Balancing the Tradeoffs between Query Delay and Data Availability in MANETs

Yang Zhang, *Student Member, IEEE,* Liangzhong Yin, Jing Zhao, and Guohong Cao, *Fellow, IEEE*

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

Index Terms—Data replication, data availability, query delay, mobile ad hoc network (MANET).

1 Introduction

In mobile ad hoc networks (MANETs), since mobile nodes move freely, network partition may occur, where nodes in one partition cannot access data held by nodes in other partitions. Thus, data availability (i.e., the number of successful data accesses over the total number of data accesses) in MANETs is lower than that in conventional wired networks. Data replication has been widely used to improve data availability in distributed systems, and we will apply this technique to MANETs [1]. By replicating data at mobile nodes which are not the owners of the original data, data availability can be improved because there are multiple replicas in the network and the probability of finding one copy of the data is higher. Also, data replication can reduce the query delay since mobile nodes can obtain the data from some nearby replicas. However, most mobile nodes only have limited storage space, bandwidth and power, and hence it is impossible for one node to collect and hold all the data considering these constraints. By taking these issues into consideration, we expect that mobile nodes should not be able (or willing) to replicate all data items in the network (more discussions in Appendix A.)

One solution to improve the data access performance considering the resource constraints of mobile nodes is to let them cooperate with each other; i.e., contribute part of their storage space to hold data of others [2], [3]. When a node only replicates part of the data, there will be a tradeoff between query delay and data availability. For example, replicating most data locally can reduce the query delay, but it reduces the data availability since many nodes may end up replicating the same data locally, while other data items are not replicated by anyone. To increase the data availability, nodes should not replicate the same data that neighboring nodes already have. 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 much longer than accessing it locally.

In this paper, we propose new data replication techniques to address query delay and data availability issues. As both metrics are important for mobile nodes, we propose techniques to balance the tradeoffs between data availability 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. The next section presents some preliminaries of data replication. In Section 3, we describe the proposed schemes in detail. Section 4 evaluates the proposed schemes through extensive simulations and Section 5 concludes the paper.

2 Data Replication

Data replication has been extensively studied in the Web environment and distributed database systems (See Appendix B in the supplemental material for detailed literature review). However, most of them either do not consider the storage constraint or ignore

- *Y. Zhang, L. Yin, J. Zhao and G. Cao are with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, 16802.*
E-mail: {yangzhan,yin,jizhaogcao}@cse.psu.edu
- *This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-0721479, and by Network Science CTA under grant W911NF-09-2-0053.*

the link failure issue. Before addressing these issues by proposing new data replication schemes, we first introduce our system model.

In a MANET, mobile nodes collaboratively share data. Multiple nodes exist in the network and they send query requests to other nodes for some specified data items. Each node creates replicas of the data items and maintains the replicas in its memory (or disk) space. During data replication, there is no central server that determines the allocation of replicas, and mobile nodes determine the data allocation in a distributed manner.

The MANET studied in this paper can be represented as an undirected graph $G(V, E)$ where the set of vertices V represent the mobile nodes in the network, and $E \subset V \times V$ is the set of edges in the graph, which represents the physical or logical links between the mobile nodes. Two nodes that can communicate directly with each other are connected by an edge in the graph. Let $\mathcal{N}$ denote a network of m mobile nodes, $N_1, N_2, ..., N_m$ and let $\mathcal{D}$ denote a collection of n data items $d_1, d_2, ..., d_n$ distributed in the network. For each pair of mobile nodes N_i and N_j, let t_{ij} denote the delay of transmitting a data item of unit-size between these two nodes. Similar to [4], we assume that the delay function defines a metric space; that is, they are non-negative, symmetric and satisfy the triangle inequality. Links between mobile nodes may fail and the link failure probability between N_i and N_j is denoted as f_{ij}, which is equal to f_{ji} as we assume symmetric links. The failed links may cause network partitions. Queries generated during network partition may fail because the requested data items are not available in the partition to which the requester belongs.

Each node maintains some amount of data locally and the node is called the original owner of the data. Each data item has one and only one original owner. For simplicity, we assume that data items are not updated, and similar techniques used in [5], [6] and [7] can be used to extend the proposed scheme to handle data update or data consistency issues. To improve the data availability, these data items may be replicated to other nodes. Because of limited memory size, each node can only host $C(C < n)$ replicas besides its original data. The data replication problem, either optimizing the query delay or optimizing the availability, has been proved to be a reduction from the metric uncapacitated facility location problem, which is known to be NP-hard (see Appendix C). Therefore, instead of trying to find a complex algorithm that is not practical to solve or approximate the problem, we use heuristics that can provide satisfying performance with much less computation overhead.

The following notations are used in this paper.

- $\mathcal{N}$: the set of mobile nodes in the network.
- m: the total number of mobile nodes.
- $\mathcal{D}$: the set of available data items in the network.
- n: the total number of data items.
- s_i: the size of d_i.
- C: the memory size of each mobile node for hosting data replicas.
- t_{ij}: the delay of transmitting a data item of unit-size between node N_i and N_j.
- f_{ij}: the link failure probability between node N_i and N_j.
- a_{ij}: the data access frequency of node N_i to d_j.

3 The Proposed Data Replication Schemes

In this section, we propose several schemes to address the data replication problem based on heuristics. Before presenting these heuristics, we first use an example to illustrate the basic ideas.

3.1 A Motivating Example

Suppose a network has only two nodes N_1 and N_2. These two nodes may access four data items $d_1, ..., d_4$ with equal size, and each node only has enough space to host two data items. Similar to [8], we assume that the access probability of a mobile node to a data item is available. These probabilities are listed in Table 1.

TABLE 1
Access Probability to Data Items

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

According to the Dynamic Access Frequency and Neighborhood (DAFN) scheme proposed by Hara [8], neighboring nodes should try to remove duplicated data items to save storage space and increase data availability. In the first replication step, nodes replicate the data that they are interested in, and hence both nodes replicate d_1 and d_2 locally. In the second step of DAFN, when two neighboring nodes have the same data item d_i, the node that has a lower access probability should replace d_i with the next most frequently accessed 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 hosts d_1 and d_3 whereas N_2 hosts d_2 and d_4.

From this example and verified by simulations in [8], DAFN is a good scheme because duplicated data can be removed from neighboring nodes and the memory size can be used effectively. However, the data availability may be affected when the link failure probability is high. Figure 1 illustrates the average data availability as a function of link failure probability. As can be seen, the average data availability of DAFN drops as the link failure probability increases.

To address the weakness of DAFN, we can design a new scheme, denoted as "Our" in Figure 1. In our scheme, both N_1 and N_2 host d_1 and d_2 due to their high data access frequency. As shown in Figure 1, the data availability in our scheme becomes higher than that of DAFN when the link failure probability is higher than 0.25. Another advantage of our scheme is the low query delay. Since the data items are buffered locally, the query delay of our simple scheme should be much shorter than that of DAFN when accessing d_1 or d_2.

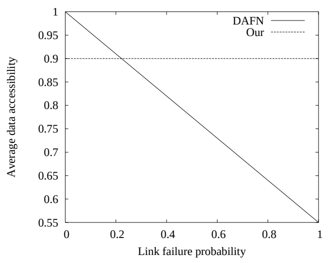


Fig. 1. The data availability under different link failure probability between N_1 and N_2

In this example, the simple solution outperforms the DAFN scheme because DAFN does not consider two important factors: the link stability between mobile nodes and the query delay. Due to the complexity of the data replication problem shown in Appendix C, we propose some heuristics.

Heuristics Because mobile nodes have limited memory, it is impossible for them to hold all their interested data items. As a result, 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 items are replicated by every node while some other data items are not replicated by anyone. Therefore, it is important for mobile nodes to cooperate with each other and contribute part of their memory to hold data for other nodes. The problem is to determine the memory space that a mobile node should contribute because bad cooperation may actually degrade the performance, as shown in the above example.

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 availability; 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 availability. For query delay, it is better to allocate data near the interested nodes. The degree of cooperation affects both the data availability and the query delay. In the following, we propose various schemes to achieve various performance goals.

3.2 The Greedy Data Replication Scheme

One naive greedy data replication scheme is to allocate the most frequently accessed data items until the memory is full. However, this naive scheme, referred to as **Greedy**, does not consider the data size difference between different data items. The data size should be considered because smaller data require less memory space, and hence replicating them can save some memory space for other data items. Therefore, a better greedy scheme is to calculate the data access frequency of a data item d_k by normalizing it against the data size, i.e., a_{ik}/s_k.

This greedy scheme, referred to as **Greedy-S**, lets node N_i repeatedly pick the data item with the largest a_{ik}/s_k value from the data set that has not yet been replicated at N_i until no more data can be replicated in the memory. One drawback of the greedy scheme is that it does not consider the cooperation between the neighboring nodes and hence its performance may be limited. We present the performance analysis and numerical results in Appendix D of the supplement material. The following sections present schemes that apply different levels of cooperation between neighboring nodes following our heuristics.

3.3 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 replicate. Suppose node N_i and N_j are neighboring nodes. N_i calculates the *combined access frequency* value of N_i and N_j to data item d_k at N_i, denoted as CAF_{ij}^k, by using the following function:

$$CAF_{ij}^k = (a_{ik} + a_{jk} \times (1 - f_{ij}))/s_i \qquad (1)$$

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

$$CAF_{ji}^k = (a_{jk} + a_{ik} \times (1 - f_{ij}))/s_i \qquad (2)$$

We also need to consider the increased data availability due to neighboring nodes. If the neighboring node N_j of N_i has already replicated the data and the link failure probability between N_i and N_j is low, N_i is less likely to replicate this data because it can always get the data from N_j. However, if the link failure probability is high, N_i may like to replicate the data locally. Therefore, we define a priority value for node N_i to replicate data d_k given its neighboring node N_j, denoted as $\mathcal{P}_{ij}^k$, by using the following function:

$$\mathcal{P}_{ij}^k = CAF_{ij}^k \times \omega_{ij}^k \qquad (3)$$

where ω_{ij}^k indicates the impact on data availability by the neighboring node and the link failure probability. The value of ω_{ij}^k is calculated as follows:

$$\omega_{ij}^k = \begin{cases} f_{ij} & \text{if data } d_k \text{ is replicated at } N_j; \\ 1 & \text{if data } d_k \text{ is not replicated at } N_j. \end{cases}$$

Each node sorts the data according to the priority value $\mathcal{P}$ and picks data items with the highest $\mathcal{P}$ to replicate in its memory until no more data items can be replicated. The $\mathcal{P}$ value function is designed so that 1) it considers the access frequency from a neighboring node to improve data availability; 2) it considers the data size. If other criteria are the same, the data item with smaller size is given higher priority for replicating because this can improve the performance while reducing memory space; 3) it gives high priority to local data access, and hence the interested data should be replicated locally to improve data availability and reduce query delay; 4) it considers the impact of data availability from the neighboring node and link quality. Thus, if the link between two neighboring nodes are stable, they can have more cooperations in data replication.

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 to the replication scheme. This rule is also applied to other replication schemes proposed in the following. The detailed pseudo-code and descriptions of the OTOO scheme and the following schemes are provided in Appendix E of the supplemental masterial.

3.4 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 a factor of the link failure probability. To further increase the degree of cooperation, we propose the Reliable Neighbor (RN) scheme which contributes more memory to replicate data for neighboring nodes. In this scheme, part of the 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 N_i's reliable neighbors. The total contributed memory size of N_i, denoted as $Cc(i)$, is set to be

$$Cc(i) = C \times min(1, \sum_{N_j \in nb(i)} (1 - f_{ij})/\alpha) \tag{4}$$

where α is a system tuning factor which affects the memory allocated to itself and its neighbors.

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 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, N_i can hold more data for neighboring nodes as they also hold data for N_i. Because links are stable, such cooperation can improve the data availability. If links are not stable, data on neighboring nodes have low availability and may incur high query delay. Thus, cooperation in this case cannot improve data availability and nodes should be more "selfish" in order to achieve 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 $\mathcal{P}$ to a list called the neighbor's interest list. The $\mathcal{P}$ value of N_i to d_k is defined as:

$$\mathcal{P}_i^k = (\sum_{N_j \in nb(i)} y_j^k \times a_{jk} \times (1 - f_{ij}))/s_k \tag{5}$$

The memory space of $Cc(i)$ is used to allocate data with the highest $\mathcal{P}$ 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.

3.5 Reliable Grouping (RG) Scheme

OTOO only considers one neighboring node when making data replication decisions. RN further considers all one-hop neighbors. However, the cooperations in both OTOO and RN are not fully exploited. To further increase the degree of cooperation, we propose the reliable grouping (RG) scheme which shares replicas in large and reliable groups of nodes, whereas OTOO and RN only share replicas among neighboring nodes. The basic idea of the RG scheme is that it always picks the most suitable data items to replicate on the most suitable nodes in the group to maximize the data availability and minimize the data access delay within the group.

In the RG scheme, there is no redundant replication until every data item is replicated at least once. Therefore, the maximum degree of cooperation within the reliable group can be achieved. Because the function for selecting the best node to place each data replica considers the access delay between the query node and the nearest replication node in the group, the RG scheme can reduce the number of hops that the data need to be transferred to serve the query.

Due to the page limitation, the detailed protocol description and a comprehensive performance complexity and bound analysis of the proposed schemes are presented in Appendix E and F in the supplemental material.

4 Performance Evaluations

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 $\alpha = 16$), and RG by comparing them with the DAFN scheme [8] and the Greedy scheme through extensive simulations.

4.1 Simulation Setup

We have developed a simulator based on CSIM 19 [9] to evaluate the performance of the data replication schemes. At the beginning of the simulation, m nodes are placed randomly in a 2500m $\times$ 2500m area. The radio range is set to be D. If two nodes N_i and N_j are within the radio range (i.e., $D(i,j) < D$), they can communicate with each other. The communication link between them may fail and the link failure probability f_{ij} is defined as

$$f_{ij} = (\frac{D(i,j)}{D})^2 \cdot \beta \tag{6}$$

Equation (6) is adopted according to the facts that the wireless signal strength decreases with a rate between the order of r^2 , where r is the distance to the signal source. For example, if two connected nodes have a long distance, they are easier to disconnect and the link failure probability between them is higher. β is used to adjust f_{ij} to a more reasonable value. The proposed schemes do not depend on the failure model in Equation (6) and they are able to work as long as the failure probability between neighboring nodes can be estimated.

Similar to [8], 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}. 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 items. 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:

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

This means that the most frequently accessed data item id is moved to be $offset_i$ instead of 1; the second frequently accessed data item id is $offset_i + 1$ instead of 2, and so on.

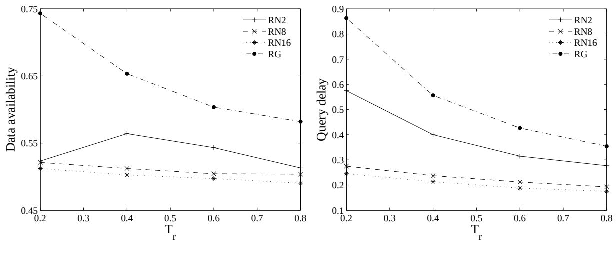
2) All nodes have the same access pattern and they have the same access probability to the same data item.

In order to avoid routing cycles on the query path, a maximal hop count is used to limit the number of hops for each query. It is set to be $\frac{\sqrt{2}\cdot 2500}{D}$, where 2500 is the size of the simulation area.

The performance metrics used in the simulation are mainly data availability and query delay. The amount of query traffic is also evaluated to show the protocol overhead. Here, we note that the amount of query traffic can also be used as a metric of system power consumption. This is because in wireless communication, data transmission is the key factor affecting the system power consumption compared to other factors such as disk or CPU operations [10]. Therefore, if there is more query traffic, more energy consumption is expected. If one replication scheme generates less traffic, it is more power efficient.

TABLE 2
Simulation parameters

Parameter	Default value	Range
Number of nodes m	300	100, 300, 500
Memory size C	20	
Radio range D	150m	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
β	0.95	0.7 - 1.0
V	0	0 - 60



(a) Data Availability (b) Query Delay

Fig. 2. Performance of RN and RG as a function of $\mathcal{T}_r$ when nodes have the same access pattern

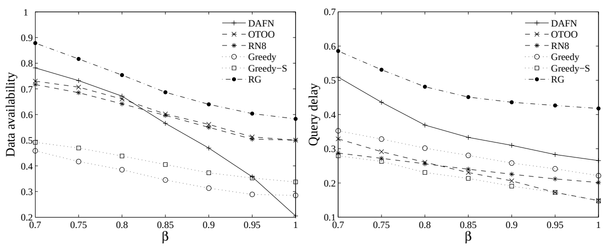
When a query for data d_k is generated by node N_i, if d_k can be found locally, or at a node that is reachable through single or multi-hops, this access is considered successful. The query delay is the number of hops from N_i to the nearest node that has d_k multiplied by the data size, and query traffic is defined as all messages involved to serve the query. If d_k is in the local memory of N_i, the query delay and query traffic are both 0. Most system parameters are listed in Table 2.

4.2 Simulation Results

Experiments were run using different workloads and system settings. The performance analysis presented here is designed to compare the effects of different workload parameters such as *Zipf* parameter, network size, radio range, memory size and node mobility (due to the space limitation, the effects of different mobility models are provided in Appendix G). For each workload parameter (e.g., the mean update arrival time or the mean query generate time), the mean value of the measured data is obtained by collecting a large number of samples such that the confidence interval is reasonably small. In most cases, the 95% confidence interval for the measured data is less than 10% of the sample mean.

4.2.1 Fine-tuning $\mathcal{T}_r$

In Figure 2, we evaluate the effects of $\mathcal{T}_r$, which affects the number of cooperative neighbors in the RN scheme and the RG scheme. Larger $\mathcal{T}_r$ results in smaller number of cooperative neighbors, and vice



(a) Data Availability (b) Query Delay

Fig. 3. Performance as a function of β when nodes have the same access pattern

versa. We can see that $\mathcal{T}_r$ has more significant effects on the performance of RN2 (RN with $\alpha = 2$) than RN8 and RN16, because RN2 contributes the largest portion of the memory size to neighbors. The performance of the RG scheme is also affected by $\mathcal{T}_r$, because the change of $\mathcal{T}_r$ affects the number of nodes in a reliable group. From Figure 2(a), we can see that when $\mathcal{T}_r < 0.4$, as long as $\mathcal{T}_r$ increases, the data availability of RG, RN8 and RN16 are decreasing while the data availability of RN2 is increasing; when $0.4 < \mathcal{T}_r < 0.6$, all schemes have a decreasing trend in data availability as $\mathcal{T}_r$ increases; similar trend can be found when $\mathcal{T}_r > 0.6$. In Figure 2(b), when $\mathcal{T}_r$ changes from 0.2 to 0.4, RG and RN2 have a large decrease in query delay; however, when $\mathcal{T}_r$ becomes larger than 0.4, all four schemes have stable and small delay decrease. When $\mathcal{T}_r$ is around 0.6, all schemes have relatively stable performance, which means the change of $\mathcal{T}_r$ does not have significant effect on the relative performance of different data replication schemes. Thus, we use $\mathcal{T}_r = 0.6$ in the following.

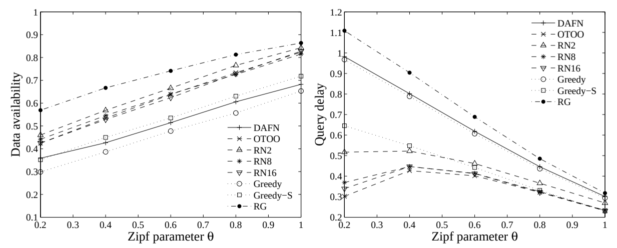
4.2.2 Fine-tuning β

By controlling β, the link failure probability can be adjusted. When the link failure probability deceases, data availability increases as shown in Figure 3. We choose $\beta = 0.95$ to achieve a balance among all replication schemes.

As can be seen from the figure, DAFN has high query delay because it tries to avoid duplicated data among neighboring nodes. 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 the querying neighboring nodes, which increases the query delay. For similar reasons, the query delay of RG is also high. However, RG considers all nodes in a reliable group during data replication. It organizes data better within each reliable group, which helps RG achieve higher data availability.

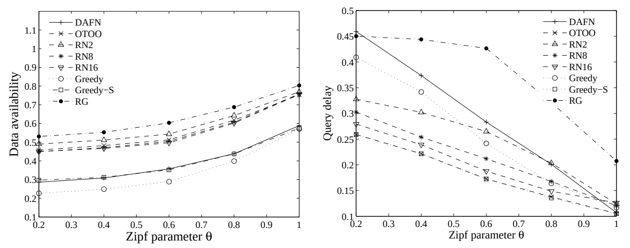
4.2.3 *Effects of the* Zipf *Parameter* (θ)

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 data availability is expected to increase.



(a) Data Availability (b) Query Delay

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



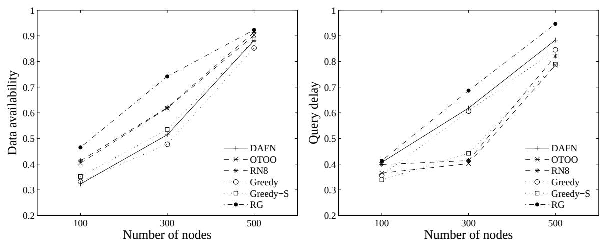
(a) Data Availability (b) Query Delay

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

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 availability in most cases. The reasons are as follows: first, our schemes consider the link failure probability when replicating data (for OTOO and RN) or organizing groups (for RG); second, the OTOO and RN schemes avoid replicating data items that are not frequently accessed by using the $\mathcal{P}$ 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 3.1.

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

Figure 5 shows the effects of the *Zipf* parameter θ on the system performance when nodes have the same access pattern. We can see from Figure 5 that all the proposed schemes perform much better than the DAFN scheme in terms of data availability and all the proposed schemes in most situations perform better than DAFN in terms of query delay. Greedy-S performs better than Greedy because it gives higher



(a) Data Availability (b) Query Delay

Fig. 6. Performance as a function of number of nodes in the network when nodes have different access pattern

priority to data items with smaller size, and thus more important data can be replicated and the performance is improved. Comparing RN2, RN8, RN16, OTOO, and RG, we find that the relation of their data availability is $RG > RN2 > RN8 > RN16 \approx OTOO$ (RG performs the best as expected) while the relations of their query delay is $RG > RN2 > RN8 > RN16 > OTOO$ (OTOO performs the best). This clearly shows the tradeoffs between these two performance metrics. Higher degree of cooperation improves the data availability, but it also increases the query delay because more data items need to be retrieved from neighboring nodes. This figure also gives us directions on how to achieve certain performance goals. If high data availability is required, nodes should be more cooperative with neighboring nodes so that more data can be replicated in the network. If low query delay is more important, nodes should be more "selfish" so that requests can be served locally instead of by neighboring nodes.

Since RN2, RN8 and RN16 exhibit similar performance when other parameters change, to make the simulation figures clear, we will only use RN8 to represent the RN schemes.

4.2.4 Effects of the Number of Nodes in the Network (m)

The number of nodes in the network indicates the node density of the network. When the number of nodes increases, the density of the network increases and it becomes better connected and the data availability increases. Figure 6 shows the effects of the number of nodes on the system performance. In Figure 6(a), we can see that when there are only 100 nodes in the network, all schemes have relatively lower data availability due to the sparse network connectivity. As the number of nodes increases, nodes have more opportunities to get the data from their neighboring nodes, and all schemes have performance improvements in terms of data availability as expected. When the network density further increases, e.g., in a 500-nodes scenario, the data availability of all schemes approaches to 0.9. Similar observations can be found in Figure 6(b). Therefore, we choose $m = 300$ as the default setting to see the effects of different schemes on the system performance.

4.2.5 Effects of the Radio Range (D)

Figure 7 shows the effects of the radio range on the system performance under different access pattern. When the radio range increases, the network is better connected and the data availability is expected to increase. Figure 7 (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 availability. This is because the network partition is very rare in this situation and most data can be found in a reachable node.

Figures 7 (b) and (c) show that the query delay and query traffic increase as the radio range increases. This is because when the network is better connected, some previously unavailable data can be found at faraway nodes. The proposed schemes always result in lower query delay and traffic than the DAFN scheme. When the radio range is extremely small, the query delay of all schemes reduces to near zero, since it is hard to find a neighbor with such small radio range and almost all requests are served locally.

4.2.6 Effects of Memory Size (C)

In this section, we evaluate the system performance when the memory size (C) changes. As C increases, more data can be hosted by a node and the data availability increases. Similarly, more data can be found locally as C increases and the query delay and query traffic decrease.

Figure 8 shows that when nodes have different access patterns, the proposed schemes increase the data availability while providing lower query delay and query traffic compared to the DAFN scheme. The difference of data availability for OTOO, RN8, Greedy, Greedy-S and DAFN is not very large because when nodes have different access pattern, they can simply replicate their interested data locally to achieve a high data availability. Thus the room for improvement is small. RG, however, organizes data replications within each reliable group. It can provide more different data items in each group. Thus its data availability is much higher than other schemes.

5 Conclusions

In MANETs, due to link failure, network partitions are common. As a result, data saved at other nodes may not be accessible. One way to improve data availability is through data replication. In this paper, we proposed several data replication schemes to improve the data availability and reduce the query delay. The basic idea is to replicate the most frequently accessed data locally and only rely on neighbor's memory when the communication link to them is reliable.

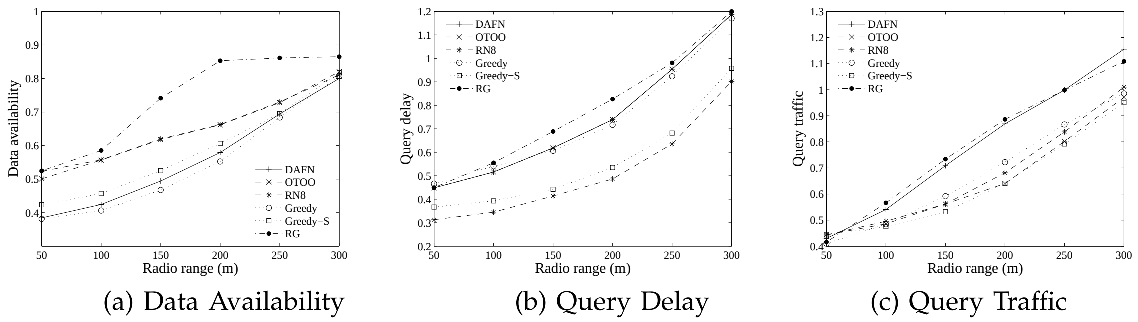


Fig. 7. Performance as a function of the radio range when nodes have different access pattern

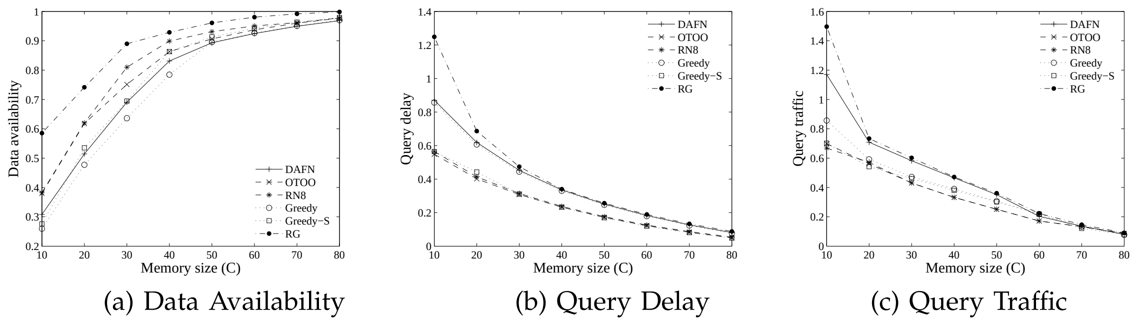


Fig. 8. Performance as a function of memory size when nodes have different access pattern

Extensive performance evaluations demonstrate that the proposed schemes outperform the existing solutions in terms of data availability and query delay. Results also show that there is a fundamental tradeoff between data availability and query delay. Higher degree of cooperation improves the data availability, but it also increases the query delay because more data need to be retrieved from neighboring nodes.

References

[1] T. Hara and S. K. Madria, "Data replication for improving data accessibility in ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 11, pp. 1515–1532, 2006.

[2] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *IEEE Transaction on Mobile Computing*, vol. 5, no. 1, January 2006.

[3] B. Tang, H. Gupta, and S. Das, "Benefit-based data caching in ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 3, pp. 289–304, March 2008.

[4] I. Baev and R. Rajaraman, "Approximation algorithms for data placement in arbitrary networks," *ACM-SIAM SODA*, pp. 661–670, 2001.

[5] T. Hara and S. Madria, "Consistency management strategies for data replication in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 7, pp. 950–967, 2009.

[6] T. Hara, "Replica allocation in ad hoc networks with periodic data update," *International Conference on Mobile Data Management (MDM)*, 2002.

[7] J. Cao, Y. Zhang, G. Cao, and L. Xie, "Data consistency for cooperative caching in mobile environments," *IEEE Computer*, vol. 40, no. 4, pp. 60–66, 2007.

[8] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," *IEEE INFOCOM*, 2001.

[9] H. Schwetman, "Csim19: a powerful tool for building system models," *The 33nd Conference on Winter Simulation*, pp. 250–255, 2001.

[10] R. Kravets and P. Krishnan, "Power management techniques for mobile communication," *ACM MOBICOM*, pp. 157–168, 1998.

Yang Zhang received both BS degree and ME degree from Nanjing University, Nanjing, China, in 2002 and 2005, respectively. He is currently working toward the Ph.D degree at the Pennsylvania State University. His research interests include distributed systems, mobile computing, and vehicular adhoc networks. He is a student member of IEEE.

Liangzhong Yin received the Ph.D. in computer science and engineering from the Pennsylvania State University, in 2005. His research interests include wireless networks and mobile computing.

Jing Zhao received the BS degree from Peking University, Beijing, China. He is currently pursuing the PhD degree in computer science and engineering at the Pennsylvania State University. His research interests include wireless networks, distributed systems, and mobile computing, with a focus on mobile ad hoc networks.

Guohong Cao received the BS degree from Xian Jiaotong University, China. He received the MS degree and PhD degree in computer science from the Ohio State University in 1997 and 1999 respectively. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Professor. His research interests are wireless networks and mobile computing. He has published more than 150 papers in the areas of wireless sensor networks, wireless network security, vehicular ad hoc networks, data access and dissemination, and distributed fault tolerant computing. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, and has served as program committee members of many conferences. He was a recipient of the NSF CAREER award in 2001. He is a Fellow of the IEEE.

Supplemental Material: Balancing the Tradeoffs between Query Delay and Data Availability in MANETs

Yang Zhang, *Student Member, IEEE,* Liangzhong Yin, Jing Zhao, and Guohong Cao, *Fellow, IEEE*

CONTENTS OF THE SUPPLEMENTARY FILE

Some detailed contents were omitted in the main manuscript due to space limit. These contents are provided in this supplementary file. Appendix A provides detailed discussions of using data replication to improve data access performance in MANETs, which also gives the motivation of this work. Appendix B provides a comprehensive literature review of existing works on data replication. Appendix C formulates the data replication problem and Appendix D analyzes the performance of the Greedy scheme. Details of the proposed data replication schemes are introduced in Appendix E and Appendix F presents the computation complexity and the performance bound of these schemes. More simulation results based on different mobility models are given in Appendix G, and finally we discuss further research issues in Appendix H. Note that some text and references are similar to the main manuscript to make the supplementary file self-contained.

APPENDIX

A: USING DATA REPLICATION TO IMPROVE DATA ACCESS PERFORMANCE IN MANETS

Figure 1 shows an example of how data replication can be used to improve the performance of data access when network partitions. There are four nodes in the network. N_4 is a web camera which continuously records video clips (d_1) of its surroundings. Two clients N_1 and N_2 periodically access these video clips by using N_3 as relay. However, when a disconnection occurs between N_3 and N_4 due to a link failure, d_1 becomes unaccessible to the other three nodes. To improve data availability, a copy of d_1 can be replicated at N_3 before the disconnection. Then both N_1 and N_2 can access d_1 even if they are not able to connect with N_4. Further, by replicating a copy of d_1 at N_3, N_1 and N_2 can access d within one hop, reducing the query delay.

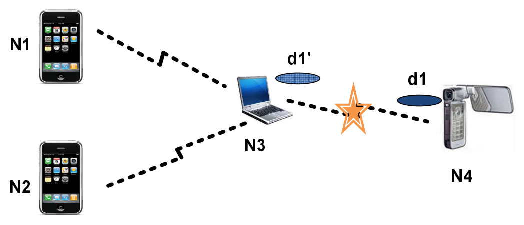


Fig. 1. Network partition due to link failure in a MANET

However, many mobile nodes only have limited storage space, bandwidth and power, and hence it is impossible for one node to collect and hold all the data considering these constraints. For example, in disaster recovery, some rescue workers may only have resource constrained cellular phones which have limited storage, and thus cannot replicate all the data such as pictures or video clips. In applications such as data centric sensor networks [1], resource constrained sensor nodes only have limited storage (less than a megabyte). To help mobile sinks get the data quickly, the data may have to be carefully placed to increase the data availability and reduce the query delay [2]. For applications such as geological exploration, although mobile devices have large storage, these geologists may have to record videos or generate large volumes of raw scientific data, which can quickly use up the storage. Thus, replicating the data everywhere is not a good solution due to storage limitation. Furthermore, to replicate data, nodes need to transmit it from other nodes and obviously there will be huge bandwidth and power cost for large volume of data. By taking these issues into consideration, we expect that mobile nodes should not be able (or willing) to

- *Y. Zhang, L. Yin, J. Zhao and G. Cao are with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, 16802.*
E-mail: {yangzhan,yin,jizhaogcao}@cse.psu.edu
- *This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-0721479, and by Network Science CTA under grant W911NF-09-2-0053.*

replicate all data items in the network.

B: Related Work of Data Replication in MANETs

Data replication has been extensively studied in the Web environment [3], [4], where the goal is to place some replicas of the web servers among a number of possible locations so that the query delay is minimized. In the Web environment, links and nodes are stable. Thus, the performance is mainly measured by the query delay. Moreover, these schemes replicate 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 done at the data item level, i.e., how to replicate data items to various nodes with limited memory space.

Data replication has been studied in distributed database systems [5]. In such systems, nodes that host the database are more reliable and less likely to fail/disconnect compared to those in MANETs. Therefore, a small number of replicas can be used to provide high availability. However, in MANETs, node/link failure occurs frequently, and data availability becomes an important issue.

Padmanabhan [6] *et al.* identified several research issues in data replication in MANET and attempted to classify existing data replication techniques. Hara [7], [8] proposed data replication schemes for ad hoc networks. These schemes are based on the intuition that replicating the same data near neighboring nodes should be avoided in order to improve data availability. However, this intuition may not be valid when the link failure probability is taken into consideration. Also, it only considers the availability, without considering the query delay. We will address these issues in this paper to provide better data replication.

Some other researchers address data access issues in MANETs considering network partitions. Jorgic and Stojmenovic *et al.* [9] introduced several localized algorithms to detect critical nodes and links for connectivity in MANETs. Huang and Chen [10] addressed the problem of replica allocation in a MANET by exploring group mobility. Wang and Li [11] proposed schemes to deal with network partitions due to node movement by replicating services in the network. Their schemes can provide guaranteed service with minimum number of replicated services. Hara [12] proposed several metrics to evaluate the impact of mobility on data availability. Ramany and Bertok [13] studied solutions for replicating location dependent data in MANETs to handle unreliable network connections. Boulkenafed and Issarny [14] presented a middleware service that allows collaborative data sharing among ad hoc groups that are dynamically formed according to the network connectivity. Different from these previous works, we study the tradeoffs between query delay and data availability.

Besides data replication, caching can also be used to improve data availability and reduce the query delay [15]–[17]. In [18], we have 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 the same data, and the intermediate nodes can return the data or the path to the data. As the number of hops is reduced, the query delay is also reduced. Implementation issues of cooperative cache are further studied in [19]. 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 is only cached after some nodes start to use it. To further increase the data availability and reduce the query delay, we study proactive data replication techniques where nodes actively replicate data in this paper.

C: Problem Formulation

A data replication is a function $\mathcal{N} \rightarrow 2^{\mathcal{D}}$ that returns the set of data items stored at each node. For a replication to be valid, the total amount of data stored at each node should not be larger than C. Different performance metrics can be used for the data allocation problem. To simplify the problem, we first consider how to optimize data replication with one metric. Then, the goal is to allocate n data items into m mobile nodes so that a certain performance metric (data availability, query delay, etc.) is optimized.

To optimize query delay:

For any mobile node N_i and data item d_k, the demand-weighted access delay for N_i to access d_k is equal to $a_{ik} \cdot t_{ij} \cdot s_k$, where N_j is the node nearest to N_i that has a replica of data d_k and s_k is the size of data d_k. Then the total query delay of a replication is given by the sum of all access delay (taken over all nodes $\in \mathcal{N}$ and all data items $\in \mathcal{D}$).

To optimize query delay, the replication problem can be written as an integer linear programming (ILP) as follows. For each data $d_k \in \mathcal{D}$, let binary variable y_j^k, $N_j \in \mathcal{N}$, indicate if node j is selected to store a replica of data d_k, and binary variable x_{ij}^k, $N_i, N_j \in \mathcal{N}$, indicate if node N_j is the nearest accessible node to N_i that stores the copy of data d_k. That means node N_j is assigned to serve the queries issued by N_i for data d_k:

$$min \sum_{k \in \mathcal{D}} \sum_{N_i, N_j \in \mathcal{N}} a_{ik} \cdot t_{ij} \cdot s_k \cdot x_{ij}^k$$

subject to

$$\sum_{N_j \in \mathcal{N}} x_{ij}^k = 1 \qquad N_j \in \mathcal{N}, d_k \in \mathcal{D} \tag{1}$$

$$x_{ij}^k \le y_j^k \qquad N_i, N_j \in \mathcal{N}, d_k \in \mathcal{D} \tag{2}$$

$$\sum_{d_k \in \mathcal{D}} s_k \cdot y_j^k \le C \qquad N_j \in \mathcal{N} \tag{3}$$

$$x_{ij}^k \in \{0,1\} \qquad N_i, N_j \in \mathcal{N}, d_k \in \mathcal{D} \tag{4}$$

$$y_j^k \in \{0,1\} \qquad N_j \in \mathcal{N}, d_k \in \mathcal{D} \tag{5}$$

Here constraint (1) guarantees that each query for some data item is always assigned to some node that has the replication of the data. Constraint (1) also indicates that for each query from node N_i for data d_k, there is always one and only one closest node N_j which has the data replication. Constraint (2) ensures that, whenever a node N_j serves the query from other nodes for data d_k, it must have stored the data. Constraint (3) ensures that each node cannot store more data than its memory capacity. Constraint (4) and constraint (5) indicate that x_{ij}^k and y_i^k can only be binary variable.

To optimize data availability:

To optimize the data availability, the data replication problem can also be stated as an integer linear programming. Because f_{ij} is the link failure probability between node N_i and node N_j, each query node can always access its query data until all nodes that buffer the requested data are unaccessible. Thus, the demand-weighted availability of node N_i for data item d_k is equal to $a_{ik} \cdot (1 - \prod_{N_j \in \mathcal{N} \wedge y_j^k = 1} f_{ij})$. Then, the goal is to find the best replication arrangement so as to optimize the data availability function:

$$max \sum_{k \in \mathcal{D}} \sum_{N_i, N_j \in \mathcal{N}} a_{ik} \cdot (1 - \prod_{N_j \in \mathcal{N} \wedge y_j^k = 1} f_{ij})$$

subject to

$$\sum_{d_k \in \mathcal{D}} s_k \cdot y_j^k \le C \qquad N_j \in \mathcal{N} \tag{6}$$

$$y_j^k \in \{0,1\} \qquad N_j \in \mathcal{N}, d_k \in \mathcal{D} \tag{7}$$

The data replication problem, either optimizing the total query delay or optimizing the total availability, has been proved to be a reduction from the metric uncapacitated facility location problem in the literature [3], [20], which is known to be NP-hard. Therefore, instead of trying to find a complex algorithm that is not practical to solve or approximate the problem, we use heuristics that can provide satisfying performance with much less computation overhead.

D: Performance Analysis of Greedy Scheme

For simplicity, the data size is assumed to be the same in the analysis, i.e., $s_i = 1, i = 1, 2, ..., n$. Due to the computational complexity of the optimal scheme, we give an upper bound of the data availability by using a *super-optimal* algorithm, similar to the approach used in [3]. 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 items are S_C. Then, the data availability for the greedy scheme, denoted as A_{greedy}, is:

$$A_{greedy} \ge \sum_{d_k \in S_C} a_{ik} \tag{8}$$

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

$$A_{super} = \sum_{d_k \in S_C} a_{ik} + (1 - f_{N_i}) \times \sum_{d_k \notin S_C} a_{ik} \tag{9}$$

Therefore,

$$\frac{A_{greedy}}{A_{super}} \ge \frac{\sum_{d_k \in S_C} a_{ik}}{\sum_{d_k \in S_C} a_{ik} + (1 - f_{N_i}) \times \sum_{d_k \notin S_C} a_{ik}} \ge \frac{\sum_{d_k \in S_C} a_{ik}}{\sum_{k=1}^{n} a_{ik}} \tag{10}$$

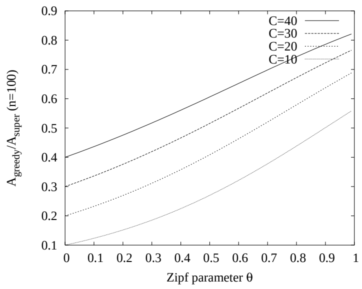


Fig. 2. The worst case data availability of the greedy scheme compared to the super-optimal scheme under different memory size C (the total number of data items $n = 100$)

Numeric Results: Figure 2 shows the numeric results that compare the worst case data availability of the greedy scheme with that of the super-optimal scheme. The node access pattern is based on *Zipf-like* distribution [21], which has been frequently used to model non-uniform distribution [22]. In the Zipf-like distribution, the access probability of the k^{th}

($1 \leq k \leq n$) data item is represented as follows.

$$P_k = \frac{1}{k^\theta \sum_{i=1}^{n} \frac{1}{i^\theta}} \tag{11}$$

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.

As shown in Figure 2, the greedy scheme performs relatively well even when compared to the 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.

E: The Details of Proposed Data Replication Schemes

The One-To-One Optimization (OTOO) Scheme

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 $\mathcal{P}$ values and each node allocates data items with the highest $\mathcal{P}$ 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) Two or more nodes may 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 to the replication scheme. This rule is also applied to other replication schemes proposed in the following. The detailed pseudo-code of OTOO scheme is provided in Algorithm 1.

Algorithm 1 OTOO

```
Procedure:
1: initiate an array $sort$ to store the sorted values;
2: $sort$=Sort();{call the sorting procedure}
   {begin to replicate data}
3: mark all the nodes as "white";
4: for $i = 1$ to $m$ do
5:     $id1$= current node id;
6:     if $N_{id1}$ is "black" then
7:         do nothing and continue to the next node;
8:     end if
9:     find the smallest id ($id2 > id1$) among all the "white" neighbors of $N_{id1}$;
10:    if $id2$ exists then
11:        calculate all the $\mathcal{P}$ values for $N_{id1}$ and $N_{id2}$ separately;
12:        sort these 2 sets of $\mathcal{P}$ values and store them in two arrays $A1$ and $A2$;
13:        get data from array $A1$ from max to min and fill them into $N_{id1}$'s memory until full;
14:        get data from array $A2$ from max to min and fill them into $N_{id2}$'s memory until full;
15:        mark $N_{id1}$ and $N_{id2}$ as "black";
16:    else
17:        do nothing and continue to the next node;
18:    end if
19: end for
   {process remaining nodes which can't find a "white" neighbor}
20: while "white" nodes exist do
21:    allocate its own most interested data from $sort$ to its memory;
22: end while
```

The Reliable Neighbor (RN) Scheme

The detailed pseudo-code of the RN scheme is provided in Algorithm 2.

Reliable Grouping (RG) Scheme

The RG scheme is outlines as follows:

1) At a relocation period, all nodes broadcast their ids and their access frequency for each data item.
2) In each set of connected nodes, starting from the node with the lowest suffix of node id, an algorithm to find biconnected components is executed in which two node N_i and N_j are considered to be connected only when they have a reliable link, i.e.,

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

where $\mathcal{T}_r$ is the same threshold value used in RN scheme. Then, each biconnected component is put to a group. If a node belongs to more than one biconnected component, i.e., the node is an *articulation point*, it belongs to only one group in which the corresponding biconnected component is first found in executing the algorithm.
3) In each reliable group, the average access probability of each data item is calculated as the priority value of the data item in the group. Similar to the OTOO scheme and the RN scheme,

Algorithm 2 RN

```
Procedure:
1: initiate an array $sort$ to store the sorted values;
2: $sort$=Sort();{call the sorting procedure}
   {begin to replicate data}
   {step 1: calculate $Cc$ value for each node}
3: for $i = 1$ to $m$ do
4:     find its reliable neighbors;
5:     calculate $Cc[i]$ value for it;
6: end for
   {step 2: allocate self-interested data}
7: for $i = 1$ to $m$ do
8:     allocate its own most interested data from $sort$ to its memory
       up to $C - Cc[i]$;
9: end for
   {step 3: fill neighbors interest}
10: for $i = 1$ to $m$ do
11:     calculate $N_i$'s $\mathcal{P}$ value for each data item;
12:     sort these $\mathcal{P}$ values and store in array $A$;
13:     while $A$ is not empty && $N_i$'s memory is not full do
14:         $data$ = current value from $A$;
15:         if $data$ doesn't exist in $N_i$'s memory then
16:             allocate it into $N_i$'s memory;
17:         end if
18:     end while
19: end for
   {step 4: fill self-interested data in case the memory isn't full
   after step 3}
20: for $i = 1$ to $m$ do
21:     allocate its own most interested data from $sort$ to its memory
        up to $C$ and make sure no repeat data allocated;
22: end for
```

the average access probability of a data item d_k should be normalized based on the data size. It is denoted as

$$\mathcal{P}_k = \frac{\sum_{i=1}^{g} a_{ik}}{g \times s_k} \tag{12}$$

where g is the total number of nodes in the group.

4) Let T_{jk} denote the demand-weighted access delay if a copy of data d_k is placed at node N_j, then T_{jk} is calculated as

$$T_{jk} = \sum_{i=1}^{g} a_{ik} \times t_{ij} \times s_k \tag{13}$$

5) Starting from the data item with the highest priority, say $\mathcal{P}_k$, select the best node, say N_i in the group to replicate the data d_k, which minimizes the total delay to access d_k for the all the nodes in the group, i.e.,

$$T_{ik} = min\{T_{xk}\} \qquad \forall \text{ node } N_x \in \text{the group} \tag{14}$$

If the memory of node N_i is full, d_k is given to the node with the next lowest T_{ik}.

6) The allocation process is repeated for all data items in the order of their average access probability until the memory of all nodes in the group are filled.

After all data items have been replicated once, there should be still memory available in the group. In this case, the allocation process starts again from the most

Algorithm 3 RG

```
Procedure:
1: initiate an array $sort$ to store the sorted values;
2: $sort$=Sort();{call the sorting procedure}
   {determine the reliable group}
3: sequential biconnected components detection
   {begin to replicate data}
4: for each reliable group do
5:     $g$ =the number of nodes in the group;
6:     calculate the $\mathcal{P}$ value of each data in the group;
7:     sort this set of values from max to min and store them into
       an array $A$;
8:     for $i = 1$ to $m$ do
9:         for $j = 1$ to $g$ do
10:            calculate the total query delay of data $d_i$ if a copy of
               $d_i$ is replicated at node $N_j$ in the group;
11:        end for
12:        sort this set of values from min to max and store their
           suffixes in an array $B_i$;
13:    end for
14:    while there is available memory in the group do
15:        repeat
16:            get data from array $A$ from max to min;
17:            $data$ = current value from $A$;
18:            $id_d$ = the suffix of $data$;
19:            get the first element in array $B_{id_d}$;
20:            $id_n$ = current value from $B_{id_d}$;
21:            if $data$ doesn't exist in $N_{id_n}$ || $N_{id_n}$'s memory is not
               full then
22:                allocate $data$ in $N_{id_n}$' memory;
23:            else
24:                check and allocate $data$ in the memory of the next
                   node from $B_{id_d}$;
25:            end if
26:        until all data items have been replicated once || memory
           is full
27:    end while
28: end for
```

frequently accessed to the least frequently access data as shown above.

F: Computation Complexity and Performance Bound of the Proposed Schemes

Computation Complexity

The proposed (OTOO, RN) schemes need to sort all the data items according to the $\mathcal{P}$ value. The computational complexity of sorting is $O(n \log n)$. This is the same as that of the DAFN scheme, although the constant factor may be higher due to the calculation of the $\mathcal{P}$ value. However, as shown in the following section, our schemes are able to provide much better performance than the DAFN scheme. The RG scheme needs to sort the estimation of the access delay of all nodes in the group for each data item and the total access probability of each data. Given the average group size is g, the computation complexity of sorting all of them is $O(n \log g + n \log n)$.

Performance Bound

The proposed schemes (OTOO, RN, RG) first sort all the data items according to the $\mathcal{P}$ value (OTOO and RN) or the data access probability (RG) and then select the most suitable data items to replicate into the memory at each iteration. Without loss of generality, we assume that the memory of all nodes are empty at start. Then as data items are replicated at the nodes at each iteration, the data access performance improves since more data items become available for access. For formal analysis, we first define a set of variables ξ_{ij}^k, where selection of a variable ξ_{ij}^k indicates that node N_i selects to replicate data d_k at its jth replication iteration[1].

Let Γ denote the set of variables that have already been selected by the node at some iteration. Then the profit (i.e., performance improvement [23]) of a variable ξ_{ij}^k with respect to Γ is denoted as $\beta(\xi_{ij}^k, \Gamma)$ and is defined as follows:

$$\beta(\xi_{ij}^k, \Gamma) = \begin{cases} Undefined & \text{if } \xi_{ij}^{k'} \in \Gamma,\ k' \neq k \\ 0 & \text{if } \xi_{ij'}^k \in \Gamma \\ \tau(\Gamma) - \tau(\Gamma \cup \{\xi_{ij}^k\}) & \text{otherwise} \end{cases}$$

where $\tau(\Gamma)$ is the performance improvement before selecting ξ_{ij}^k. The first condition of the above definition stipulates that if the jth memory page of node N_i is not empty (i.e., it has already been selected to store another data item $d_{k'}$ due to $\xi_{ij}^{k'} \in \Gamma$), then the profit $\beta(\xi_{ij}^k, \Gamma)$ is undefined. The second condition specifies that the benefit of ξ_{ij}^k with respect to Γ is zero if the data item d_k has already been stored at some other memory page j' of the node N_i. Then we can give the following theorem.

Theorem 1: The proposed schemes (OTOO, RN, and RG) can achieve a replication allocation solution whose total profit is at least half of the optimal profit.

Proof: Because the total number of memory size is C and each time the algorithm always selects one data item into the memory, the total number of replication iterations of the proposed schemes are C. Because each node selects data to replicate one by one during each replication iteration, we let Γ_l be the set of variables selected at the end of lth iteration, and let ζ_l be the set of variables added to the set Γ_{l-1} in the lth iteration, $\zeta_l = \{\xi_{1l}^{k_1}, \xi_{2l}^{k_2} .... \xi_{nl}^{k_n}\}$, which means that in the lth iteration, node N_1 decides to store data d_{k_1} in the memory, node N_2 decides to store data d_{k_2} in the memory, and so on. Without loss of generality, we can assume that the optimal solution also stores data items in all memory. Now let λ_l be the variable set $\{\xi_{1l}^{k'_1}, \xi_{2l}^{k'_2} .... \xi_{nl}^{k'_n}\}$, where $d_{k'_1}$, $d_{k'_2} ... d_{k'_n}$ is the data items stored by the optimal solution in the lth iteration. Then by the greedy choice of ζ_l, we can have

$$\beta(\zeta_l, \Gamma_{l-1}) \geq \beta(\lambda_l, \Gamma_{l-1}), \qquad \forall l \leq C. \tag{15}$$

Here the functions $\beta(\zeta_l, \Gamma_{l-1})$ and $\beta(\lambda_l, \Gamma_{l-1})$ mean that at the $(l-1)$th iteration, both schemes have the same deployment of data replication, which is Γ_{l-1}.

1. For the simplicity of the analysis, we assume each data item takes one memory page at each iteration. Therefore, the total replication process takes C iterations, where C is the maximum number of data items each mobile node can hold.

Then, let $\mathcal{O}$ be the optimal profit, and $\mathcal{R}$ be the profit of our proposed schemes. [23] has shown that a solution with optimal profit also has optimal access performance. Then we note that

$$\mathcal{R} = \sum_{l=1}^{C} \beta(\zeta_l, \Gamma_{l-1}) \tag{16}$$

Now, consider a modified network wherein each node N_i has a memory capacity of $2C$. We construct a replication allocation by taking a union of data items selected by the proposed replication scheme and data items selected in an optimal solution for each node. More formally, for each variable set $\lambda_l = \{\xi_{1l}^{k'_1}, \xi_{2l}^{k'_2} .... \xi_{nl}^{k'_n}\}$ as define above, create a variable set $\lambda'_l = \{\xi_{1l'}^{k'_1}, \xi_{2l'}^{k'_2} .... \xi_{nl'}^{k'_n}\}$ where $l' = C + l$. Obviously, the profit $\mathcal{O}'$ of the set of the variable set $\{\zeta_1, \zeta_2, ..., \zeta_C, \lambda'_1, \lambda'_2, ..., \lambda'_C\}$ is greater than or equal to the optimal profit $\mathcal{O}$. Now, to compare $\mathcal{O}'$, we add the variable sets in the order of $\zeta_1, \zeta_2, ..., \zeta_C, \lambda'_1, \lambda'_2, ..., \lambda'_C$ and add up the profits of each newly added variable set. Let $\Gamma'_l = \{\{\zeta_1, \zeta_2, ..., \zeta_C\} \cup \{\lambda'_1, \lambda'_2, ..., \lambda'_C\}$, and recall that $\Gamma_l = \{\zeta_1, \zeta_2, ..., \zeta_C\}$. We have

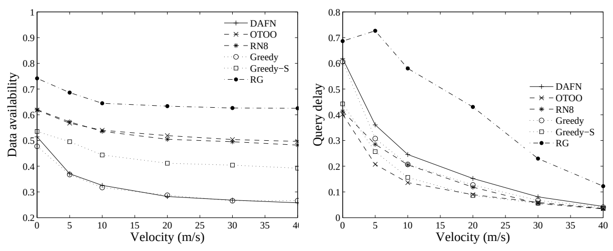
$$\begin{aligned} \mathcal{O} \quad &\leq \quad \mathcal{O}' = \sum_{l=1}^{C} \beta(\zeta_l, \Gamma_{l-1}) + \sum_{l=1}^{C} \beta(\lambda'_l, \Gamma'_{l-1}) \\ &= \quad \mathcal{R} + \sum_{l=1}^{C} \beta(\lambda'_l, \Gamma'_{l-1}) \qquad \text{According to Eq.(16)} \\ &\leq \quad \mathcal{R} + \sum_{l=1}^{C} \beta(\lambda_l, \Gamma_{l-1}) \qquad \text{Since } \lambda_l = \lambda'_l \text{ and } \Gamma_{l-1} \subseteq \Gamma'_{l-1} \\ &\leq \quad 2\mathcal{R} \qquad \text{According to Eq.(15) and (16)} \end{aligned}$$

Therefore, the proposed schemes (OTOO, RN, and RG) can achieve a replication allocation solution whose total profit is at least half of the optimal profit. Theorem 1 holds.

We can also show that our proposed schemes (OTOO, RN, and RG) is an improvement of the 20.5-approximation result of [20] when the optimal access delay is at least $1/40$ of the total access delay, without the replications.

Theorem 2: If the access delay without data replication is less than 40 times of the optimal access delay using optimal data replication allocation, then the total access delay of the proposed schemes is less than 20.5 times the optimal access delay.

Proof: Let the total access delay without replications be $\mathcal{W}$, and the optimal access delay (using optimal replication allocation) be $\mathcal{O}$. Thus the optimal profit is $\mathcal{W} - \mathcal{O}$. Since the profit of the proposed schemes is at least half of the optimal profit, the total access delay of the proposed schemes is at most $\mathcal{W} - (\mathcal{W} - \mathcal{O})/2$, which is at most $20.5\mathcal{O}$. Theorem 2 holds.



(a) Data Availability (b) Query Delay

Fig. 3. Performance as a function of velocity when nodes have different access pattern

G: Simulation Results Based on Different Node Mobility Models

To evaluate the effects of mobility, we first investigate the performance of replication schemes under a single mobility model to see the effects of mobility. Then, we compare the performance of different schemes under various mobility models to identify the relationship between data replication and node mobility.

We first compare different data replication schemes using the Random Walk (RW) model [24], which is commonly used in MANETs. In RW, at every time unit, each mobile node randomly determines a movement direction, and randomly determines a movement speed from 0 to V m/sec.

During the initial short time period, all nodes are static so that some initial knowledge can be built up. Later, all nodes start moving, and in each data replication interval, they replicate data based on different schemes. When the replication starts, the node gathers some knowledge about the replica distribution and store such information locally. Through simple flooding, each node will know which data item or replica it can access at that moment. The replication is performed only based on the information collected at the starting time. Since the network environment keeps changing because of node movement, and the information is not updated between two replication intervals, the data replication may not be optimal between two intervals. When nodes are moving, the query sent out based on the original knowledge may be invalid, e.g. the cooperating neighbors may move out of range or even become disconnected. When the moving velocity is not very high, the destination node and all the intermediate nodes on the path can still be accessed within the radio range or through some other relay nodes.

Figure 3 shows the performance when the moving velocity changesApparently, the data availability drops as the nodes move faster, because the replication schemes rely on the collaboration between neighbors. When the nodes move faster, more neighbors may become unreachable. However, when mobility is already high and further increasing the moving speed (e.g. 20m/s to 40m/s), has less impact. This is because when the node velocity is high enough, most neighbors who help to store the data have already moved out of the transmission range, thus further increasing the velocity does not increase unreachable neighbors too much. Compared to other schemes, RN8, OTOO, Greedy-S and RG are less sensitive to mobility. This is because OTOO, RN8 and Greedy-S are proposed in order to serve more queries by the local cache, and hence they are less dependent on the cooperation of the neighbors. RG aims to allocate the data evenly among nodes in the network, thus it is also less dependent on the change of network topology. Therefore, our schemes not only help reduce the data access delay, but also keep data availability when mobility increases.

In Figure 3 (b), the decrease of the query delay is at the cost of data availability. As most of the data is accessed from the local cache, the average delay becomes smaller. The delay of the DAFN scheme drops much faster. This can be explained as follows. Normally, in the DAFN scheme, most of the data (around 50%) is accessed from other nodes, which contributes the major part of the delay. However, due to mobility, their delay is not counted since these data are most likely unaccessible.

The performance of the OTOO, RN, and RG scheme can be further improved by using the moving velocity as a factor to estimate the link failure probability f_{ij}, e.g. larger velocity increases the value of f_{ij}. Then nodes in the OTOO scheme will become more selfish, and RN and RG will have less number of reliable neighbors or reliable groups as the mobility becomes higher. This can reduce the chance of contributing memories to the nodes that are not accessible.

Besides the RW mobility model, we also compare the performance of different data replication schemes under other three typical mobility models: Random Waypoint (RWP) [25], Manhattan Mobility (MM) [26], and Reference Point Group Mobility (RPGM) [27].

TABLE 1
Data availability when nodes have different access pattern

	RW	RWP	MM	RPGM
DAFN	0.5055	0.2723	0.5821	0.7342
OTOO	0.5398	0.4510	0.6231	0.7472
RN8	0.5356	0.4563	0.7293	0.8237
Greedy-S	0.4436	0.5021	0.5628	0.6830
RG	0.6446	0.4672	0.7123	0.9221

TABLE 2
Query delay when nodes have different access pattern

	RW	RWP	MM	RPGM
DAFN	0.2456	0.1743	0.4743	0.7452
OTOO	0.1363	0.1832	0.4275	0.7370
RN8	0.2056	0.2231	0.4274	0.7453
Greedy-S	0.1563	0.1323	0.3429	0.7283
RG	0.58031	0.5832	0.7692	0.7688

In RWP, each node remains stationary for a pause time S seconds. Then, it selects a random destination in the entire area and moves to the destination at a speed determined randomly between 0 and V m/sec. After reaching the destination, it pauses again, and then repeats this process. In this model, mobile nodes tend to gather at the center of the area. The MM mobility mode emulates the node movement on streets where nodes only travel on the pathways in the map. Manhattan grid maps of horizontal and vertical streets are used to restrict the node movement. On each street, the mobile nodes move along the lanes in both directions. At each intersection, the mobile nodes choose their directions and speed (0 to V m/sec) randomly. RPGM is used to model group mobility. Each group has a logical "center" called a reference point and group members (nodes). Each reference point moves according to the RWP model with V m/sec (maximum speed). In each group, nodes are uniformly distributed within a certain radius from the reference point.

Table 1 and Table 2 show the system performance of different data replication schemes under various mobility models. Since both Greedy-S and Greedy schemes are based on the same greedy idea, and Greedy-S always outperforms Greedy, in the following, we use Greedy-S as the representative scheme for the greedy replication.

As for the RW mobility model, we can see that OTOO, RN, and RG achieve higher data availability than other two schemes. This benefit comes from the small mobility of the RW mobility mode, wherein closely connected nodes have relatively stable connectivity. Therefore, the cooperative data replication schemes such as OTOO, RN, and RG have better performance. Here we node that although DAFN also considers cooperation between neighboring nodes, its data availability is still lower than other schemes, because it ignores the possible link failure and relies too much on cooperation. In the RWP mobility, there is no mobility similarity among neighboring nodes and no reliable connectivity existing. In this case, the Greedy-S scheme outperforms other schemes. In the MM and the RPGM mobility models, due to the road layout constraint and the restricted mobility pattern, the network has relatively higher density from node perspective. As expected, more reliable and larger partitions can be formed in MM and RPGM compared to RW and RWP. Therefore, the data availability and query delay become higher in the MM model and the RPGM model. In the RPGM model where nodes move following strict group mobility, RG outperforms others.

H: Discussions

Due to the complexity of the problem, we have to make some assumptions similar to most existing works in this area. Some of them are further discussed as follows.

First, we assume that mobile nodes have limited memory space. This assumption is used extensively in existing works that study data access related issues. Even though the concern of limited memory may have been alleviated to some extent with the recent developments in storage technology, the memory constraint still exists.

Second, how to determine the link failure probability is very complicated. In this paper, for simplicity, we assume that the failure probability depends on the wireless signal strength which decreases with a rate between the square order of the distance to the signal source. Although this assumption has been used extensively in the literature, we are still looking for better solutions which consider more factors such as node mobility, wireless channel interference, etc, to provide more accurate predictions.

Third, we use our simulator to generate mobility and data access models. As future work, we also consider using other more widely used simulators.

Finally, our solution can be extended along various directions. For example, we did not take into account the relationship among different data items, the data update issue, and the data discovery issue. Data semantics can be used to improve the performance of query, and hence we can apply semantic based technique [28] to further improve the performance of our solutions. Similarly, techniques used in [29]–[31] and [32] can be used to extend the proposed scheme to handle data update, data consistency or data discovery issues.

References

[1] W. Zhang, G. Cao, and T. L. Porta, "Data dissemination with ring-based index for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 7, pp. 832–847, 2007.

[2] B. Sheng, Q. Li, and W. Mao, "Data storage placement in sensor networks," *ACM MobiHoc*, pp. 344–355, 2006.

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

[4] H. Yu and A. Vahdat, "Minimal replication cost for availability," *ACM Symposium on Principles of Distributed Computing (PODC)*, 2002.

[5] 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.

[6] P. Padmanabhan, L. Gruenwald, A. Vallur, and M. Atiquzzaman, "A survey of data replication techniques for mobile ad hoc network databases," *The VLDB Journal*, vol. 17, pp. 1143–1164, 2008.

[7] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," *IEEE INFOCOM*, 2001.

[8] T. Hara and S. K. Madria, "Data replication for improving data accessibility in ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 11, pp. 1515–1532, 2006.

[9] M. Jorgic, I. Stojmenovic, M. Hauspie, and D. Simplot-Ryl, "Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks," *The Third Annual Mediterranean Ad Hoc Networking Workshop*, pp. 360–371, 2004.

[10] J.-L. Huang and M.-S. Chen, "On the effect of group mobility to data replication in ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 5, pp. 492–507, 2006.
[11] K. Wang and B. Li, "Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks," *IEEE INFOCOM*, 2002.
[12] T. Hara, "Quantifying impact of mobility on data availability in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 241–258, 2010.
[13] V. Ramany and P. Bertok, "Replication of location-dependent data in mobile ad hoc networks," *ACM MobiDE*, pp. 39–46, 2008.
[14] M. Boulkenafed and V. Issarny, "A middleware service for mobile ad hoc data sharing, enhancing data availability," *ACM Middleware*, 2003.
[15] G. Cao, L. Yin, and C. Das, "A cooperative cache based data access framework for ad hoc networks," *IEEE Computer*, Feb. 2004.
[16] M. Fiore, F. Mininni, C. Casetti, and C. Chiasserini, "To cache or not to cache?" *IEEE INFOCOM*, 2009.
[17] Y. Du, S. Gupta, and G. Varsamopoulos, "Improving on-demand data access efficiency in manets with cooperative caching," *Ad Hoc Networks*, vol. 7, pp. 579–598, May 2009.
[18] L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *IEEE Transaction on Mobile Computing*, vol. 5, no. 1, January 2006.
[19] J. Zhao, P. Zhang, G. Cao, and C. R. Das, "Cooperative caching in wireless p2p networks: design, implementation, and evaluation," vol. 21, no. 2, pp. 229–241, February 2010.
[20] I. Baev and R. Rajaraman, "Approximation algorithms for data placement in arbitrary networks," *ACM-SIAM SODA*, pp. 661–670, 2001.
[21] G. Zipf, "Human behavior and the principle of least effort," *Addison-Wesley*, 1949.
[22] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," *IEEE INFOCOM*, 1999.
[23] B. Tang, H. Gupta, and S. Das, "Benefit-based data caching in ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 3, pp. 289–304, March 2008.
[24] K. Pearson, "The problem of the random walk," *Nature*, vol. 72, no. 1867, p. 342, 1905.
[25] J. Broch, D. Maltz, D. Johnson Y. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *ACM MOBICOM*, pp. 85–97, October 1998.
[26] F. Bai, N. Sadagopan, and A. Helmy, "Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks," *IEEE INFOCOM*, 2003.
[27] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A group mobility model for ad hoc wireless networks," *ACM Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 53–60, 1999.
[28] Q. Ren, M. Dunham, and V. Kumar, "Semantic caching and query processing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, pp. 192–210, 2003.
[29] T. Hara, "Replica allocation in ad hoc networks with periodic data update," *International Conference on Mobile Data Management (MDM)*, 2002.
[30] T. Hara and S. Madria, "Consistency management strategies for data replication in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 7, pp. 950–967, 2009.
[31] J. Cao, Y. Zhang, G. Cao, and L. Xie, "Data consistency for cooperative caching in mobile environments," *IEEE Computer*, vol. 40, no. 4, pp. 60–66, 2007.
[32] F. Sailhan and V. Issarny, "Scalable service discovery for manet," *Pervasive Computing and Communications, IEEE International Conference on*, vol. 0, pp. 235–244, 2005.