

# Information Dissemination in Partitionable Mobile Ad Hoc Networks

Goutham Karumanchi      Srinivasan Muralidharan      Ravi Prakash  
Department of Computer Science  
University of Texas at Dallas  
Richardson, TX 75083-0688.  
E-mail: {goutham,mrsrini,ravip}@utdallas.edu

## Abstract

*Ad-hoc wireless networks have no wired component, and may have unpredictable mobility pattern. Such networks can get partitioned and reconnected several times. One possible approach for information dissemination in such networks is to replicate information at multiple nodes acting as repositories, and employ quorum based strategies to update and query information. We propose three such strategies that also use local knowledge about the reachability of repositories to judiciously select quorums. The primary goal is high availability of information in the face of network partitioning. We also consider four policies to determine the appropriate time to perform updates. Experimental results indicate that a hybrid information management strategy and an absolute connectivity-based update trigger policy are most suited for partitionable ad-hoc networks.*

## 1 Introduction

Existing solutions for dissemination of information in static or cellular networks may not be applicable to ad-hoc networks. First, these solutions usually do not consider changing topology of the network backbone that contains the information servers. Second, the possibility of partitioning means that some nodes may not be able to communicate updates to other nodes and/or may be unable to retrieve the latest information on queries. Third, earlier solutions that do consider network partitioning approach the problem from the direction of replica consistency in distributed databases. While the problems are similar, several instances of information dissemination problem in ad-hoc networks are simpler in nature. Employing the sophisticated replica consistency solutions would result in reduced availability of data, while also incurring unacceptably high communication overheads. Fourth, unlike traditional distributed database systems where the timing of updates is independent of network topology, in ad-hoc networks location sensitive information should be updated as a function of network topology. So, it is

important to determine:

1. When to update information?
2. Where to send updates?
3. Which nodes to query for information?

### 1.1 Problem Description and System Model

Let us consider a building on fire. A large number, say  $N$ , of firefighters have been assigned to extinguish the fire. Among the  $N$  firefighters are a small number, say  $n$ , of officers whose job is to collectively manage the entire operation. All the officers need to be able to communicate amongst themselves. Also, an officer can act as a proxy for all ordinary firefighters with whom it can communicate.

To increase the efficiency of the operation, and for the safety of the firefighters, it is important to:

1. track the location of each firefighter, and
2. gather information about the surroundings of each firefighter.

A firefighter is responsible for updating his/her location and state information. Also every firefighter should be able to retrieve the latest information about every other firefighter. Let each firefighter have a small, light-weight wireless communication device with the same range to send and receive information. The devices of all the firefighters collectively form a connected multi-hop wireless network. The officers have additional memory in their devices to maintain the state information. Thus, the officers' devices collectively act as information servers for other firefighters.<sup>1</sup>

The multi-hop wireless network formed by the firefighters may get partitioned into disjoint components. The duration for which the network stays partitioned, the identities of nodes in each partition, and the ways in which partitions merge are non-deterministic.

---

<sup>1</sup>While it is possible to add more memory to every firefighter's device without significantly increasing the weight, broadcasting every update to every node would incur high communication costs and drain the batteries quickly.

Thus, a firefighter should choose the officer(s) to receive its updates without any prior knowledge of future access patterns to its data. Similarly, a querying node should send its query to some officer(s) without any prior knowledge of the mobility pattern of the node whose information it is trying to retrieve. The goal is to maximize the availability of latest state information about nodes in a partitionable network while incurring reasonable communication overheads.

The set of firefighters constitute an ad-hoc network of  $N$  nodes. The  $n$  officers ( $n \ll N$ ) correspond to designated servers in the network. The remaining firefighters are ordinary nodes or clients. There is no *a priori* association between a client and a server, and the network may get partitioned. To mitigate the impact of partitioning we propose to use a data replication scheme for information management. Each update by a client is sent to a subset of servers. Similarly, each query is also sent to a subset of servers. Designing subsets (quorums) that always intersect has been extensively studied in the context of connected networks. The challenge is to construct the query and update subsets so as to maximize the probability of a *hit* (a query returning latest state information about a node) even when the network is partitioned.

The proposed solution has to perform efficiently both when the network is connected and when it is partitioned. Successive updates/queries by a node need not be sent to the same subset of servers. This is because the set of reachable servers changes with time. However, a node could employ recent information about server reachability to determine the subset of servers to which queries and updates are sent.

## 2 Related Work

Davidson, Garcia-Molina and Skeen [4] state that data replication results in high availability of information in the presence of failures and network partitioning. However, if unconstrained updates by different nodes are permitted in multiple partitions of a network, the replica values may diverge. Consequently, queries in different partitions would return inconsistent values. To ensure correctness, one would need to prevent updates in all but one partition. Thus, availability and correctness are mutually conflicting goals. Basically, the causes of error are the write-write conflicts and read-write conflicts among replicas in a partitioned network. A detailed description can be found in [4].

In order to avoid conflicts various mutual exclusion based solutions are proposed. In the voting approach proposed by Gifford [6] each replica is assigned a number of votes. A majority of votes is needed for updates. Also, the sum of votes needed for queries and updates should exceed the total number of votes. This ensures that two updates cannot happen concurrently. Also, if the network is partitioned, the same data item cannot be updated in two different partitions. However,

if the network is partitioned into more than two partitions such that no partition has a majority of votes the update operations are not possible. Thus, data availability is reduced.

To address this problem, *dynamic voting* schemes have been proposed by Paris and Long [13], Jajodia and Mutchler [8], Barbara, Garcia-Molina and Spauster [3], among others. In [13], dynamic voting allows data access to proceed as long as strict majority of current alive physical copies are accessible. If the number of accessible copies is equal to the number of inaccessible copies data accesses are not possible. However, this problem can be resolved by *lexicographic dynamic voting* wherein all nodes are totally ordered by node identity which is used to break the ties [8]. In [3] two dynamic voting schemes have been described, namely the *group consensus* approach and the *autonomous re-assignment* approach. In the group consensus approach the nodes in the majority agree upon a new vote assignment either in a distributed fashion or by electing a leader. Then, the two-phase commit protocol is employed among the majority nodes to install the new votes. In the autonomous reassignment approach each node independently picks a new vote value which can be installed only if the node can obtain a majority of the votes. These dynamic voting schemes require extensive communication between servers for vote reassignment. This is a serious concern for ad-hoc wireless networks with scarce communication bandwidth.

El Abbadi, Skeen and Cristian [1] proposed the *accessible copies* algorithm that employs the *read-one/write-all* protocol. A datum is accessible if a majority of its replicas are present in the same partition. A query on accessible datum is performed by reading its nearest copy. An update writes to all copies of the datum in the partition. Thus, queries and updates can be performed in only one partition. Also, all copies of the datum remain consistent.

Instead of relying on strict majorities, quorum based solutions create a set of subsets of the nodes in the system. Each subset is called a quorum. To perform an update or a query permission from all nodes in at least one quorum is sufficient. The quorums need not be a majority of nodes. Herlihy [7] proposed a *dynamic quorum adjustment* scheme in which quorums can be adjusted on partitioning and mergers. If an operation is unable to progress using one quorum it may be able to make progress by using another more favorable quorum. Such a selection is aided by hints provided by the underlying system about node accessibility.

**Previous Research:** Though the problem we set about solving appears very similar to the replica consistency problem, there are a few subtle differences. The algorithms described above seek to avoid write-write conflicts across partitions, and ensure serializability among all the update transactions. For some weak consistency solutions it is acceptable if read-only

transactions are not serializable with respect to each other.

In the problem at hand, write-write conflicts are simply not possible. A datum is updated by only one node. Let a node perform successive writes to two different quorums of replicas. If we assume that each node has a monotonically increasing local clock, like Lamport's clock [10], the two writes can be easily serialized if each replica stores the data item as a  $\langle \text{value}, \text{writer's timestamp} \rangle$  tuple. Let a query be sent to a set of replicas, returning different values for the datum. The most recent of these values can be identified using the timestamp received with the value.

We employ a variation of the quorum based scheme. As in [1, 7] we, too, use previous knowledge about node accessibility to select sets of replicas that receive queries and updates. However, unlike [1], we do not assume an ideal network in which nodes detect network partitioning immediately.

In most replica consistency solutions, on merger of partitions the replicas in the merging partitions are synchronized so that they have the same value. This incurs very high communication overheads which are not acceptable in low bandwidth wireless networks. The synchronization traffic on mergers may preclude all other updates and queries in the network for some time. So, we intend to adopt an optimistic approach. We do not perform synchronization between copies at the time of mergers. This saves on communication cost without significantly degrading the accuracy of information returned by queries. Moreover, when new links are established we do not have to run tests to determine if hitherto disjoint partitions are merging. Also, the simplicity of the solution is important for its fast and error-free implementation in an actual network.

Our approach is guided by similar principles as the Bayou project [16]. However, there are also some important differences. In Bayou servers engage in *anti-entropy sessions* to exchange information and reach consensus about the order of updates. We do not use such a scheme for reasons described above. Also, in Bayou each datum has a primary server responsible for committing updates. We do not have the notion of primary servers. Also, Bayou is not designed for real-time applications and has tentative updates that are later committed or rolled back. We are targeting real-time applications where inaccuracy of information is sometimes preferable to no information at all.

### 3 Quorum Based Solution

Given a set  $S$  of servers, a quorum system is a set of  $m$  subsets of  $S$ , namely  $S_0, S_1, \dots, S_{m-1}$ , such that:

1.  $\cup_{i=0}^{m-1} S_i = S$ .
2.  $S_i \cap S_j \neq \phi$ , for  $0 \leq i, j \leq m - 1$ .

The sets  $S_i$  are constructed *a priori* and every node knows the membership of these sets. Given  $n$  servers, it is possible to form quorums of size  $O(\sqrt{n})$  [11].

**Information Update:** When a node  $x$  wishes to update some information it timestamps the datum with its local clock value. We assume loosely synchronized clocks such that the time between successive updates is greater than the maximum clock skew. Then, the following actions are performed:

1. Node  $x$  randomly selects a quorum  $S_i$  from the set of quorums and sends UPDATE message, timestamped with its local clock value, to all servers in the quorum.
2. The servers, on receiving the UPDATE message, overwrite their old copy of the data item with the new copy. If they do not have an old copy of that data item, they simply add the information received in the message to their database. Optionally, the servers may also send a positive acknowledgment to  $x$ .

Let an UPDATE message be first sent to quorum  $S_i$ . Later, let another UPDATE, for the same data item, be sent to quorum  $S_j$ . Then, all servers in the set  $S_i - S_j$  have outdated versions of the data item, while all servers in  $S_j$  have the latest version of the data item. An alternative would have been to send a DELETE message to servers in the set  $S_i - S_j$ . While this alternative would reduce the memory requirements, it would increase the communication overheads.

**Information Query:** When node  $y$  wishes to make a query the following actions are performed:

1. Node  $y$  randomly selects a quorum  $S_j$  and sends a QUERY message to all servers in the quorum.
2. When a server receives a QUERY for a datum and has a copy of it, the server sends a REPLY containing the information along with the timestamp associated with the datum. Otherwise, the server sends a NULL reply.
3. Receiving all the REPLY messages,  $y$  selects the value of the datum with the greatest timestamp.

As two quorums always intersect in the fixed network, the set of queried servers is bound to contain at least one server that belonged to the quorum that received the latest update. Hence, each query returns the latest value of the queried data item. Such a query and update strategy has been previously employed for location management in cellular networks [9, 14, 15].

**Why random selection of quorums:** Let a quorum be initially selected by a mobile node such that all servers in the quorum are reachable from that node. Subsequently, if the same quorum is selected whenever the node has to make an update or a query, there is no guarantee that all servers in that quorum would be reachable. So, deterministic quorum selection is not a good idea. Moreover, random selection gives a better chance of load balancing among the servers even if some node suddenly starts generating updates and queries at a higher rate than other nodes.

## 4 Proposed Solution

We need to address the issues of when to trigger updates, and where to send updates and queries so as to mitigate the impact of network partitioning.

### 4.1 When to Update?

Our aim is to propagate information about nodes such that other nodes, on querying for information, get as recent a version as possible. Had there been a means to predict network partitioning, updates could be performed just before such partitions occur. However, in the absence of such an oracle, a node may have to observe changes in network topology and make guesses about reachability to/from other nodes. We propose four different policies to trigger updates in increasing order of sophistication:

1. In the *time-based* strategy the time between successive location update attempts by a node is exponentially distributed, with a mean of  $t$  units. For our simulations we set  $t$  to 1.0 unit.
2. The *time and location-based* strategy is similar to the time based strategy. However, in this strategy a node remembers its location when it last sent an update. If the node's location unchanged since the last successful update, no update requests are sent.
3. In the *absolute connectivity-based* strategy a node sends an update when a certain *pre-specified number* of links incident on it have been established or broken since the last update.
4. In the *percentage connectivity-based* strategy an update is triggered when a *pre-specified percent* of the links incident on it have changed since the last update.

The last two update strategies are motivated by the feeling that in ad-hoc networks the frequency of updates should be a function of the dynamism exhibited by the network. In [2], Bar-Noy, Kessler and Sidi showed that movement-based and distance-based updates performed better than time-based updates in a cellular network. As there is no notion of cells in ad-hoc networks the movement-based strategy is not applicable. Also, the relative position of nodes (and the resultant topology) is a more important piece of information than their absolute location or absolute distance moved. Hence, distance-based updates may have limited usefulness in ad-hoc networks. The absolute connectivity- and the percentage connectivity-based approaches are an attempt to capture the relative change in topology.

### 4.2 Where to Update and Query?

There is an implicit assumption in the quorum based scheme that all the servers in the selected quorum

are reachable from the node that sends the update or query. This assumption is valid for cellular networks where base stations and servers maintaining location information belong to the wired backbone network and are reachable from each other virtually all the time. However, in an ad-hoc network there is a possibility that the network gets partitioned as described earlier in the context of firefighters in a building.

Let node  $x$  select quorum  $S_i = \{s_{x1}, s_{x2}, \dots, s_{xn}\}$  to update the value of a data item. If some elements of  $S_i$  are in a different partition from  $x$  at the time of the update. They do not receive the update. Only the servers in  $S'_i \subseteq S_i$  receive the update. Subsequently, let another node  $y$  select a quorum  $S_j = \{s_{y1}, s_{y2}, \dots, s_{yn}\}$  to query the value of the data item.

Some elements of  $S_j$  may be in a different partition from  $y$  at the time of the query. So, only servers in  $S'_j \subseteq S_j$  receive the query. Thus, there is a possibility that the query may not return any information, or return stale information. This is due to the fact that even though  $S_i \cap S_j \neq \phi$ , it is possible that  $S'_i \cap S'_j$  is empty.

One possible solution would be to send the query once again to another quorum  $S_k$  hoping that  $S'_i \cap S'_k$  is non-empty. However, this solution is not acceptable for two reasons:

1. It increases the communication overheads in a bandwidth poor network, without any guarantees of returning the latest information.
2. The querying node may not be able to realize that it has stale information. If at least one server returns a non-NULL reply the node accepts the value with the highest timestamp. There may be no way for the querying node to know that there exists at least one copy of the data item with a higher timestamp, somewhere in the network.

To alleviate the problem of query failures we need to make informed selection of quorums at the time of updates and queries. The idea is to select quorums so that the sets  $S_i - S'_i$  and  $S_j - S'_j$  are as small as possible. The smaller these sets, the greater the probability of the set of reachable queried servers intersecting with the set that received the latest update.

**Disqualified List:** We propose to use a heuristic to select servers for updates and queries. Node  $x$  maintains a *disqualified list*,  $DQL_x$ , containing servers that  $x$  believes to be unreachable. Note that  $DQL_x$  represents  $x$ 's perception of unreachability. There may be servers that are not reachable from  $x$ , yet not in  $DQL_x$ . Similarly, there may be servers that are reachable from  $x$ , but also in  $DQL_x$ .

The instance of the heuristic running at node  $x$  performs the following tasks:

1. Maintenance of  $DQL_x$ .
2. Selection of servers for updates/queries on the basis of  $DQL_x$ 's composition.

- Communication with selected servers for updates and queries.

One of the goals of the heuristic is to be able to update the disqualified list as part of the query and update operations, without incurring any extra communication overheads and delays.

### 4.3 Heuristic

Initially, *disqualified list*,  $DQL_x$  is empty. We propose three strategies to select servers for updates and queries using information in the disqualified list, namely: (i) *Select\_then\_Eliminate (STE)*, (ii) *Eliminate\_then\_Select (ETS)*, and (iii) *Hybrid* strategies.

**Select\_then\_Eliminate (STE) Strategy:** In this strategy, node  $x$  randomly selects a quorum  $S_i$ . From the set of servers  $S_i$ ,  $x$  eliminates those in  $DQL_x$ . Let  $S'_i = S_i - DQL_x$ . Update/query messages are sent to all servers in  $S'_i$ .

Node  $x$  expects to receive a reply from every server to which a message is sent within time  $T_{timeout}$ . An update is considered to be successful if at least one server sends an acknowledgment of receipt of the update message. In the case of queries, if one or more servers send non-NULL timestamped information, the received copy with the greatest timestamp is selected, and the operation is considered to be successful. This process is repeated until there is success.

If a reply is not received from a server  $s$  within  $T_{timeout}$ , node  $x$  concludes that  $s$  is not reachable and adds  $s$  to  $DQL_x$ . Once a server  $s$  has been added to  $DQL_x$  it stays in it for a *disqualification duration*,  $\delta_{DQL}$ . At the end of  $\delta_{DQL}$ ,  $s$  is removed from  $DQL_x$ . The value of  $\delta_{DQL}$  is a parameter of the heuristic and should be determined on the basis of the connectivity characteristics of the network.

If all the nodes are concentrated in a small area, or if the nodes have a large wireless range, the probability of network partitioning is low. Even if the network gets partitioned, the mean time before the partitions merge with each other is also expected to be small. In such cases having a small value of  $\delta_{DQL}$  would be advisable.

If a much smaller value of  $\delta_{DQL}$  is selected, a server may be removed from  $DQL_x$  when it is still unreachable from  $x$ . So,  $x$  will end up trying to probe servers that are still not reachable, increasing the communication overheads. On the other hand, if a very large value of  $\delta_{DQL}$  is selected, some previously unreachable servers will not be probed for a long time even after they are once again reachable. This may result in unavailability of information, reduced utilization of some servers, and excessive load on other servers.

**Eliminate\_then\_Select (ETS) Strategy:** In this strategy, node  $x$  first *eliminates* all quorums that have at least one node in  $DQL_x$ . One of the remaining quorums is randomly selected and messages are sent to

servers in this quorum. If at least one server sends an acknowledgment in the case of an update, or a non-NULL value in the case of a query, the operation is said to succeed. Let some servers not respond within  $T_{timeout}$ , while all others send a NULL reply. Then the servers that did not respond are added to  $DQL_x$  for  $\delta_{DQL}$  time. Also, quorums containing at least one server in the expanded  $DQL_x$  are eliminated, and one of the remaining quorums is selected for the update/query operation. This process is repeated until the update/query operation succeeds. If all quorums get eliminated before there is success, the update/query operation is said to fail.

**Hybrid Strategy:** The hybrid strategy uses ETS for updates and STE for queries. The idea is to maximize the number of servers receiving the updates with the hope that this will result in more accurate information retrieval by queries. The motivation for the hybrid strategy will become clear in the following paragraphs.

**Comparison of STE, ETS and Hybrid Strategies:** STE tries to maximize availability of information, while ETS tries to maximize accuracy of queries at the expense of availability. In the ETS strategy an attempt is made to identify a quorum with no server in the disqualified list. This reduces the number of quorums available for update or query while maximizing the number of servers receiving the query/update. In the STE strategy all the quorums are available for update or query thus increasing availability.

However, the increased availability of STE may actually reduce the accuracy of information. One of the concerns of a transaction processing system for partitioned networks is *correctness within partition*. Correctness within partition means that the queries in a partition should return the values stored by the latest successful update in that partition to the same data item. When updates are sent to fewer servers, the probability of queries hitting at least one server with the latest information is also lower. This probability is further reduced as the queries may also be sent to fewer servers. Even though the query and update sets intersect, their intersection may belong to another network partition. STE seems to run counter to the desired goal of minimizing the size of the set  $S_i - S'_i$ , where  $S'_i$  is the set of reachable servers of quorum  $S_i$ . Hence, the probability of obtaining stale information is greater.

Let us consider the situation when  $DQL_x$  accurately represents the servers that are not reachable from node  $x$ . Then, due to the reason stated above, queries in the STE strategy may return stale information or no information at all. However, in the ETS strategy, due to the way the quorums are eliminated *a priori*, all servers in the quorums selected for an update and a subsequent query belong to the same partition, and are indeed reachable. Hence, correctness within partition is guaranteed for ETS in this situation.

However, the ETS strategy provides a lower availability of data than STE. Once again, let  $DQL_x$  accurately represent reachability information, and let updates or queries succeed in one partition. This means that all servers of at least one quorum are within that partition. Due to the non-empty intersection property of quorum pairs, all quorums will be eliminated in other partition(s) by the ETS strategy. So, queries and updates cannot be performed in those partition(s).

If availability of stale information is preferable to no information at all, STE may be preferred over ETS. Of course, if  $\delta_{DQL}$  is different from the elapsed time between partition detection and merger,  $DQL_x$  will be inaccurate, and even ETS may return stale information while STE may have reduced availability.

The hybrid strategy attempts to increase the accuracy of future queries by performing updates using the ETS strategy. At the same time it strives to maximize the availability of information during queries by using the STE strategy. It is hoped that performing updates at a greater number of sites will actually increase the probability of a query returning fresh information, even though the queries are sent to fewer servers. This may be especially useful in systems where there are many more queries than updates. Savings in communication overheads during the more frequent queries may more than offset for the extra communication overheads incurred by the less frequent updates.

#### 4.4 Communication and Storage Overheads

A copy of every data item, fresh or stale, may be resident at each server. So, every data item requires  $O(n)$  storage. This can be reduced if, at the time of update, the node also sends delete messages to servers that were in its previous update quorum, and are not in its present update quorum. However, this will require additional communication.

Given  $n$  servers, there exist quorum formation schemes that give quorums of size  $O(\sqrt{n})$  [11]. So, even if an update or query requires a constant number of retries before success, the communication complexity is  $O(\sqrt{n})$ . Note that  $n$  is usually much smaller than the total number of nodes  $N$ . Hence, the communication and storage overheads are reasonable.

### 5 Simulation Experiments

We conducted simulation experiments using the CSIM18 simulation engine [12]. We compared the performance of the STE,ETS and Hybrid strategies with the simple quorum based strategy that does not account for partitioning and does not use the disqualified list. In the simple quorum based scheme, a node continues to randomly select a quorum and send queries/updates until either the operation is performed successfully, or all quorums have been tried without success. In the latter case, the operation is said to fail.

We assumed a system composed of 100 mobile nodes. Of these, 25 nodes were assumed to act as

servers. Initially, the 100 nodes are randomly distributed in a square region with edges of 1000 length units. During the simulation experiment all the nodes are allowed to move only within this square.

In order to model mobility of nodes we cycle through the following two steps: (i) the duration for which a node stays at a location is exponentially distributed with mean 0.5 time units, (ii) at the end of this duration, the node randomly selects another location within the square region that is at most 75 length units away from its current location, and instantaneously moves to the new location.

A wireless link is assumed to be present between a pair of nodes if they are within 150 length units of each other. Each time a new link is established, or an old link is broken, the simulator executes Floyd-Warshall's algorithm [5, 17] to determine the reachability of nodes, and the presence of partitions in the network. Note that this computation of reachability *does not have to be performed* by nodes as part of their update and query function. In subsequent experiments we vary the wireless range to values other than 150 and measure the impact of partitioning on the information dissemination strategies.

The 25 servers are *logically* arranged in a  $5 \times 5$  square grid, and 25 quorums are formed: each quorum is composed of the union of a row and a column of servers in this logical grid. Thus, each quorum consists of 9 servers. The quorums are assigned distinct sequence numbers in the range 1 – 25. Every pair of quorums has two servers in common.

The update and query operations work on location dependent information.<sup>2</sup> As each node has its unique location, there are 100 distinct pieces of information. The time between successive queries by a node is exponentially distributed, with a mean of 0.5 time units. A querying node randomly selects one of the 100 nodes as the node whose location it wishes to query. Query messages are sent to servers based on the strategy in use: ETS, STE, or Hybrid. If a node discovers that a server is unreachable, the node places the server in its disqualified list for a period  $\delta_{DQL} = 0.25$  time units. We also varied the value of  $\delta_{DQL}$  to measure its impact on server availability, utilization and load balancing.

In each simulation run, no data collection was performed until the first 10,000 queries were finished (successfully or unsuccessfully). This was done to eliminate the impact of any initial transient effects. Subsequently, data collection was performed until the next 100,000 queries were completed. Each data point in the graphs represents the mean value obtained from five simulation runs with identical input parameters, but different random number generation seeds.

---

<sup>2</sup>Note that each firefighter sends its location and the status of the fire, etc. in his/her location as part of the update.

## 5.1 Simulation Results

We measured various performance characteristics of the ETS, STE and Hybrid strategies versus the simple quorum based strategy that does not use the disqualified list. For updates the *time and location-based* policy was employed.

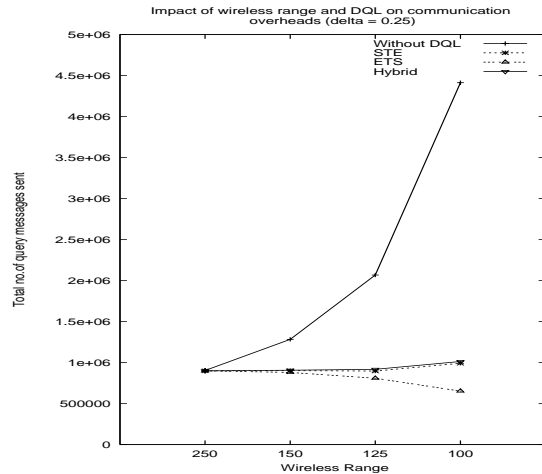
**Query success and freshness:** The impact of wireless range of nodes on the number of successful queries and the number of queries returning *fresh* information (information stored by the latest update for that datum) is shown in Figure 1. When the wireless range is equal to 250, or 150 length units almost all the queries succeed regardless of the strategy employed, as seen in the graph on the left hand side of Figure 1. This is because for these values of wireless range there are very few partitions in the network. Even when partitions do occur, the partitions get merged quickly. However, there is a significant drop in the number of successes for wireless range of 100 length units. This is due to increased probability of network partitioning. The ETS strategy suffers the most due to increased partitioning probability. This is due to the fact that when the network is partitioned the ETS strategy may not be able to find any quorum because at least one server of every quorum is in the querying node’s disqualified list.

When the wireless range is equal to 250 or 150 most of the queries return latest information. But, when the range is reduced to 100, the number of queries returning the latest information is significantly reduced. Once again, this is due to increased likelihood of network partitioning. Moreover, the *fraction* of successful queries that yield the latest information is much higher for the ETS strategy than the STE and simple quorum based strategies, as observed by looking at the graphs on the left as well as on the right in Figure 1. This is consistent with our prior analysis of the STE strategy emphasizing availability over accuracy, and ETS emphasizing accuracy over availability.

In Figure 1, the value of  $\delta_{DQL}$  is set to 0.25. Similar results were obtained for other values of  $\delta_{DQL}$  as well. Among the four strategies, ETS has fewest queries returning the latest information. Also, ETS is significantly poorer than the other strategies in terms of query success. This leads us to believe that ETS is not a good strategy for information dissemination.

**Communication Overheads:** Figure 2 shows that the ETS, STE, Hybrid, and simple quorum based strategies incur comparable communication costs for larger values of wireless range. However, for wireless range of 100 length units when the probability of network partitioning is more, the simple quorum based strategy incurs very high communication cost. For the same wireless range the communication costs are comparatively low for the STE, Hybrid, and ETS strategies, as was expected. The ETS strategy has the lowest cost because quite often it cannot even find a quorum to

send the queries to. The simple quorum based strategy has the highest overheads because the nodes try to send their queries to even those servers that are not reachable. Also, when there is a failure, the nodes keep trying other quorums until either some information is retrieved or all quorums have been tried.

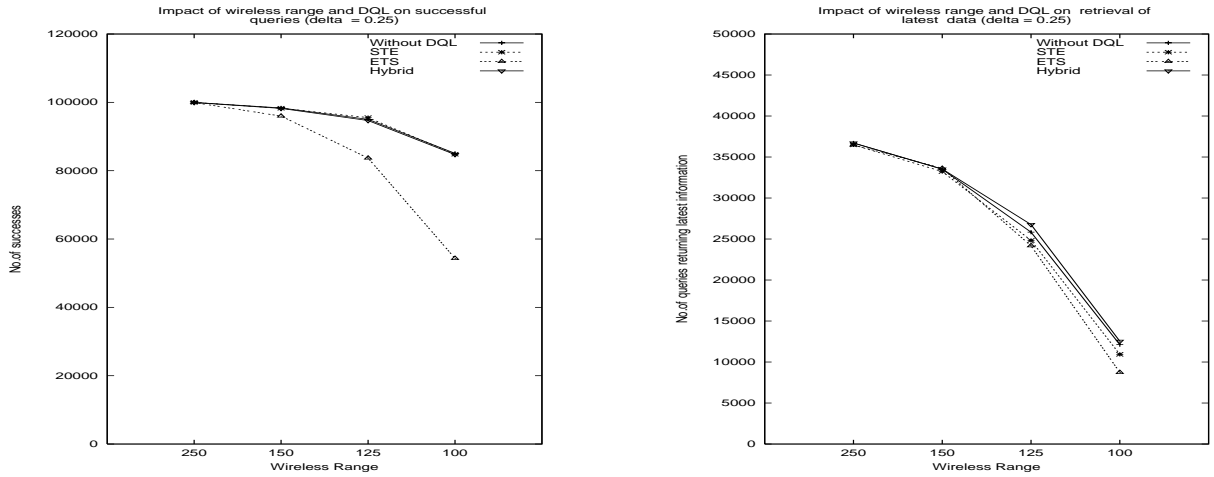


**Figure 2. Impact of wireless range and quorum selection policy on communication overheads.**

**Load Distribution:** We measured the distribution of load (total number of messages received by a server) on all servers when  $\delta_{DQL}$  was set to 0.25 and the wireless range was 100. In all the four strategies different servers receive a different number of query messages, with the simple quorum based scheme (without DQL) experiencing the maximum traffic and also the maximum variance in traffic. The reason for the load to be unevenly distributed is that the network gets partitioned. Due to this every server is no longer equally likely to receive the query messages. However, when the wireless range was increased to 150 and 250 distance units network partitioning was a rare occurrence and the load was evenly distributed across all servers.

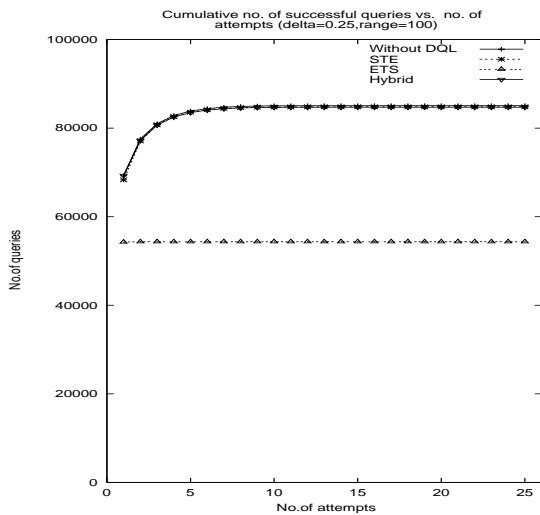
**Number of attempts:** Figure 3 shows the cumulative number of successful queries versus the number of query attempts, where a query attempt corresponds to selection of a quorum and sending messages to reachable servers in that quorum. The STE, Hybrid, and the simple quorum based strategies require almost same number of attempts for a success. The ETS strategy requires fewer attempts because the number of quorums available after eliminating those that contain servers in *DQL* is less. Similar results were obtained for other values of wireless range and  $\delta_{DQL}$ .

Most of the successful queries require only one attempt. As the curves plateau after about four to five attempts it can be safely said that if a query has not



**Figure 1. Impact of wireless range and quorum selection policy on the number of successful and accurate queries.**

succeeded after four to five attempts, there is no point in trying further. The small increase in the number of successes may not be worth the significant increase in the communication cost incurred in doing so.



**Figure 3. Number of attempts required by a successful query.**

**Impact of Update Policy:** Figure 4 shows that the update policy can have a significant impact on the number of information updates performed by nodes during the lifetime of the simulation. As the Hybrid strategy performs significantly better than ETS, and has lower communication overheads than simple quorum based strategy, we only studied the impact of the update policy on the Hybrid strategy. For the absolute connectivity-based policy, an update is triggered when *five* edges incident on the node have changed since the last update. For the percentage connectivity-based

strategy, an update is triggered when 20% of the links have changed since the last update.

Maximum number of updates were performed when the percentage connectivity-based policy was employed. This is because with 100 nodes scattered over the entire region, the network is very sparsely connected. So, only a few links need to change to exceed the 20% threshold. For the same reason, it takes a long time for five links to change. Hence, the absolute connectivity based policy triggers very few updates. However, in spite of the significantly fewer updates, the number of successful, fresh and stale queries for the absolute connectivity-based update policy is comparable to that of the other policies.

However, before we rush to judgment, let us also consider the error in the location information returned by all queries (fresh as well as stale). Note that a query returns the location of the queried node at the time that node updated its location and state information. Error indicates the distance between the actual location of the queried node at the time of the query, and the location information returned to the querying node. Recalling our example, a large error in location determination will mislead the firefighters about the position of their colleagues, and seriously jeopardize both their safety and the success of their operation. Figure 4 shows the number of queries on the y-axis with location error no more than the value on the x-axis. The percentage connectivity-based policy has the best performance with about 20,000 queries returning location information that is within 25 distance units of the actual location of the queried node. About 40,000 queries return location information that is no more than 100 distance units away from the actual location of the queried node, and so on. The graph also indicates that in spite of significantly fewer updates, the error in location determination for the absolute connectivity-based

policy is comparable to other policies.

*This seems to indicate that, at least for sparse networks, the absolute connectivity-based policy is most suitable.* Even though the other policies perform more updates, due to the highly partitioned nature of the network most of the updates are visible to a small subset of querying nodes. A majority of nodes in the other partitions do not see the updates and continue to return outdated location information.

**Impact of Network Density:** We increased the number of nodes in the network from 100 to 200, while keeping the number of servers unchanged at 25. As can be seen in Figure 5, with increased density (due to the presence of 200 nodes in the same area), network connectivity improved and there were fewer partitions. As a result, the error in location information returned by queries was reduced. This is because the servers receiving the latest update were reachable from more nodes in the network.

Also, with increased connectivity the absolute connectivity-based update policy triggered more location updates. Greater the number of links incident on a node, the less time it takes for five links to change since the last update. Still the absolute connectivity-based update policy performs fewer updates than the time-based and percentage connectivity-based update policies, and only slightly more updates than the time and location-based update policy. Just as in the case of sparse networks, the number of successful, fresh and stale queries are comparable for all the four policies.

Thus, our simulation experiments seem to indicate that *the absolute connectivity-based update policy* is the most suitable among the four policies considered. When the network is sparse and updates are going to have less of an impact it triggers fewer updates. When the network is dense, the topology is changing more rapidly, and updates can be seen by more nodes it triggers a greater number of updates.

**Impact of disqualification period:** For values of  $\delta_{DQL}$  greater than 0.25 (we tried values of 0.50 and 1.00) the simple quorum based strategy (without DQL) exhibited a higher number of successes than the ETS, STE and Hybrid strategies. This may be due to be due to the fact that with a larger value of  $\delta_{DQL}$ , if a server is found to be unreachable, a node does not send updates and queries to that server for quite some time even after the server has once again become reachable.

A  $\delta_{DQL}$  value lower than 0.25 yielded the same performance in terms of query success, freshness, and staleness. However, the communication overheads increased. This seems to be due to the possibility that nodes may resume sending updates and queries to servers prematurely: the servers may still be in a different partition.

## 6 Conclusion

A scenario for the use of an ad-hoc network of mobile nodes was presented, where the network can get partitioned and reconnected several times. The problem of information dissemination in such situations was formalized. In order to increase the availability of information, data replication was considered. Three quorum based strategies STE, ETS and Hybrid (ETS for updates, STE for queries) have been proposed. The STE strategy optimizes availability of information, whereas ETS tries to maximize accuracy of information at the expense of availability. This is also indicated by the simulation results. The Hybrid strategy exploits the advantages of both the strategies. Four policies to trigger updates were considered. Of these the absolute connectivity-based policy was found to have the best performance. The absolute connectivity-based policy triggers more updates in dense networks where the impact of updates will be visible to a greater number of nodes. In sparse networks this policy triggers fewer updates as the updates have a lower potential of being visible to other nodes.

## References

- [1] A. El Abbadi, D. Skeen, and F. Cristian. An Efficient Fault-Tolerant Algorithm for Replicated Data Management. In *Proceedings of the 5<sup>th</sup> ACM SIGACT-SIGMOD Symposium on the Principles of Database Systems*, pages 215–229, 1985.
- [2] A. Bar-Noy, I. Kessler, and M. Sidi. Mobile Users: To Update or not to Update? In *Proceedings of IEEE INFOCOM*, pages 570–576, 1994.
- [3] D. Barbara, H. Garcia-Molina, and A. Spauster. Increasing Availability Under Mutual Exclusion Constraints with Dynamic Vote Reassignment. *ACM Transactions on Computer Systems*, 7(4):394–426, November 1989.
- [4] S.B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in Partitioned Networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.
- [5] R.W. Floyd. Algorithm 97 (SHORTEST PATH). *Communications of the ACM*, 5(6):345, 1962.
- [6] D.K. Gifford. Weighted Voting for Replicated Data. In *Proceedings of the 7<sup>th</sup> Symposium on Operating Systems Principles*, pages 150–162. ACM, 1979.
- [7] M. Herlihy. Dynamic Quorum Adjustment for Partitioned Data. *ACM Transactions on Database Systems*, 12(2):170–194, June 1987.
- [8] S. Jajodia and D. Mutchler. Integrating Static and Dynamic Voting Protocols to Enhance File Availability. In *Proceedings of the 4<sup>th</sup> International Conference on Data Engineering*, pages 144–153. IEEE, 1988.
- [9] G. Krishnamurthi, M. Azizo glu, and A.K. Somani. Optimal Location Management Algorithms for Mobile Networks. In *Proceedings of The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, pages 223–232, October 1998.

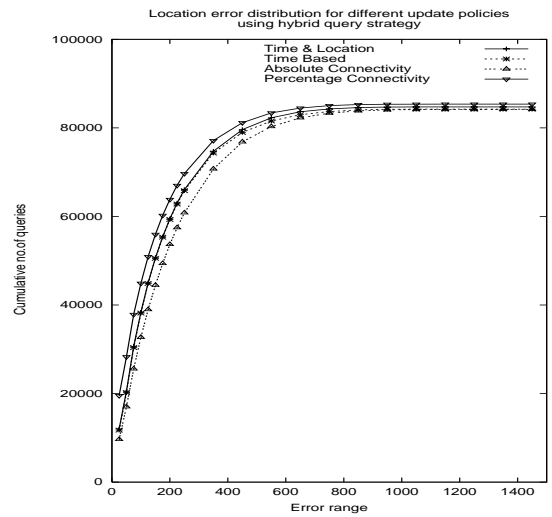
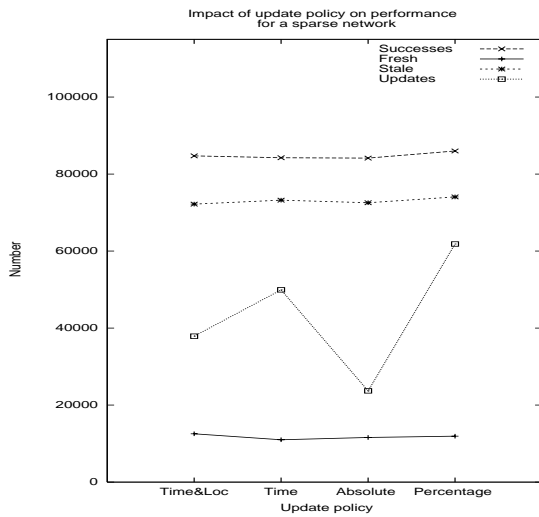


Figure 4. Impact of update policy on freshness and accuracy of information returned by queries.

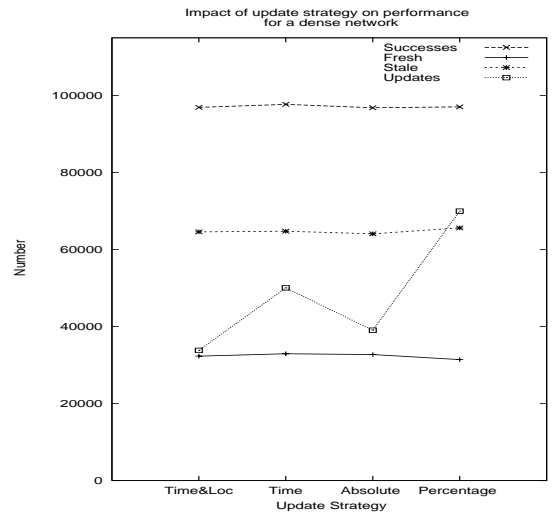
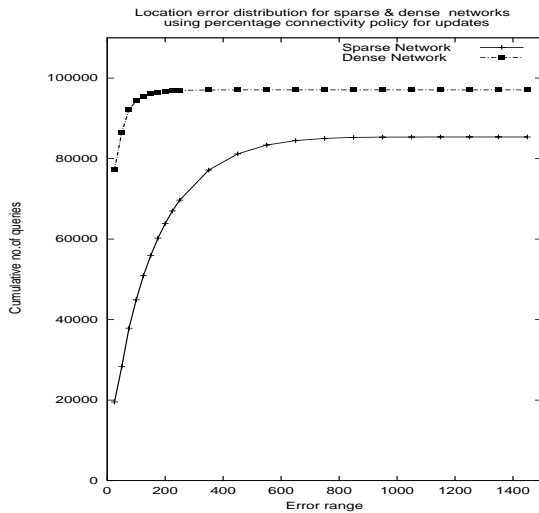


Figure 5. Impact of network density on freshness and accuracy of information returned by queries.

- [10] L. Lamport. Time, Clocks and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7):558–565, July 1978.
- [11] M. Maekawa. A  $\sqrt{N}$  Algorithm for Mutual Exclusion in Decentralized Systems. *ACM Transactions on Computer Systems*, pages 145–159, May 1985.
- [12] Mesquite Software, Inc., 3925 West Braker Lane, Austin, TX 78759. *CSIM18 Simulation Engine*, 1998.
- [13] J.-F. Paris and D.D.E. Long. Efficient Dynamic Voting Algorithms. In *Proceedings of the 4<sup>th</sup> International Conference on Data Engineering*, pages 268–275. IEEE, 1988.
- [14] R. Prakash, Z. J. Haas, and M. Singhal. Load Balanced Location Management for Mobile Systems using Dynamic Hashing and Quorums. Technical Report UTDCS-05-97, The University of Texas at Dallas, October 1997.
- [15] R. Prakash and M. Singhal. A Dynamic Approach to Location Management in Mobile Computing Systems. In *Proceedings of the 8<sup>th</sup> International Conference on Software Engineering and Knowledge Engineering (SEKE'96)*, pages 488–495, Lake Tahoe, Nevada, June 1996.
- [16] D. B. Terry, M. M. Theimer, K. Peterson, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of SIGOPS'95*, pages 172–183, 1995.
- [17] S. Warshall. A Theorem on Boolean Matrices. *Journal of the ACM*, 9(1):11–12, 1962.