Neighborhood-Consistent Transaction Management for Pervasive Computing Environments*

Filip Perich, Anupam Joshi, Yelena Yesha, and Timothy Finin

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle,
Baltimore, MD 21250, USA
{fperic1,joshi,yeyesha,finin}@csee.umbc.edu

Abstract. This paper examines the problem of transaction management in pervasive computing environments and presents a new approach to address them. We represent each entity as a mobile or static semi-autonomous device. The purpose of each device is to satisfy user queries based on its local data repository and interactions with other devices currently in its vicinity. Pervasive environments, unlike traditional mobile computing paradigm, do not differentiate between clients and servers that are located in a fixed, wired infrastructure. Consequently, we model all devices as peers. These environments also relax other assumptions made by mobile computing paradigm, such as the possibility of reconnection with a given device, support from wired infrastructure, or the presence of a global schema. These fundamental characteristics of pervasive computing environments limit the use of techniques developed for transactions in a "mobile" computing environments. We define an alternative optimistic transaction model whose main emphasis is to provide a high rate of successful transaction terminations and to maintain a neighborhood-based consistency. The model accomplishes this via the help of active witnesses and by employing an epidemic voting protocol. The advantage of our model is that it enables two or more peers to engage in a reliable and consistent transaction while in a pervasive environment without assuming that they can talk to each other via infrastructure such as base stations. The advantage of using active witnesses and an epidemic voting protocol is that transaction termination does not depend on any single point of a failure. Additionally, the use of an epidemic voting protocol does not require all involved entities to be simultaneously connected at any time and, therefore, further overcomes the dynamic nature of the environments. We present the implementation of the model and results from simulations.

1 Introduction

Maintaining data consistency between devices in distributed mobile environments has always been, and continues to be, a challenge. These environments represent networks composed of stationary and mobile nodes that share a subset of a global data repository. The devices use their network connectivity to exchange data with other nodes in the

* This work was supported in part by NSF awards IIS 9875433 and 0209001, and DARPA contract F30602-00-2-0591.

V. Mařík et al. (Eds.): DEXA 2003, LNCS 2736, pp. 276–286, 2003.

network. In order to operate correctly, devices involved in a transaction must ensure that their data repositories remain in a consistent state. While stationary nodes often embody powerful computers located in a fixed, wired infrastructure, mobile nodes represent devices with low-bandwidth communication, limited battery life and with limitation to other resources. Consequently, transacting devices must accommodate mobility and, in turn, possible failures due to a network disconnection. The challenge of providing data consistency is especially substantial for pervasive computing environments.

Pervasive computing environments extend the traditional concept of mobile networks [10,18]. Mobile devices in pervasive computing environments consist of hand helds, wearables, computers in vehicles, computers embedded in the physical infrastructure, and (nano) sensors. A device satisfies user queries by relying on its local data repository and data available in other devices in its vicinity. Additionally, every device is equipped with short range ad-hoc networking technologies such as Bluetooth [3]. The ad-hoc networking technology allows mobile devices to spontaneously interact with other devices, both fixed and mobile, in their vicinity. For example, two cars passing each other on the street can establish a network connection and exchange data while within range of each other. At the same time, pervasive computing environments do not guarantee any infrastructure support, a crucial requirement for traditional mobile systems [6,7,24]. Hence unlike traditional mobile computing, the pervasive environment does not differentiate between mobile clients and servers located in a fixed, wired infrastructure. Instead we model all devices as peers and any two devices may engage in a transaction, a case not covered by traditional mobile computing paradigm. In the mobile computing paradigm, only one transacting device, the mobile client, is allowed to move during a transaction. This allowed previous solutions to rely on the help of the infrastructure by using mobile support stations as proxies; however, there is no default infrastructure support in pervasive computing environments. Additionally, pervasive environments relax other assumptions made in mobile computing paradigm. As all devices may be mobile, the vicinity of each device is likely to change in both spatial and temporal dimensions. This not only limits data and data source availability but the serendipitous nature of the environment also limits the possibility of reconnection between transacting devices. In pervasive computing environments, there is no guarantee that all devices wishing to transact may be concurrently available and that two disconnected devices will *meet* again. For example, when two people serendipitously meet at an airport and agree to exchange a song for a micro-payment, their electronic wallets must be updated correctly even when one person leaves the airport before the transaction completes [1]. Consequently, transacting peers must either trust each other or rely on a third party. In a traditional mobile paradigm, the third party is a server located in a fixed, wired infrastructure. In that case, transacting peers must send all relevant data to the server before they disconnect. This may not always be possible in pervasive environments. Instead, an alternative approach is to use other peers in the environment as third parties. This raises the issue of trust since there is no guarantee that these peers will behave correctly. We address the issue via the use of a random witness selection policy which reduces the probability of obtaining malicious witnesses. In summary, the change in perception of mobile devices, together with other characteristics of pervasive computing environments, limits the use of traditional mobile transactions.

To address the problem, we present a novel transaction model designed for use in pervasive computing environments. We focus on maintaining consistency of transactions, which has generally been termed as the most important ACID property of transactions for mobile environments [9]. Consistency is, however, not critical in read only transactions [18]. Our *Neighborhood-Consistent Transaction* model (NC-Transaction) provides a higher rate of successful transactions in comparison to models designed for traditional mobile computing environments. NC-Transaction maintains neighborhood consistency among devices in the vicinity. It does not ensure global consistency, a task often impossible since there is no guarantee that two devices will ever reconnect in a pervasive computing environment. NC-Transaction accomplishes neighborhood consistency and high successful termination rate by employing active witnesses and an epidemic voting protocol. NC-Transaction defines witnesses as devices in a vicinity that can *hear* both transacting devices and agree to monitor the status of a transaction. Each witness can cast a vote to commit or abort a transaction. A transacting device must collect a quorum of the votes, defined as a percentage of all witness votes, to decide on the final termination action for a transaction. By using a voting scheme and redundancy of witnesses, NC-Transaction ensures that transacting devices terminate in a consistent state. Additionally, information stored by each witness can be used to resolve conflicts between devices involved in a transaction.

The remainder of the paper is structured as follows: We present related work in Section 2. In Section 3, we define the NC-Transaction model and the generic consistency protocol in the context of MoGATU [18]. In Section 4, we present our experimental setup. We empirically show how the NC-Transaction model improves successful transaction termination rate and how it affects the computing cost for all entities in the environment. We conclude and describe directions for future work in Section 5.

2 Related Work

The NC-Transaction model is designed within the context of MoGATU [18] – a lightweight architecture for profile–driven data management in pervasive computing environments. The work on MoGATU spans research areas on both data management and ad-hoc networking technologies. MoGATU is currently implemented both as a prototype running on Linux based computers with support for Bluetooth and Ad–Hoc 802.11 [11] and as an extension of the GloMoSim simulator [23].

The problem of transaction management in wireless networks has drawn a significant degree of attention. Most of the proposed solutions are based a client/proxy/server model. These solutions place primary data on servers located within the wired infrastructure and treat mobile devices solely as clients. Using this approach, the solutions assume that mobile devices are only one-hop away from a wired network and attempt to overcome issues caused by the characteristics of the one-hop wireless link. The solutions accomplish their task by relaxing some of the ACID properties, by enabling a non-blocking execution in a disconnected mode and by adapting commit protocols [5, 8,15,9,20]. For example, Kangaroo Transaction [9] model addresses mobility of mobile hosts (MH) that hop from one mobile support station (MSS) to another. The model exploits the concept of split transactions. Each MSS acts as a proxy between a server

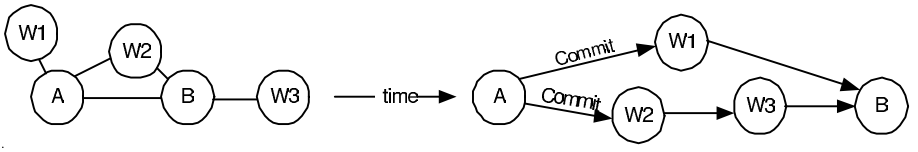

Fig. 1. Help of Witnesses in NC-Transaction

and a mobile client and manages all sub-transactions for the time period it serves as a proxy. PRO-MOTION [22] is a mobile transaction processing system that also exploits the concept of nested-split transactions by relaxing the atomicity restriction and using *compacts*, a local cache of objects with additional state information. These objects are synchronized upon a reconnection. In contrast, the Bayou architecture [8] uses an epidemic protocol for synchronizing objects; however, objects must be first committed on a primary copy. An alternative solution is presented in the context of Deno [13], which is also a replicated-object system for mobile environments. Unlike in Bayou, every replica in Deno has an equal chance of committing an update whenever it can obtain a voting quorum. Each replica obtains a quorum by gathering weighted votes from other replicas in the system and by providing its vote to others. Similarly, Coda [14] and Ficus [17] provide support for disconnected operations in the domain of distributed file systems. Each mobile device can modify its cached files while disconnected. Upon a reconnection to the network, the device connects to a subset of replica holders, the so-called AVSG in Coda, in order to commit its updates. These servers then propagate updates to the remaining nodes. Unlike the NC-Transaction, however, most of these solutions either depend on infrastructure support or require reconnection in order to synchronize *dirty* data, an option not guaranteed in pervasive computing environments.

3 Neighborhood-Consistent Transaction Model

The NC-Transaction model is defined in terms of a session among multiple devices in a vicinity that wish to transact with each other. A session extends the classical concept of a transaction defined for distributed database systems [16]. Traditionally, transactions consist of three phases – *start*, *execution* and *end*. In the first phase, *S*, all devices are synchronized. In the next step, each device executes a sequence of read and write operations, $\{R/W\}$. Finally, in the last step, all devices synchronize themselves to either unilaterally commit or abort the transaction, denoted as *C/A*. Formally, a transaction is defined as:

$$T = (S, \{R/W\}, C/A) \tag{1}$$

Similarly, a session in NC-Transaction consists of three phases: (i) negotiation, (ii) execution and (iii) termination. The most important part of NC-Transaction is to terminate in a consistent state. Transacting devices in a NC-Transaction achieve this goal by soliciting and relying on the help of other devices in the vicinity that agree to serve as active witnesses. The use of witnesses has traditionally been a social device to ensure the fairness and correctness of an event involving two parties [4]. Witnesses are requested to monitor an event and provide testimony that can help decide its outcome.

NC-Transaction abstracts the notion of witnesses for a similar purpose; to ensure consistency via fairness and correctness. In our model, transacting devices request other

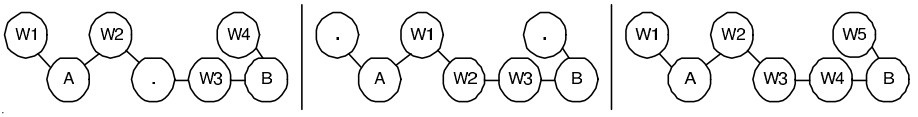

Fig. 2. Alternative Witness Selection Protocols for NC-Transaction

devices in their vicinity to monitor their transaction and these witnesses are responsible for voting on the veracity of the transaction's outcome. The advantage of using witnesses for transactions is threefold: (i) In traditional mobile computing paradigms, commit protocols, such as 2PC [16], depend on network reliability and the existence of a wired infrastructure, neither of which is guaranteed in pervasive computing environments. As we will show, witnesses mitigate the negative impacts on transactions within pervasive computing environments by serving as intermediaries. As illustrated in Figure 1, witnesses can propagate intentions to abort or commit from one transacting device to another. (ii) Witnesses provide redundancy, increasing the probability of a consistent outcome of a transaction because they do not rely on each other. Each witness monitors transacting devices independently. It attempts to collect enough information to decide whether a transaction should be committed or aborted. Transacting devices can then choose the number of witness' votes necessary for a forming a quorum to decide on the terminating action. (iii) Witnesses provide a higher level of assurance on fairness and correctness than transacting devices would have otherwise. This is due to the fact that the decision to commit or abort does not depend on a single device, rather it must be a *collective decision*.

Negotiation Phase – In the initial phase of NC-Transaction, a device wishing to transact with other devices first negotiates the query terms that each should execute, the planned duration of a transaction and the set of active witnesses. The negotiation is based on the principles of Contract Nets [21]. NC-Transaction supports any type of a query that can be decomposed into a sequence of reads and writes over a local data repository for each transacting device. In addition, the query may include operations for exchanging data between multiple devices. Mobile peers also negotiate the planned duration of the transaction to provide a default fall-back for terminating a transaction by aborting it. Each transacting device must gather enough votes from witnesses to commit a transaction before the time period elapses in order to commit the results locally.

Witness Selection Protocols – During the *negotiation* phase, transacting devices also select other devices in the vicinity as active witnesses. The transacting devices attempt to collect at least three witnesses by employing the principles of Contract Nets. Each node in the vicinity is presented with a list of transacting devices, terms of queries and the planned duration period. A possible witness evaluates the terms and decides to accept or reject the task. The transacting devices determine the list of devices that agree to witness a transaction and choose at most n witnesses, a variable parameter for NC-Transaction. We define three methods for selecting witnesses in pervasive computing environments and illustrate them in Figure 2. In the first method, each transacting device attempts to use only its current one-hop neighbors as witnesses. The advantage of this approach is that mobile nodes in a human environment move as a group in a predictable manner. For example, many cars on a street travel in the same direction. This improves the chances that witnesses and transacting devices are able to *hear* each other. Alternatively,

transacting devices attempt to use only those devices that route packets between them. The advantage of this approach is that the underlying routing protocol may utilize a majority of the route for most of the transaction, and thus the devices are guaranteed to *hear* each other for most of the time. The third method combines the other two approaches by selecting witnesses on the route and those currently around each transacting device.

Execution Phase – In this phase, each transacting device begins to execute the negotiated sequence of reads and writes over its local data repository. Given the ad-hoc nature of the pervasive computing environments, the device may not be able to obtain "locks" on all replicas in the environment. Instead, we assume an optimistic concurrency control which allows each device to modify its data independently from others. When a device either completes or has to abort its part of the transaction, it attempts to inform as many witnesses as possible about its intention. In contrast, each witness uses the *execution* phase to collect evidence in order to monitor the progress of the transaction and intentions about a commit/abort by transacting devices. Once a witness has enough knowledge to decide on an outcome, it attempts to send a commit or abort message to all transacting devices.

Termination Phase – In the *termination* phase each transacting device either commits or aborts the transaction. A device can commit a transaction only when it is able to gather a quorum of commit votes from witnesses [13]. In NC-Transaction, a voting quorum represents a predefined percentage of votes cast by all witnesses and can range from at least one vote to all 100% votes from all witnesses.

4 Performance Experiments

The primary goal of NC-Transaction is to provide a high rate of successful transaction terminations while maintaining neighborhood-based consistency. Consequently, in this section we focus on the following properties: (i) performance of NC-Transaction compared to that of a representative of traditional mobile transaction models, (ii) the appropriate number of witnesses per NC-Transaction and (iii) the effects of different percentages for voting quorums on transaction termination and neighborhood consistency. We use Kangaroo Transaction [9] as the representative of traditional mobile transaction models but have modified it in order to suit the experimental environment by allowing the server to be represented by a mobile device. We have implemented both transaction models within the context of the MoGATU framework using the GloMoSim simulator [23].

We employ a spatio-temporal environment that consists of a 200 x 200m field and 100 nodes – 36 stationary and 64 mobile nodes – for a period of 50 minutes. The stationary nodes represent a variable infrastructure support, which is required for the traditional mobile transaction model. In contrast, the mobile nodes represent devices in a pervasive computing environment. The mobile devices move according to a random waypoint mobility model [2]. Using this model, a device chooses a random point within the field and a random speed. Next, the device moves using the chosen speed and direction until it reaches its target location. The device then waits for a predefined time period and repeats the process. In our experiments, we have set the waiting period to 5 seconds and varied the speed from 1m/s, 3m/s to 9m/s to simulate different mobile environments. All

devices in the environment employ the AODV protocol [19] for routing packets at the network level.

To study the performance of NC-Transaction, we concentrate on transactions initiated by a mobile device *A*. *A* attempts to initiate a transaction with another mobile device *B* at one minute intervals, a total of 50 transactions per simulation run. Each transaction is negotiated to last from 10 to 20 seconds, after which both devices must commit the transaction based on the voted received from witnesses or, by default, abort. We vary the probability of infrastructure support from 0% to 100%, in order to compare the performance to that of Kangaroo Transactions. Complete, 100%, infrastructure support represents an environment where each mobile device is in the range of a stationary node. For lesser probability values, the number of stationary nodes decreases accordingly from 36 to 0, thus creating areas of no support. We differentiate among three types of NC-Transaction based on the different witness selection policies specified in Section 3. Using these policies, witnesses to a transaction between *A* and *B* can include only their one-hop peers or only those devices on the route between *A* and *B* or a combination of both. Additionally, we vary the number of witnesses that *A* and *B* must gather in order to start executing a transaction from 3 to 33. Finally, we vary the number of votes required for quorum from 0% to 100%.

4.1 Infrastructure Support vs. Transaction Success and Consistency

In the first experiment, we compare the performance of NC-Transaction against that of a traditional mobile transaction. We differentiate among three versions of NC-Transaction based on a witness selection policy: (i) one-hop peers *O NC-T*, (ii) routing peers *R NC-T*, and (iii) both one-hop and routing peers *O+R NC-T*. Figure 3 shows the empirical results for different levels of infrastructure, base station type support. For traditional transactions, only infrastructure nodes can route between *A* and *B*. For NC-Transactions every device, mobile or within the infrastructure, can provide routing functionality for any other device. Each stacked bar represents the average distribution of initiated transactions given a transaction model and infrastructure support. We differentiate among five cases: (i) a transaction is successfully executed and terminated by both *A* and *B*, (ii) a transaction is aborted in a consistent state by both *A* and *B*, (iii) a transaction does not start because *A* or *B* cannot be reached and, hence, leaves both *A* and *B* in a consistent state, (iv) a transaction does not start because *A* and *B* are unable to obtain enough witnesses, again resulting in a consistent state, and (v) a transaction leaves *A* and *B* in an inconsistent state. Accordingly, a transaction model should maximize the amount of transactions falling into the first category and minimize all other cases. Due to the limited amount of space, we report only a subset of our results. Specifically, we have fixed the witness size to 5 and the voting quorum percentage to 66%, which yielded optimal results against other witness group sizes and voting quorums in our experiments.

As depicted in Figure 3, NC-Transaction performs better than the traditional mobile model. This is expected because NC-Transaction was defined specifically for the pervasive computing environment. The NC-Transaction model using broadcast-based witness selection protocol performs the best. On average, 75% of transactions successfully executed and only 0.3% of transactions resulted in an inconsistent state. This may be because some witnesses travel in a similar direction as *A* or *B* and are, therefore, able to

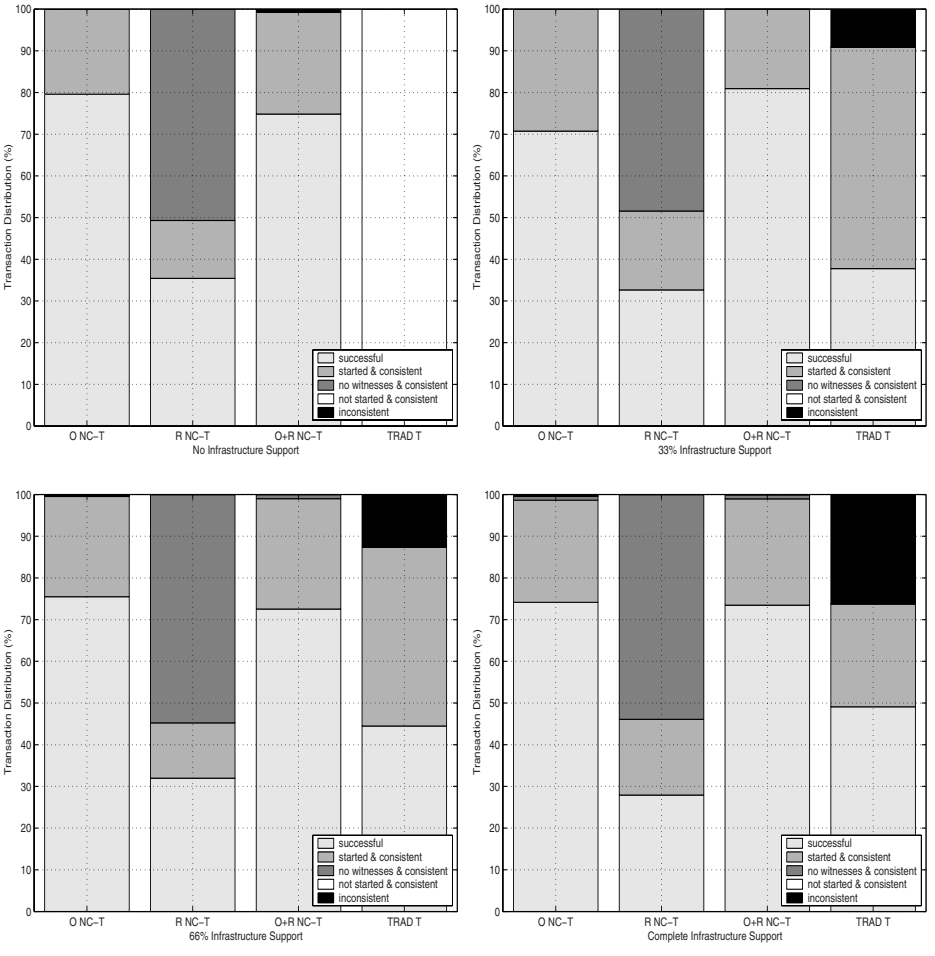

Fig. 3. Infrastructure Support vs. Transaction Success and Consistency

monitor the ongoing transactions. NC-Transaction using routing-peer witness selection protocol does not perform as well because, 51.9% of the time on average, the model was unable to collect enough witnesses. This may be because *A* and *B* were often only 3 hops away during transaction initiation. NC-Transaction combining one-hop and routing-peer witness selection protocols performs similar to the *O NC-T* model. As expected, in environments without infrastructure support, no Kangaroo Transaction could be initiated because *A* and *B* could not communicate with each other. As infrastructure support was increased, *A* and *B* were able to exchange messages and initiate transactions; however, many of the transactions gracefully aborted or created inconsistency between *A* and *B*. This is due to the fact, that *B* committed and informed *A* to also commit; however, *A* did not receive the instruction since it had moved out of the range and could not connect to the infrastructure.

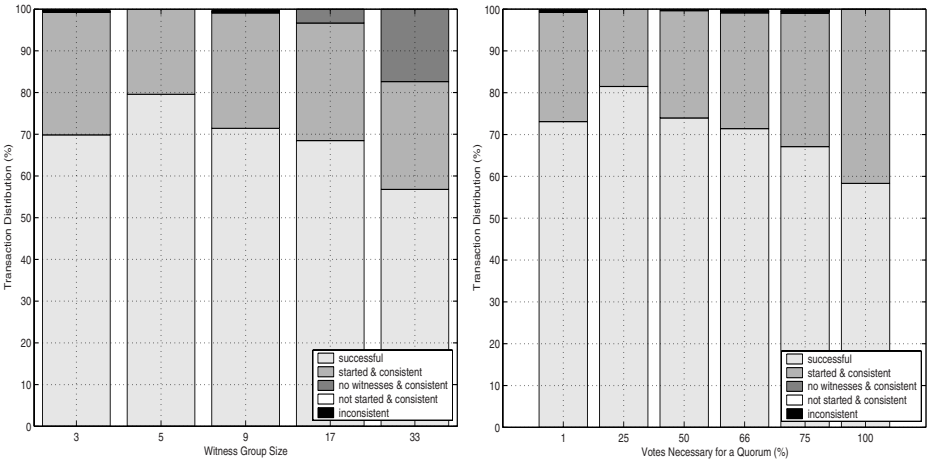

Fig. 4. (a) Witness Group Size vs. Transaction Execution. (b) Voting Quorum vs. Successful Transaction Execution.

4.2 Witness Group Size vs. Transaction Execution

In the next experiment, we study the effect of witness group size on transaction execution. We use only the NC-Transaction model utilizing the one-hop witness selection protocol because it performed the best in the previous experiment. We set the voting quorum to 66%, requiring a device to obtain at least two thirds of all votes in order to commit or abort before a transaction deadline, and exponentially vary the number of witnesses, K, from 3 to 33. As depicted in Figure 4 (a), we see that for a small number of witnesses, *i.e.,* $K<5$, message loss can cause inconsistency on or aborts by both A and B. This is because in these cases a transaction depends on every witness. Optimal performance was obtained for $K=5$ as suggested in the previous experiment. For larger groups of required witnesses, we see that the number of successfully executed transactions decreases. This is because many witnesses were unable to cast a vote or that transacting devices were unable to collect enough votes. In addition, for very large groups, we also see that A and B are unable to obtain enough witnesses, 17.7% for $K=33$, in order to start executing a transaction.

4.3 Voting Quorum vs. Successful Transaction Execution

In this experiment, we again study the performance of the NC-Transaction model utilizing a one-hop witness selection protocol; however, we fix the size of the required witness group K to 9. We use this base size in order to better illustrate the effects of different voting quorums, Q, on transaction execution. We vary Q from 0% to 100% and measure its effects on consistency and transaction execution. As shown in Figure 4 (b), the best performance was obtained for $Q=25\%$, which is not as good as the performance for $K=5$ and $Q=66\%$ from the first experiment. For higher values of Q, we see that the number of gracefully aborted transactions increases as the transacting devices A and B were unable to obtain enough votes matching the voting quorum level.

5 Conclusions and Future Work

We have presented the design of the NC-Transaction model and its implementation in the context of the MoGATU framework. We have shown that NC-Transaction, using one-hop witness selection protocol, radically increases the number of successfully executed transactions and at the same time decreases the number of inconsistent transactions. NC-Transaction accomplishes this via the help of active witnesses and by employing an epidemic voting protocol. The role of each witness is to monitor an ongoing transaction and gather enough information to assist in terminating the transaction. In turn, each transacting device serendipitously collects votes from witnesses in order to successfully commit or abort. In this manner, a device does not depend on a single controller and is, therefore, less prone to errors due to the nature of pervasive computing environments. We have measured the effects on transaction execution and consistency for different witness group sizes, K, and the number of votes required to form a quorum, Q. Our measurements suggest best performance for K=5 and Q=66%. Our measurements also suggest that both K and Q play an important role in NC-Transaction performance.

We have not addressed the issues related to trust. In our initial design, we have assumed that all devices are reliable and, therefore, each transacting or witnessing device behaves correctly. This clearly may not be the case of a *real-world* pervasive environment where any device possessing ad-hoc network connectivity may choose to interact with its peers. We will address this issue in future work by incorporating distributed trust mechanisms discussed in [12].

References

1. S. Avancha, P. D'souza, F. Perich, A. Joshi, and Y. Yesha. P2P M-Commerce in Pervasive Environments. In *ACM SIGEcom Exchanges*, 2003.
2. C. Bettstetter. Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks. In *MSWiM'01*, 2001.
3. Bluetooth SIG. Specification. `http://bluetooth.com/`.
4. A. Borr. Transaction Monitoring in Encompass: Reliable Distributed Transaction Processing. In *VLDB*, 1981.
5. O. Bukhres, S. Morton, P. Zhang, E. Vanderdijs, C. Crawley, J. Platt, and M. Mossman. A Proposed Mobile Architecture for Distributed Database Environment. Technical report, Indiana University, Purdue University, 1997.
6. M. Cherniak, E. Galvez, D. Brooks, M. Franklin, and S. Zdonik. Profile Driven Data Management. In *VLDB*, 2002.
7. P. Chrysanthis and E. Pitoura. Mobile and Wireless Database Access for Pervasive Computing. In *ICDE*, 2000.
8. A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and B. B. Welch. The bayou architecture: Support for data sharing among mobile users. In *IEEE Workshop on Mobile Computing Systems & Applications*, 1994.
9. M. Dunham, A. Helal, and S. Balakrishnan. A Mobile Transaction Model that Captures Both the Data Movement and Behavior. *ACM MONET*, 1997.
10. M. Franklin. Challenges in ubiquitous data management. In *Informatics*, 2001.
11. IEEE 802.11 Working Group. Ad-hoc 802.11. `http://ieee802org/11`.
12. L. Kagal. Rei : A Policy Language for the Me-Centric Project. Technical report, HP Labs, 2002.

13. P. Keleher and U. Cetintemel. Consistency Management in Deno. *ACM MONET*, 1999.
14. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 1992.
15. S. Lauzac and P. Chrysanthis. Utilizing versions of views within a mobile environment. In *Conference on Computing and Information*, 1998.
16. M. Oezsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, Inc., New Jersey, 2nd edition edition, 1999.
17. T. Page, R. Guy, J. Heidemann, D. Ratner, P. Reiher, A. Goel, G. Kuenning, and G. Popek. Perspectives on Optimistically Replicated Peer-to-Peer Filing. In *Software – Practice and Experience*, 1998.
18. F. Perich, S. Avancha, D. Chakraborty, A. Joshi, and Y. Yesha. Profile Driven Data Management for Pervasive Environments. In *DEXA*, 2002.
19. C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *IEEE Mobile Computing Systems and Applications*, 1999.
20. E. Pitoura. A Replication Schema to Support Weak Connectivity in Mobile Information Systems. In *DEXA*, 1996.
21. R. Smith. *Readings in Distributed Artificial Intelligence*, chapter The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. 1988.
22. G. Walborn and P. Chrysanthis. Transaction Processing in PRO-MOTION. In *ACM Symposium on Applied Computing*, 1999.
23. X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Workshop on Parallel and Distributed Simulation*, 1998.
24. Y. Zhang and O.Wolfson. Satelitte-Based Information Services. *ACM MONET*, 2002.