. All other reviewed techniques are proposed for distributed homogeneous database systems, thus, they do not need to address this issue.

3.3 Cached/Replicated Data

In order to improve data access time and availability, caching/replication is a process that creates several duplications of the same data and stores one copy at a different site [14]. Caching slightly differs from replication in that cached data is only available to the site where the data is cached. However, in MANETs, since every node is battery-powered, to save and balance the power consumption of primary copy servers must be addressed; otherwise, no services would be provided when those servers run out of power. Servers with frequent disconnections (due to mobility or power failure) should not be considered as primary copy servers or caching/replication servers because it is difficult to maintain the cached/replicated data consistency and guarantee the availability.

Among the reviewed techniques, MTC-SG/SQ [13], BUC [2], PVTO [17] and CCM [20] apply caching solutions, where static primary copy servers broadcast data and validation information periodically and mobile clients cache necessary data. However, because of limited bandwidth, battery power and storage in MANETs, it is impossible for some node to keep broadcasting like a static server without draining its power fast.

Without the help of broadcast, the caching/replication techniques of these four algorithms do not work effectively any more.

In MV [16], each data item stored in mobile nodes is also replicated at a fixed node in case of disconnections. In V-Lock [18], to reduce the access latency due to disconnections, read-only transactions and their associated data are cached, and the parity-based signature is applied to invalidate cached data at the static primary copy servers. However, in MANETs every node is mobile and battery-powered, so how to elect nodes to replace the fixed nodes is an issue.

3.4 Real-Time Applications

A real-time transaction is defined as a transaction that must be executed within its deadline. In some cases, data items associated with real-time transactions have temporal constraints, that is, they remain valid only within a certain time interval. These data are called temporal data. Due to frequent disconnections, unbounded disconnection time and long-lived transactions in MANETs, how to meet transaction deadlines (and how to process data within their valid time intervals if temporal data exist) is critical for real-time applications in addition to guaranteeing the consistency of the database.

DHP-2PL [15] applies higher priority preference and transaction restart to meet transaction deadlines, where the priority of a transaction is based on its deadline and criticality. But, a high priority transaction can be restarted by a low priority one if it is in a disconnected state. Since in MANETs disconnections are frequent due to mobility and limited power, consequently frequent restarts are possible in DHP-2PL. In MV [16], the earliest deadline first is applied to assign priorities to transactions, and stale data items (one version of multiple versions) are acceptable to process only soft transactions, which are associated with soft deadlines. However, when the size of a database is large, MV is not scalable because of limited storage in MANETs. Also, MV cannot be applied in mission-critical MANET applications because they contain firm real-time transactions and cannot tolerate the stale data. An example of a firm transaction is “Finding the latest location of enemy before firing a missile in the battlefield”.

4. Summaries of Reviewed Techniques

In this section, we summarize each discussed design issue and its possible solutions from the existing techniques along with the identified MANET database characteristics (the environment issues), which are reviewed in Sections 2 and 3. For every issue, the following specified terms/answers are expected to distinguish the reviewed techniques.

- *Types of CC Algorithms*: Which type does the CC technique adopt to guarantee the isolation property: pessimistic, optimistic, or hybrid?
- *Rules of Producing Serializability*: Which rule does the CC technique utilize to produce serializability: locking, timestamp ordering, serialization graph testing or serial execution?
- *CC Granularity*: What kind of granularity does the CC technique apply: row level, page level, table level, database level or unspecified?
- *Mobile System Architecture*: Which architecture does the CC technique adopt: traditional mobile network or MANET?

- *Location of CC Manager(s)*: Where are the CC managers located: centralized or distributed?
- *Improving System Performance*: How does the CC technique improve the system performance: response time or abort rate?
- *Cascading Rollback*: Does the CC technique handle the cascading rollback?
- *Insert and Delete Operations*: Does the CC technique handle insert and delete operations?
- *Level of Consistency*: Which level of consistency does the CC technique support: relaxed or strict?
- *Transaction Model*: Which transaction model is utilized in the CC technique: flat or nested?
- *Global Serializability*: Does the CC technique guarantee the isolation property of global transactions?
- *Degree of Local Autonomy*: In order to guarantee the isolation property of global transactions, what does the CC technique do to the autonomy of local database system: preserved, violated or unspecified?
- *Cached/Replicated Data*: Does the CC technique apply any caching/replication scheme?
- *Real-Time Applications*: Does the CC technique support real-time transactions?
- *Mobility*: Does the CC technique address mobility?
- *Low Bandwidth*: Does the CC technique address low bandwidth?
- *Multi-hop Communication*: Does the CC technique address multi-hop communication?

- *Limited Battery Power*: Does the CC technique address limited battery power?
- *Limited Storage*: Does the CC technique address limited storage?
- *Frequent Disconnections*: Does the CC technique address frequent disconnections?
- *Long-lived Transactions*: Does the CC technique address long-lived transactions?

As shown in Table 1, none of the reviewed techniques addresses all the general issues and identified MANET database characteristics. In addition, we can observe the following from the table:

- Most techniques are pessimistic and guarantee serializability by using locking, but they do not fit well in MANETs databases because blocking time may be unbounded and abort rate may be high as a consequence.
- No techniques specify the CC granularity.
- All techniques except SESAMO [6] adopt traditional mobile networks, while SESAMO applies in MANETs.
- CC managers are distributed in all techniques, so that the bottleneck problem does not occur.
- Each technique either improves the response time or reduces the abort rate to improve the system performance, but no one addresses both.
- Only KT [9] and PGSG [8] do not handle the cascading rollback because they may apply compensating transactions to rollback committed transactions if necessary.
- Only DHP-2PL [15] partially takes into account insert and delete operations.

- All techniques support the strict consistency and use the flat transaction model except KT [9] and PGSG [8], which relax the consistency and use the nested transaction model.
- Half of techniques guarantee global serializability. KT [9] supports global transactions, but it completely relaxed the global serializability to reduce the abort rate.
- Most techniques do not specify the autonomy of local database system because of no global transactions.
- Most techniques apply caching/replication schemes to overcome the MANET database characteristics, but these schemes will not work without the data broadcast by mobile support stations (MSS), which are static and have unlimited power and high bandwidth.
- Only DHP-2PL [15] and MV [16] support real-time application.
- No techniques provide any solutions to address the multi-hop communication and limited storage.
- Some techniques, which are proposed for traditional mobile networks, take into account mobility, low bandwidth, limited battery power or frequent disconnections. But, these cannot be done without applying caching/replication and heavily relying on MSSs to either broadcast data periodically for mobile nodes or process transactions on behalf of mobile nodes. However, in MANETs, it is impossible for some node to play the same role like a MSS.

- SESAMO [6] addresses low bandwidth and long-lived transactions, but both of them are accomplished by assuming that global transactions do conflict with each other.
- All techniques, which are proposed for traditional mobile networks, support long-lived transactions by relying on MSSs, using caching/replication or relaxing the consistency of databases.

5. Conclusions

In this paper, we discussed the issues that must be taken into considerations when designing a CC algorithm for MANET databases. In the mean time, we reviewed how existing CC techniques for traditional mobile databases and MANET databases handled those issues. Our study shows that none of the reviewed CC techniques addresses all identified issues and the research in the area of MANET database CC algorithms is still in its early stage. In order to support MANET users who want to correctly access and manipulate data anytime and anywhere, how to design a suitable CC algorithm for MANET databases should receive much attention. While resolving the identified issues would accelerate the design process.

Acknowledgement

This material is based upon work supported by (while serving at) the National Science Foundation (NSF) and the NSF Grant No. IIS-0312746. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References:

[1] M. Abolhasan and T. Eyers. Sparse ad hoc networks for the desert. *Desert Knowledge Cooperative Research Centre Report*, no. 23, (Alice Springs, 2007).

[2] A. S. Al-Mogren and M. H. Dunham. Buc, a simple yet efficient concurrency control technique for mobile data broadcast environment. *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, (2001) 564-569.

[3] G. Bernard, C. Roncancio, P. Serrano-Alvarado and P. Valduriez. Mobile Databases: a Selection of Open Issues and Research Directions. *ACM SIGMOD Record*, 33 (2) (2004) 78-83.

[4] P. Bernstein, V. Hadzilacos and N. Goodman, *Concurrency Control and Recovery in Database Systems* (Addison-Wesley, Reading, MA, 1987).

[5] B. Bhargava. Concurrency Control in Database Systems. *IEEE Transactions on Knowledge and Data Engineering*, 11 (1) (1999) 3-16.

[6] A. Brayner and F. S. Alencar. A Semantic-serializability Based Fully-Distributed Concurrency Control Mechanism for Mobile Multi-database Systems. *Proceeding of the 16th International Workshop on Database and Expert Systems Applications (DEXA '05)*, (2005) pp. 1085-1089.

[7] I. Chlamtac, M. Conti and J. Liu. Mobile Ad Hoc Networking: Imperatives and challenges. *Ad Hoc Networks Publication*, 1 (1) (2003) 13-64.

[8] R. Dirckze and L. Gruenwald. A pre-serialization transaction management technique for mobile multi-databases. *ACM Mobile Networks and Applications*, 5 (4) (2000) 311-321.

[9] M. Dunham, A. Helal and S. Balakrishnan. A mobile transaction model captures both the data and movement behavior. *ACM Mobile Networks and Applications*, 2 (2) (1997) 149-162.

[10] L. Fife and L. Gruenwald. Research issues for Data Communication in Mobile Ad-Hoc Network Database Systems. *ACM SIGMOD RECORD,* 32 (2) (2003) 42-47.

[11] D. Georgakopoulos, M. Rusinkiewicz and A. Sheth. On serializability of multidatabase transactions through forced local conflicts. *Processing of the 7th International Conference on Data Engineering*, (1991) 314-323.

[12] P. Hofmann, K. Kuladinithi, A. Timm-Giel, C. Gorg, C. Bettstetter, F. Capman and C. Toulsaly. Are IEEE 802 Wireless Technologies Suited for Fire Fighters. *European Wireless 2006* (2006).

[13] San-Yih Hwang. On Optimistic Methods for Mobile Transactions. *Journal of Information Science and Engineering*, 16 (4) (2003) 535-554.

[14] Z. Itani, H. Diab and H. Artail. Efficient Pull Based Replication and Synchronization for Mobile Databases. *International Conference Pervasive Services*, (2005) 401-404.

[15] K. Lam, T. Kuo, W. Tsang and G. Law. Concurrency Control in Mobile Distributed Real-Time Database System. *Information Systems*, 25 (4) (2000) 261-322.

[16] K. Lam, G. Li and T. Kuo. A Multi-version Data Model for Executing Real-Time Transactions in a Mobile Environment. *ACM MobiDE*, (2001) 90-97.

[17] V. Lee, K. Lam, S. Son and E. Chan. On Transaction Processing with Partial Validation and Timestamp Ordering in Mobile Broadcast Environments. *IEEE Transactions on Computers*, 51 (10) (2002)1196 –1211.

[18] J. Lim and A. Hurson. Transaction Processing in Mobile Heterogeneous Database System. *IEEE Transactions on Knowledge and Data Engineering*, 14 (6) (2002) 1330-1346.

[19] J. Eliot B. Moss. Nested Transactions: An Approach to Reliable Distributed Computing. *The MIT Press*, (Cambridge, MA,1985).

[20] N. Prbhu, V. Kumar, I. Ray and G. Yang. Concurrency Control in Mobile Database Systems. *Proceedings of the 18th International Conference on Advanced Information Networking and Application*, (2004) 83-86.

[21] P. Serrano-Alvarado, C. Roncancio and M. Adiba. A Survey of Mobile Transactions. *Distributed and Parallel Databases*, 16 (2) (2004) 193-230.

[22] A. Silberschatz, H. F. Korth and S. Sudarshan, *Database Systems Concepts*, McGraw-Hill College (2005).

[23] G. Verdun. Understanding Battery Life in Portable Computers. *Dell White Paper* (2005).

[24] R. Zimmermann and D. A. Desai. *Ethernet Interface for Head-Mounted Display*. Technical Report, Integrated Media System Center, University of Southern California (2005).

Techniques/Issues		BUC [2]	CCM [20]	DHP-2PL [15]	KT [9]	MTC-SG/SQ [13]	MV [16]	PGSG [8]	PVTO [17]	SESAMO [6]	V-Lock [18]
General Issues	Type of CC Algorithm	Pessimistic	Pessimistic	Pessimistic	Pessimistic	Hybrid	Pessimistic	Hybrid	Optimistic	Pessimistic	Pessimistic
	Rule of Producing Serializability	Locking	Serial Execution	Locking	Locking	Locking, timestamp ordering and serialization graph testing	Timestamp Ordering	Timestamp ordering, and serialization graph testing	Timestamp Ordering	Locking	Locking
	CC Granularity	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified
	Mobile System Architecture	Traditional mobile network	Traditional mobile network	Traditional mobile network	Traditional mobile network	Traditional mobile network	Traditional mobile network	Traditional mobile network	Traditional mobile network	MANET	Traditional mobile network
	Location of CC Manager	Distributed	Distributed	Distributed	Distributed	Distributed	Distributed	Distributed	Distributed	Distributed	Distributed
	Improving System Performance	Response time	Response time	Response time	Response time	Abort rate	Abort rate	Abort rate	Abort rate	Response time	Response time
	Cascading Rollback	No	No	No	Yes	No	No	Yes	No	No	No
	Insert and delete Operations	No	No	Yes, but partially	No	No	No	No	No	No	No
	Level of Consistency	Strict	Strict	Strict	Relaxed	Strict	Strict	Relaxed or Strict	Strict	Strict	Strict
	Transaction Model	Flat	Flat	Flat	Nested	Flat	Flat	Nested	Flat	Flat	Flat
Application Dependent Issues	Global Serializability	No	No	Yes	No	No	Yes	Yes	No	Yes	Yes
	Degree of Autonomy	Unspecified	Unspecified	Unspecified	Preserved	Unspecified	Unspecified	Preserved	Unspecified	Preserved	Violated
	Cached/replicated Data	Yes	Yes	No	No	Yes	Yes	No	Yes	No	Yes
	Real-time Application	No	No	Yes	No	No	Yes	No	No	No	No
MANET Database Characteristics	Mobility	No	No	No	Yes	Yes	No	Yes	No	No	No
	Low Bandwidth	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes
	Multi-hop Communication	No	No	No	No	No	No	No	No	No	No
	Limited Battery Power	Yes	Yes	No	No	Yes	No	No	Yes	No	No
	Limited Storage	No	No	No	No	No	No	No	No	No	No
	Frequent Disconnections	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes
	Long-live Transactions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 1: Summary of the reviewed techniques and issues