Balancing the Tradeoffs between Data Accessibility and Query Delay in Ad Hoc Networks*

Liangzhong Yin and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {yin, gcao}@cse.psu.edu

Abstract

In mobile ad hoc networks, nodes move freely and link/node failures are common. This leads to frequent network partitions, which may significantly degrade the performance of data access in ad hoc networks. When the network partition occurs, mobile nodes in one network are not able to access data hosted by nodes in other networks. In this paper, we deal with this problem by applying data replication techniques. Existing data replication solutions in both wired or wireless networks aim at either reducing the query delay or improving the data accessibility. As both metrics are important for mobile nodes, we propose schemes to balance the tradeoffs between data accessibility and query delay under different system settings and requirements. Simulation results show that the proposed schemes can achieve a balance between these two metrics and provide satisfying system performance.

1. Introduction

Portable computers and wireless networks are becoming widely available, which enables users to remain connected to the Internet while moving around. For example, with cellular network techniques such as General Packet Radio Service (GPRS), mobile users can connect to the Internet while moving. However, mobile users may want to communicate with each other in situations where such kind of fixed infrastructure is not available. For example, a group of emergency rescue workers may need to form a network after an earthquake, or a group of soldiers may need to communicate during a military operation. In such circumstances, a collection of mobile nodes with wireless network interfaces may form a temporary network without the aid of any established infrastructure or centralized administration. This type of network is known as wireless ad hoc networks [8]. In ad hoc networks, direct communication between any two nodes is possible when they are within the communication range of each other, in which case we say that these two nodes are neighbors. Otherwise, the nodes communicate through multi-hop routing [14].

In ad hoc networks, since mobile nodes move freely, disconnections may occur frequently. If a network is divided into two partitions due to the movement of mobile nodes, mobile nodes in one of the partitions cannot access the data held by the mobile nodes in the other partition. Thus, data accessibility in ad hoc networks is lower than that in the conventional fixed networks. Data replication has been widely used to improve data accessibility in distributed systems, and we want to apply this technique to ad hoc networks. By replicating data at mobile nodes, data accessibility can be improved because there are multiple replicas in the network and the probability of finding one copy of the data is high. Further, data replication can also reduce the query delay, since mobile nodes can get the data from some nearby replicas.

Generally speaking, data replication can increase the data accessibility and reduce the query delay if there are plenty of storage space in the mobile nodes. However, mobile nodes only have limited storage space, bandwidth and power, and hence it is impossible for one node to hold all the data considering these limitations. Therefore, it is important for mobile nodes to cooperate with each other; i.e., contribute part of their storage space to hold data of others. When a node only replicates part of the data, there will be a tradeoff between query delay and data accessibility. For example, replicating most data locally can reduce the query delay, but it reduces the data accessibility since many nodes may end up replicating the same data locally, while some data are not replicated by anyone. To increase the data accessibility, nodes should not replicate the same data that

* This work was supported in part by the National Science Foundation (CAREER CCR-0092770 and ITR-0219711).

neighboring nodes already replicate. However, this solution may increase the query delay since some nodes may not be able to replicate the most frequently accessed data, and have to access it from neighbors. Although the delay of accessing the data from neighbors is shorter than that from the data owner, it is longer than accessing the data locally.

In this paper, we propose data replication schemes that address both the query delay and the data accessibility. As both metrics are important for mobile nodes, our schemes need to balance the tradeoffs between data accessibility and query delay under different system settings and requirements. Simulation results show that the proposed schemes can achieve a balance between these two metrics and provide satisfying system performance.

The rest of the paper is organized as follows. In the next section, a brief review of the related work is presented. Section 3 gives some preliminaries of the research. Section 4 describes the proposed schemes in detail. Section 5 evaluates the proposed schemes through extensive simulations. Section 6 concludes this paper.

2. Related Work

Data replication has been extensively studied in the Web environment [4, 10, 16, 21]. The goal is to place some replicas of web servers among a number of possible locations so that the query delay is minimized. In the Web environment, links and nodes are stable. Therefore, the performance is measured by the query delay, and data accessibility is not a big issue. These replication schemes work at the whole database level. That is, the whole database is replicated as a unit to one or more locations. It is more complex when replication is studied at data item level, i.e., how to replicate data items to various nodes with limited memory spaces.

Data replication has been studied in distributed database systems [4, 18, 19]. In such systems, nodes that host the database are more reliable and less likely to fail/disconnect than that in ad hoc networks. Therefore, a small number of replicas can be used to provide high accessibility. However, in ad hoc networks, node/link failure occurs frequently, and data accessibility becomes an important issue.

Several data replication schemes have been proposed in wireless networks [7, 15]. These schemes assume an environment where mobile nodes access database at sites in a fixed network, and create replicas of data on the mobile nodes because wireless communication is more expensive than wired communication. Their major concern is to keep the consistency between the original data and its replicas. The difference between these schemes and ours is that our schemes are proposed in multi-hop ad hoc networks. There is no central server and data are fully distributed at mobile nodes.

Hara [5] proposed data replication schemes in ad hoc networks. These schemes are based on the intuition that to improve data accessibility, replicating the same data near neighboring nodes should be avoided. However, this intuition may not be valid when the link failure probability is taken into consideration. More detailed discussion will be presented in Section 4.1. Another drawback is that it only considers the accessibility, without considering the query delay. Both drawbacks will be addresses in this paper to provide better data replication.

Some other researchers also addressed data access issues in ad hoc networks where network partition may occur. Karumanchi *et. al.* [9] and Luo *et. al.* [11] adopted the quorum based system to improve the data availability in ad hoc networks. Nuggehalli *et. al.* [13] proposed a POACH algorithm to deal with the cache placement problem so that the query delay and energy consumption can be reduced, but they did not consider link failure and the data are always available. Wang and Li [17] proposed schemes to deal with network partitions due to node movement by duplicating services in the network. Their schemes can provide guaranteed service with minimal number of duplicated services. Different from these previous works, we study the tradeoffs between the query delay and data accessibility.

Besides data replication, caching can also be used to improve data accessibility and reduce the query delay [2]. In [20], we proposed a cooperative cache based data access framework, which allows the sharing and coordination of cached data among multiple mobile nodes. In this solution, after a node sends a data request to the data owner, the data owner sends the data back. Since the data may go through multi-hops before reaching the requester. Intermediate nodes may cache the data or the path to the data. Later, if some other nodes request for similar data, and the request goes through any of these intermediate nodes, they can return the data or the path to the data. As the number of hops reduces, the query delay also reduces. Although the proposed solution can reduce the query delay, there is a limitation on how much they can achieve. Generally speaking, these schemes are passive approaches, since the data are only cached after some nodes start to use it. To further increase the data accessibility and reduce the query delay, we study proactive data replication techniques in this paper.

3. Preliminaries

3.1. System Model

The following notations are used in this paper.

- m: the total number of mobile nodes.
- N_i: mobile node i.
- n: the total number of data items in the database.

- d_i: data item i.
- s_i: the size of d_i.
- C: the memory size of each mobile node for hosting data replicas.
- f_{ij}: the link failure probability between node N_i and N_j.
- a_{ij}: the access frequency of node N_i to d_j.

The ad hoc network studied in this paper has a total of m mobile nodes, $N_1, N_2, ..., N_m$. A database of n items $d_1, d_2, ..., d_n$ is distributed in the network. At any given time, the link between N_i and N_j has a probability of f_{ij} to fail. f_{ij} is equal to f_{ji} as we assume symmetric link conditions. The failed links may cause network partitions. Queries generated during the network partition time may fail because the requested data item is not available in the partition the requester belongs to. The access frequency of node N_i to d_j is a_{ij}. Each mobile node maintains some amount of data locally and is called the original owner of these data. Each data item has one and only one original owner. For simplicity, we assume that data are not updated, and similar techniques used in [6] can be used to extend the proposed scheme to handle data update. To improve data accessibility, these data may be replicated to other nodes. Because of limited memory size, each mobile node can only host $C, C < n$, replicas beside its original data. When a mobile node N_i needs to access a data item d_j, N_i first searches its local memory. If N_i cannot find a copy of d_j in the local memory, N_i communicates with its reachable nodes (through one-hop or multi-hop links) to get d_j. If the requesting node cannot communicate with any of the nodes that have d_j, d_j is considered to be not *accessible* to N_i. *Data Accessibility* is defined as the number of successful data accesses over the total number of data accesses.

3.2. Problem Analysis

To simplify the problem, we first consider the optimization problem with only one performance metric. In this case, the goal of data replication is to allocate n data items among m mobile nodes so that a certain performance metric (data accessibility, query delay, etc.) is optimized.

The optimal solution for this problem is not very practical due to the computational complexity. Let us use the optimization of the data accessibility metric as an example. It is obvious that replicating data improve the data accessibility. The performance improvement can be viewed as the profit of data replication. However, the profit of replicating d_i at a certain node N_j is affected by nodes' access frequency to d_i, the network topology, and link failure probability. Therefore, the profit of d_i is different at different nodes. Furthermore, the profit is also affected by previous replication of d_i in the network. That is, the profit of allocating the first copy of d_i in the network at N_j is different from the case when allocating d_i at N_j but d_i has already been replicated once, twice, or more at some other node(s). We can see that at one given node, the profit of a data item d_i can take $(m-1)!$ different values, where m is the number of nodes, depending on the replication of d_i at other nodes.

To further simplify this problem, let us assume that the profit of replicating d_i at node N_j, denoted as $p(i,j)$, is not affected by the replication of d_i at other nodes. This means that the profit of replicating d_i can take m different values. This is a special case of the *Generalized Assignment Problem (GAP)* [3, 12], which can be defined as:

INSTANCE: A pair $(\mathcal{B}, \mathcal{S})$, where $\mathcal{B}$ is a set of m bins, and $\mathcal{S}$ is a set of n items. Each bin $j \in \mathcal{B}$ has a capacity of $c(j)$, and for each item i and bin j, we are given a profit $p(i,j)$ and a size $s(i,j)$.

OBJECTIVE: Find a subset $U \subseteq \mathcal{S}$ of maximum profit such that U has a feasible packing in $\mathcal{B}$.

In our case, the size of d_i is fixed and the bin size is identical (C). However, Chekuri and Khanna [3] proved that even for the following special case, where

- each data item takes only two distinct profit values,
- each data item has an identical size across all bins and there are only two distinct item sizes, and
- all bin capacities are identical,

the problem is still APX-hard, which means that there exists some constant $e > 0$ such that it is NP-hard to approximate the problem within a factor of $(1 + e)$.

The analysis above shows that the data replication problem we studied is extremely hard in terms of the computational complexity. Even for a simplified version of the problem, it is still NP-hard to approximate the problem. Therefore, instead of trying to find a complex algorithm that is totally not practical to solve or approximate the problem, we present heuristics that can provide satisfying performance with very small computation overhead.

4. The Proposed Data Replication Schemes

4.1. An Example

Here we use an example to illustrate our ideas. Suppose we are studying a network with only two nodes N_1 and N_2. N_1 and N_2 may access four same-size data items $d_1, ..., d_4$ but each node only has enough space to host two data items. Similar to [5], we assume that the access probability of nodes to data items are available. These probabilities are listed in Table 1.

According to the DAFN (Dynamic Access Frequency and Neighborhood) scheme proposed by Hara [5], neighboring nodes should try to remove duplicated data items. In the first replication step, nodes replicate data that they

Data	N_1	N_2
d_1	0.6	0.5
d_2	0.3	0.4
d_3	0.05	0.05
d_4	0.05	0.05

Table 1. Access Probability to Data Items

are interested in. Therefore, both nodes replicate d_1 and d_2 in their memory. In the second step, when two neighboring nodes have the same data item d_i, the node that has a lower access probability to d_i should replace d_i with the next most interested data. Therefore, N_1 replaces d_2 with d_3 and N_2 replaces d_1 with d_4. The final replication result is: N_1 host d_1 and d_3 while N_2 should host d_2 and d_4.

Intuitively, DAFN is good because duplicated data are removed from neighboring nodes and the memory size is used more effectively. However, its data accessibility may be affected when the link failure probability is high. The average data accessibility for N_1 and N_2 is shown by line "DAFN" in Figure 1. If instead of replicating data according DAFN, both N_1 and N_2 host d_1 and d_2, as shown by line "Our" in Figure 1, the data accessibility can be improved when the link failure probability is higher than 0.25. If the query delay is considered, the DAFN scheme obviously incurs higher query delay because many queries have to be satisfied by neighboring node.

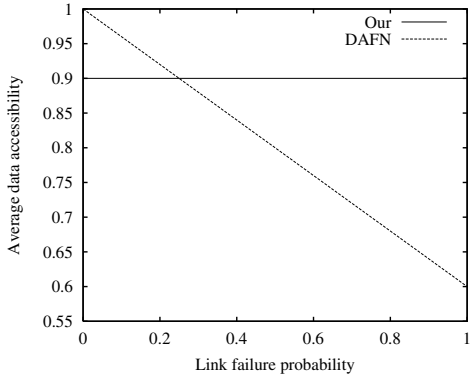


Figure 1. Data accessibility under different link failure probability between N_1 and N_2

In this example, the simple solution outperforms the DAFN scheme proposed in [5] because DAFN does not consider two important factors: the link stability between mobile nodes and the query delay. In our schemes, we consider both factors when making replication choices. Due to the complexity of the problem, next, we present the heuristics used in our solution.

4.2. Heuristics

Because mobile nodes have limited memory space, it is impossible for them to hold all their interested data. They have to rely on other nodes to get some data. If mobile nodes only host their interested data, it is possible that some data are in every node while some other data are not replicated. Therefore, it is important for mobile nodes to contribute part of their memory to hold data for other nodes. This is some kind of cooperation between mobile nodes. The problem is to decide the amount of space that a mobile node should contribute because bad cooperation may actually reduce the performance, as shown in the example above.

We have the following heuristics: For a mobile node, if its communication links to other nodes are stable, more cooperation with these nodes can improve the data accessibility; if the links to other nodes are not very stable, it is better for the node to host most of the interested data locally. The above heuristic mainly addresses the issue of data accessibility. For query delay, it is better to allocate data near the interested nodes. The degree of cooperation affects both the data accessibility and the query delay. Various schemes are proposed in this paper so that different performance targets can be achieved.

4.3. Data Replication Schemes

4.3.1. Greedy Schemes One naive greedy data replication scheme is to allocated the most frequently accessed data items until memory is full. However, this naive scheme, referred to as **Greedy**, does not consider the size difference between different data items. The data size should be considered because smaller data require less memory size, thus replicating them can save the memory size for other data items. Therefore, a better greedy scheme is to calculate the access frequency value of d_k by the following function:

$$AF_i(k) = a_{ik}/s_k \quad (1)$$

This greedy scheme, referred to as **Greedy-S**, let node N_i repeatedly picks the data item with the largest AF_i value from the data set that are not yet replicated at N_i until no more data can be replicated to the memory.

Performance Analysis: For simplicity, the data size is assume to be the same in the analysis, i.e., $s_i = 1, i = 1, 2, ..., n$. Because the computational complexity of the optimal scheme, we give an upper bound of the data accessibility by using a *super-optimal* algorithm, similar to the approach used in [16]. The solution given by the super-optimal algorithm is not a tight upper bound. It may be better than optimal and it may not be feasible. However, it is too difficult to find the tight upper bound and this super-optimal algorithm can be used for performance comparison.

A node N_i may have multiple one-hop neighbors. Assume that the probability of all links between N_i and its neighbors fail is f_{N_i}. For the greedy scheme, N_i hosts C most frequently accessed data, suppose this set of data is S_C. Then the data accessibility for the greedy scheme, A_{greedy}, follows:

$$A_{greedy} \geq \sum_{d_k \in S_C} a_{ik} \tag{2}$$

A super optimal solution for N_i would be allocating C most frequently access data in N_i, but allocating the other data in a way that they are all accessible from N_i's neighbors (this may not be possible in practice). It's data accessibility, A_{super}, is

$$A_{super} = \sum_{d_k \in S_C} a_{ik} + (1 - f_{N_i}) * \sum_{d_k \notin S_C} a_{ik} \tag{3}$$

Therefore,

$$\frac{A_{greedy}}{A_{super}} \geq \frac{\sum_{d_k \in S_C} a_{ik}}{\sum_{d_k \in S_C} a_{ik} + (1-f_{N_i}) * \sum_{d_k \notin S_C} a_{ik}} \geq \frac{\sum_{d_k \in S_C} a_{ik}}{\sum_{k=1}^{n} a_{ik}} \tag{4}$$

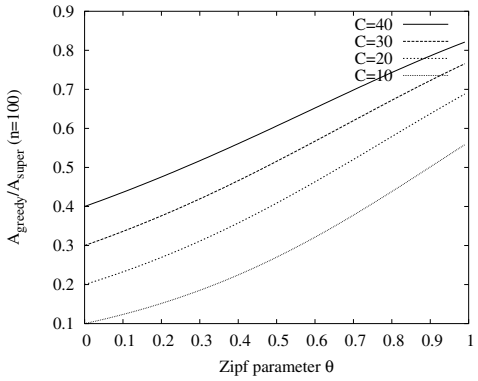


Figure 2. The worst case data accessibility of greedy scheme compared to the super-optimal scheme under different memory size C (the total number of data items $n = 100$)

Numeric Result: Figure 2 gives the numeric result that compares the worst case data accessibility of the greedy scheme with the data accessibility of the super-optimal scheme. The node access pattern is based on $Zipf - like$ distribution [22], which has been frequently used [1] to model non-uniform distribution. In the Zipf-like distribution, the access probability of the k^{th} $(1 \leq k \leq n)$ data item is represented as follows.

$$P_{a_k} = \frac{1}{k^{\theta} \sum_{i=1}^{n} \frac{1}{i^{\theta}}} \tag{5}$$

where $0 \leq \theta \leq 1$. When $\theta = 1$, it follows the strict Zipf distribution. When $\theta = 0$, it follows the uniform distribution. Larger θ results in more "skewed" access distribution. That is, more data accesses focus on the data items with small data id, which are called "hot" data.

We can see that the greedy scheme performs relatively well even when compared to a super-optimal scheme that may not be feasible at all. When the access pattern is more skewed, the greedy scheme performs better as more hot data access can be served by the replicated local copies.

One drawback of the greedy scheme is that it does not consider the cooperation between neighboring nodes and its performance may be limited. The following sections present schemes that include different degree of cooperation between neighboring nodes following our heuristics.

4.3.2. The One-To-One Optimization (OTOO) Scheme In this scheme, each mobile node only cooperates with at most one neighbor to decide which data to host. Suppose node N_i and N_j are neighboring nodes. N_i calculates the Combined Access Frequency (CAF) value of N_i and N_j to data item d_k at N_i, denoted as $CAF1_{ij}$, by using the following function:

$$CAF1_{ij}(k) = (a_{ik} + a_{jk} * (1 - f_{ij}))/s_i \tag{6}$$

Similarly N_j calculates its combined access frequency to d_k with the following function:

$$CAF1_{ji}(k) = (a_{jk} + a_{ik} * (1 - f_{ij}))/s_i \tag{7}$$

Each node sorts the data according the CAF1 value and picks data items with the highest CAF1 values to replicate in its memory until no more data items can be replicated. The CAF1 value function is designed so that 1) it considers the access frequency from a neighboring node to improve the data accessibility. 2) it considers the data size. If other criteria are the same, data with smaller size is given a higher priority for replicating because they can improve the performance while reducing the memory size requirement. 3) it gives the accessibility from the node itself a high priority so that its interested data can be replicated locally to improve the data accessibility and the query delay. The OTOO scheme works as follows:

1. All nodes are marked as "white" initially, which means that no one has executed the allocation process yet. These nodes broadcast their ids and their access frequency for each data item.
2. Among the white nodes, the node which has the smallest id among its neighboring white nodes starts the following process. It sends an invitation to the neighboring white node with which it has the lowest link failure probability. If the neighbor only receives one such invitation, these two neighboring nodes calculate the CAF1 values and each node allocates data items with the highest CAF1 values until it cannot accommodate

more data. Then both nodes are marked as "black" and no longer participate the replication process until the next allocation period.

3. In case that two or more nodes start the process at the same time, as long as they do not pick the same node as the most reliable neighbor, they can allocate their replicas at the same time. Otherwise, the node picked by more than one neighbor only accepts the invitation from the node with the lowest id. All other inviting nodes have to select another neighbor again.
4. If all neighbors of a white node are black nodes, which means that this white node cannot find any neighbor to cooperate in the allocation process, it only allocates its own most interested data items to its memory.

It is possible that according to OTOO, node N_i should host d_j but N_i is separated from nodes that have d_j because of network partitions. In this situation, N_i selects the next best candidate (data item) according the replication scheme. This rule is also applied to other replication schemes proposed in the following.

4.3.3. The Reliable Neighbor (RN) Scheme OTOO considers neighboring nodes when making data replication choices. However, it still considers its own access frequency as the most important factor because the access frequency from a neighboring node is reduced by the link failure probability factor. To further increase the degree of cooperation, we propose the Reliable Neighbor (RN) scheme that contributes more memory to replicate data for neighboring nodes. In this scheme, part of a node's memory is used to hold data for its *Reliable Neighbors*. For node N_i, a neighboring node N_j is considered to be N_i's reliable neighbor if

$$1 - f_{ij} > \mathcal{T}_r,$$

where $\mathcal{T}_r$ is a threshold value. Let $nb(i)$ be the set of the N_i's reliable neighbors. The total contributed memory size N_i, denoted as $Cc(i)$, is set to be

$$Cc(i) = C * min(1, \sum_{N_j \in nb(i)} (1 - f_{ij})/\alpha) \quad (8)$$

where α is a system tuning factor.

Intuitively, N_i contributes more memory if its links with neighboring nodes are more stable. The two extreme cases are: 1) when $Cc(i) = C$, N_i contributes all of its memory to hold data for neighboring nodes; 2) when $f_{ij} = 1, \forall N_j \in nb(i)$, N_i does not contribute any memory. The reason behind the RN scheme is that when links to neighboring nodes of N_i are stable, it is OK for N_i to hold more data for neighboring nodes as they also hold data for N_i. Because links are stable, such cooperation can improve the data accessibility. If links are not stable, data on neighboring nodes have low accessibility and may incur high query delay. Thus, cooperation in this situation cannot improve data accessibility and nodes should be more "selfish" in order to get better performance.

The data replication process works as follows. Node N_i first allocates its most interested data to its memory, up to $C - Cc(i)$ memory space. Then all the rest of the data are sorted according to $CAF2$ to a list called the neighbor's interest list. The CAF2 value of N_i to d_k is defined as:

$$CAF2_i(k) = (\sum_{N_j \in nb(i)} a_{jk} * (1 - f_{ij}))/s_k \quad (9)$$

$Cc(i)$ memory space are used to allocate data with highest $CAF2_i$ values. There may be some overlap between N_i's interested data and the allocated data interested by N_i's neighbors. If during the allocation, a data item is already in the memory, this data item will not be allocated again and the next data item on the neighbor's interest list is chosen instead.

4.3.4. Computation Overhead of the Proposed Schemes The proposed schemes need to sort all the data items according to the CAF value. The computational complexity of sorting them is $O(nlogn)$. This is the same as that of the DAFN scheme, although the constant factor may be higher because the calculation of the CAF value. However, as shown in the following section, our schemes are able to provide much better performance than the DAFN scheme.

5. Performance Evaluation

In this section, we evaluate the performance of the proposed schemes: OTOO, RN2 (RN with $\alpha = 2$), RN8 (RN with $\alpha = 8$), RN16 (RN with $\alpha = 16$), Greedy-S, by comparing to the DAFN scheme proposed by Hara [5] and the Greedy scheme.

5.1. The Simulation Model and System Parameters

In the simulation, m nodes are placed randomly in a 1500m $\times$ 1500m area. The radio range is set to be D. If two nodes N_i and N_j are within the radio range, i.e., the distance between them $D(i,j) < D$, they can communicate with each other. However, the link between them may fail. The link failure probability f_{ij} is defined as

$$f_{ij} = (\frac{D(i,j)}{D})^2 \quad (10)$$

Equation (10) is adopted according to the fact that the wireless signal strength decreases with a rate between the order of r^2 and r^4 , where r is the distance to the signal source. Note that proposed schemes do not depend on the failure model in Equation (10). They are able to work as long as the failure probability between neighboring nodes can be estimated. Because the link failure may be caused by many factors such as channel condition, node movement and node

failure, the actual link failure probability may not be estimated accurately. In the simulation, we also study the effect of inaccurate link failure probability on the proposed schemes. This is done by introducing an estimation error factor δ. The estimated link failure probability is the actual link failure probability multiplied by a factor, which is exponentially distributed with a mean value of δ. Different δ values, ranging from 0.6 to 1.4 are used to study our schemes' sensitivity to inaccurate estimations.

Similar to [5], the number of data items n is set to be the same as the number of nodes m. Data item d_i's original host is N_i, for all $i \in [1, m]$. The data item size is uniformly distributed between s_{min} and s_{max} memory units. Each node has a memory size of C.

Two access patterns are used in the simulation.

1. All nodes follow the Zipf-like access pattern, but different nodes have different hot data. This is done by randomly selecting an offset value for each node N_i: $offset_i$, which is between 1 and $n-1$. The actual access probability of N_i to data item d_k is given by:

$$P_{a_k} = \frac{1}{(((k+n-offset_i)\%n+1)^\theta \sum_{j=1}^{n} \frac{1}{j^\theta}}$$

   This means that the most frequently accessed data item id is moved to be $offset_i$ instead of 1 as given in Equation (5); the second frequently accessed data item id is $offset_i + 1$ instead of 2, and so on.
2. All nodes have the same access pattern and they have the same access probability to the same data item.

The performance metrics used in the simulation are data accessibility and query delay. When a query for data d_k is generated by node N_i, if d_k can be found at a node that is reachable through single or multi-hop links, this access is considered successful and the query delay is the number of hops from N_i to the nearest node that has d_k. If d_k is in the local memory of N_i, the query delay is 0. The average query delay only considers successful accesses, i.e., it is the total delay of successful accesses divided by the total number of successful accesses.

Most system parameters are listed in Table 2. The second column lists the default values of these parameters. In the simulation, we may change the parameters to study the impacts of these parameters. The ranges of the parameters are listed in the third column.

5.2. Simulation Results

5.2.1. Fine-tuning the RN scheme In Figure 3, we evaluate the effects of $\mathcal{T}_r$, defined in Section 4.3.3. Larger threshold value $\mathcal{T}_r$ results in smaller number of cooperative neighbors, and vice versa. We can see that $\mathcal{T}_r$ has the largest effect on the performance of RN2, which is because RN2 contributes the largest portion of the memory size to neighbors.

Parameter	Default value	Range
Number of nodes m	100	
Number of data items n	100	
s_{min}	0.1	
s_{max}	2	
Memory size C	20	
Radio range D	250m	50m - 300m
Zipf parameter θ	0.6	0.2 - 1.0
$\mathcal{T}_r$	0.6	0.2 - 0.8
Error factor δ		0.6 - 1.4

Table 2. Simulation parameters

We found that $\mathcal{T}_r = 0.6$ achieves a balance between the data accessibility and query delay, and similar results were found when nodes have different access pattern. Therefore, $\mathcal{T}_r = 0.6$ is adopted in the following.

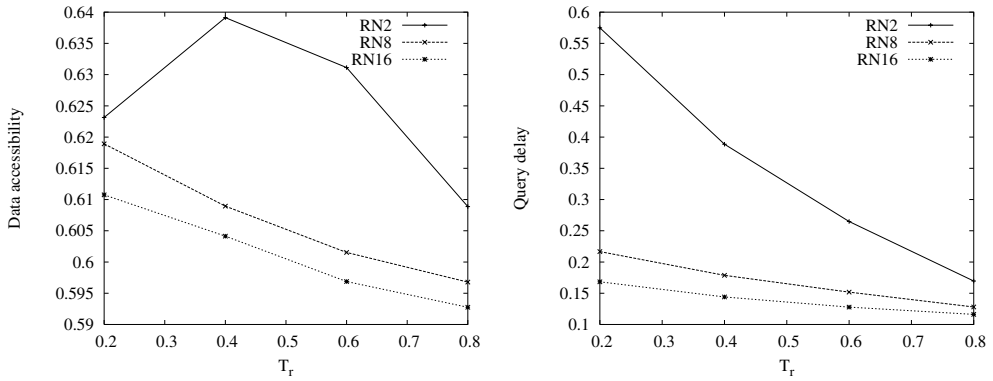
5.2.2. Effects of Zipf Parameter (θ) In this section, we evaluate the effects of the Zipf parameter θ on the system performance. As θ increases, more accesses focus on hot data items and the data accessibility is expected to increase.

Figure 4 demonstrates the effects of the Zipf parameter θ on the system performance when nodes have different access pattern. Figure 4 (a) shows that the proposed schemes outperform the DAFN scheme in terms of data accessibility in almost all cases. This is because: first, our schemes consider the link failure probability when replicating data; second, our schemes avoid replicating data items that are not frequently accessed by using the CAF value. On the other hand, the DAFN scheme does not consider the link failure probability and it sometimes replicates data items with low access frequency instead of frequently accessed data items, as shown in the example in Section 4.1.

Figure 4 (b) shows the query delay of different schemes. The DAFN scheme is outperformed by the proposed schemes in all situations. This shows that our schemes can achieve better performance in terms of both data accessibility and query delay. The DAFN scheme tries to avoid duplicated data items among neighboring nodes, which means that even if a data item is popular among two neighboring nodes, it is still allocated at only one of the neighboring nodes. Therefore, many accesses have to be satisfied by querying neighboring nodes, which increases the query delay.

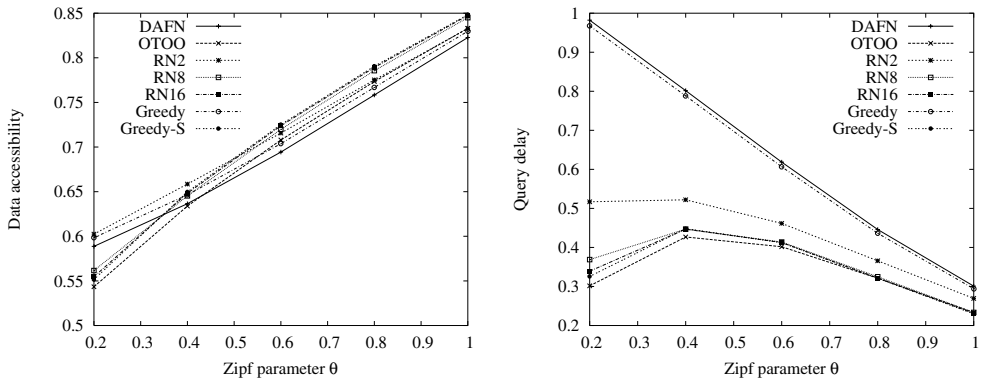
From Figure 4 (b), we can also find that the relation of query delay is $RN2 > RN8 \approx RN16 > OTOO$. This shows that when nodes have different interest, it is better for them to host data they are interested in, and cooperation among them does not show significant advantages.

Figure 5 shows the effects of the Zipf parameter θ on the system performance when nodes have the same access pat-



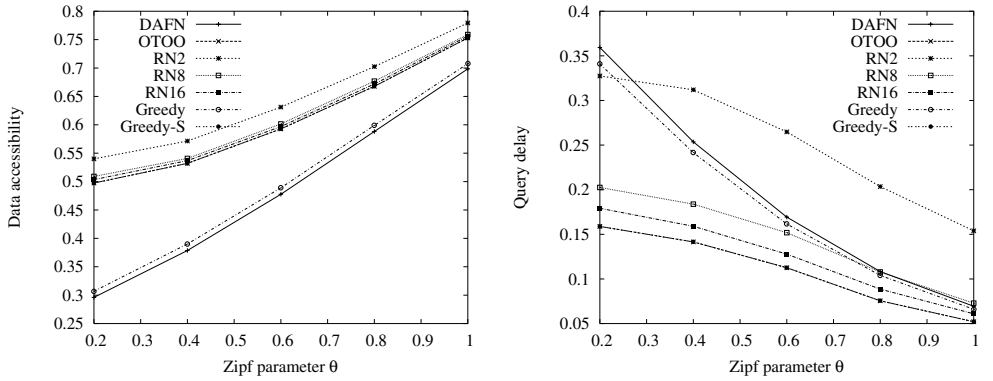
(a) Data Accessibility (b) Query Delay

Figure 3. Performance of RN as a function of $\mathcal{T}_r$ when nodes have the same access pattern



(a) Data Accessibility (b) Query Delay

Figure 4. Performance as a function of θ when nodes have different access pattern



(a) Data Accessibility (b) Query Delay

Figure 5. Performance as a function of θ when nodes have the same access pattern

tern. Note that in this situation, Greedy-S and OTOO are the same because Equation (2) only differs from Equation (6) at a constant factor. Greedy-S performs better than Greedy because it gives higher priority to data items with smaller size, and thus more important data can be replicated and the performance is improved.

We can see from Figure 5 that all the proposed schemes perform much better than the DAFN scheme in terms of data accessibility and all the proposed schemes except RN2 perform better than DAFN in terms of query delay. Comparing RN2, RN8, RN16, and OTOO, we find that the relation of their data accessibility is $RN2 > RN8 > RN16 > OTOO$ (RN2 performs the best) while the relation of their query delay is $RN2 > RN8 > RN16 > OTOO$ (OTOO performs the best). This clearly shows the tradeoffs betweens these two performance metrics. Higher degree of cooperation improves the data accessibility, but it also increases the query delay because more data need to be retrieved from neighboring nodes. This figure also gives us directions about how to achieve certain performance goals. If a high data accessibility is required, nodes should be more cooperative with neighboring nodes so that more data can be replicated in the network. If a low query delay is required, nodes should be more "selfish" so that requests can be served locally instead of by neighboring nodes.

5.2.3. Effects of Radio Range (D) Figure 6 shows the effects of the radio range on the system performance when nodes have the same access pattern. The result under the same access pattern is not shown due to space limitations. When the radio range increases, the network is better connected and the data accessibility is expected to increase. Figure 6 (a) shows that all schemes perform as expected. The proposed schemes perform much better than DAFN when the radio range is small. When the radio range is very large, different schemes have similar data accessibility. This is because the network partition is very rare in this situation and most data can be found in a reachable node.

Figure 6 (b) shows that the query delay increases as the radio range increases. This is because when the network is better connected, some data that are previously not available can now be found at faraway nodes, which increases the average query delay. The proposed schemes always result in lower query delay than the DAFN scheme. When the radio range is extremely small, the query delay of all scheme reduces to near zero, since it is hard to find a neighbor with such small radio range and almost all requests are served locally.

Figure 6 (c) evaluates the replication overhead in terms of the amount of data traffic incurred by the data replication schemes. The Greedy scheme and the Greedy-S scheme generate the lowest replication traffic because in their schemes, nodes do not cooperate with their neighbors. Because DAFN tries to remove duplicated data items in neighboring nodes, it always incurs the highest traffic. Among the RN schemes, RN2 generates the highest traffic and RN16 generates the lowest. This is because $RN2$ contributes a large amount of memory space to neighboring nodes but $RN16$ contributes the smallest. Overall, all of the proposed schemes perform better than DAFN in terms of the replication traffic. From Figure 6 (a), (b), and (c), we can conclude that our schemes can provide better performance with low traffic overhead.

5.2.4. Effects of the Error Factor of Link Failure Estimation (δ) This section evaluates the effects of error factor in link failure probability estimation, δ. The performance of DAFN, Greedy, and Greedy-S is not affected by δ as they do not depend on the estimation of link failure probability. From Figure 7, we can see that although δ affects the performance of RN2, RN8, RN16, and OTOO, the effect is not very significant even when the error is very large. We can conclude that they are robust and not sensitive to estimation errors.

6. Conclusions

In ad hoc networks, network partitions are common and data accessibility is low. In this paper, we proposed several data replication schemes to deal with this issue. In the OTOO scheme, nodes cooperate with only one neighboring node when making data replication decision. In RN2 , RN8, and RN16, nodes cooperate with more neighboring nodes and contribute more memory space to hold data for neighboring nodes. The difference between our scheme and existing schemes such as DAFN is that our schemes take link failure into account during data replication and try to balance data accessibility and query delay. Extensive performance evaluations demonstrate that the proposed schemes can provide high data accessibility. At the same time, our schemes achieve a balance between data accessibility and query delay.

References

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," *IEEE INFOCOM*, 1999.

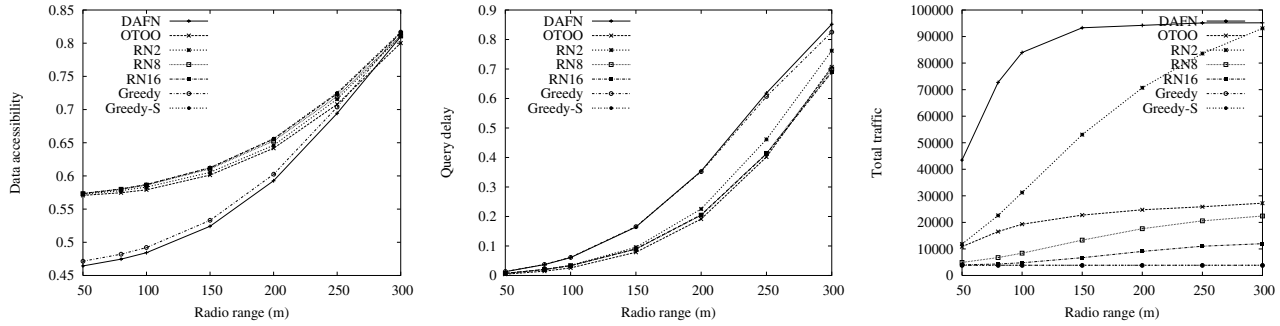
[2] G. Cao, L. Yin, and C. Das, "A Cooperative Cache Based Data Access Framework For Ad Hoc Networks," *IEEE Computer*, Feb. 2004.

[3] C. Chekuri and S. Khanna, "A PTAS for the Multiple Knapsack Problem," *Proceedings of the 11th SODA*, Jan. 2000.

[4] L. Gao, M. Dahlin, A. Nayate, J. Zheng, A. Iyengar, "Consistency and Replication: Application Specific Data Replication for Edge Services," *International conference on World Wide Web*, 2003.

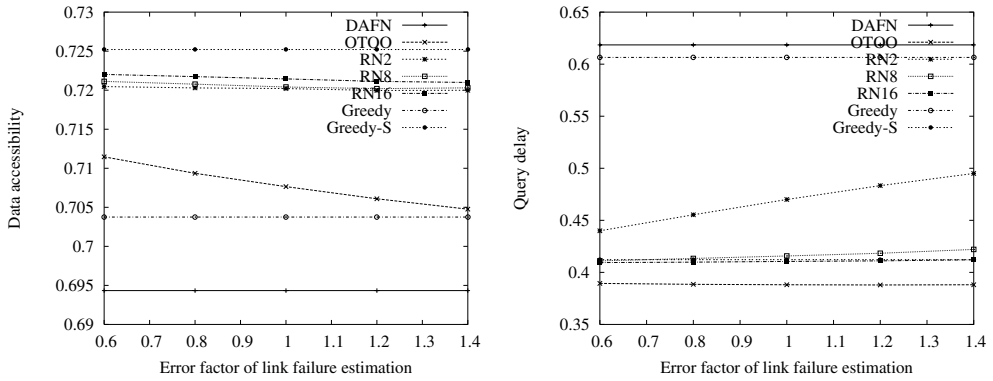
[5] T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," *IEEE INFOCOM*, 2001.

[6] T. Hara, "Replica Allocation in Ad hoc Networks with Periodic Data Update," *Proceeding of Int'l Conference on Mobile Data Management (MDM)*, 2002.



(a) Data Accessibility (b) Query Delay (c) Replication Traffic

Figure 6. Performance as a function of the radio range when nodes have different access pattern



(a) Data Accessibility (b) Query Delay

Figure 7. Performance as a function of δ when nodes have different access pattern

[7] Y. Huang, P. Sistla, and O. Wolfson, "Data Replication for Mobile Computers," *ACM SIGMOD*, pp. 13–24, 1994.

[8] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing, Kluwer*, pp. 153–181, 1996.

[9] G. Karumanchi, S. Muralidharan, and R. Prakash, "Information Dissemination in Partitionable Mobile Ad Hoc Networks," *18th IEEE Symposium on Reliable Distributed Systems*, Oct. 1999.

[10] B. Li, M. J. Golin, G. F. Ialiano, and X. Deng, "On the Optimal Placement of Web Proxies in the Internet," *IEEE INFOCOM*, March 1999.

[11] J. Luo, J. Hubaux, and P. Eugster, "Pan: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems ," *MobiHoc*, 2003.

[12] S. Martello and P. Toth, "Knapsack Problems: Algorithms and Computer Implementations," *John Wiley and Sons, Ltd.*, 1990.

[13] P. Nuggehalli, V. Srinivasan, Carla Chiasserini, and Ramesh Rao, "Energy-Efficient Caching Strategies in Ad Hoc Wireless Networks," *MobiHoc*, 2003.

[14] M. R. Pearlman and Z. J. Haas, "Determining the optimal configuration for the zone routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 8, pp. 1395–1414, Aug. 1999.

[15] E. Pitoura and B. Bhargava, "Maintaining consistency of data in mobile distributed environments," *Proc. IEEE ICDCS*, pp. 404–413, 1995.

[16] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," *IEEE INFOCOM*, 2001.

[17] K. Wang and B. Li, "Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks," *IEEE INFOCOM*, 2002.

[18] O. Wolfson and S. Jajodia, "Distributed Algorithm for Dynamic Replication of data," *Proc. ACM PODS'92*, pp. 149–163, 1992.

[19] O. Wolfson and A. Milo, "The multicast policy and its relationship to replicated data placement," *ACM Transactions on Database System*, 1991.

[20] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," *IEEE INFOCOM*, March 2004.

[21] H. Yu and A. Vahdat, "Minimal Replication Cost for Availability," *ACM Symposium on Principles of Distributed Computing (PODC)*, 2002.

[22] G. Zipf, "Human Behavior and the Principle of Least Effort," *Addison-Wesley*, 1949.