# Minimizing Query Delay in Load Balanced MANETs through Data Replication

Supritha N, Mtech(CSE), IV sem, APSCE, Bangalore
Mr.C.P.Sameerana, Asst.Prof and HOD, Dept. of CSE,APSCE

**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 and 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).**

## I INTRODUCTION

In ad hoc networks [1], 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 neighbours. Otherwise, the nodes communicate through multi-hop routing. 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 can increase the data accessibility and reduce the query delay if there is 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 trade-off 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 neighbouring 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 neighbours.

The remainder of the paper is organized as follows. In section II, we introduce some related works and compare them with our proposed methods. In section III, we propose replica allocation methods. In section IV, we show the results of simulation experiments. Finally, in section V, we summarize this paper.

## II RELATED WORK

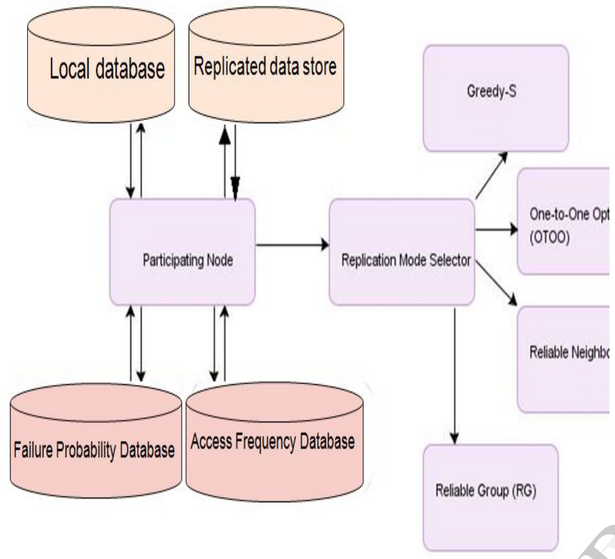
Data replication has been extensively studied in the Web environment [3], [4], where the goal is to place some replicas of the web servers among a number of possible locations so that the query delay is minimized. In the Web environment, links and nodes are stable. Thus, the performance is mainly measured by the query delay.

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. 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.

## III THE PROPOSED DATA REPLICATION SCHEMES

Fig 3: Architectural design



The following notations are used in this paper.
· $\mathcal{N}$: set of mobile nodes in the network.
· $m$: the total number of mobile nodes

· $\mathcal{D}$: the set of available data items

· $n$: 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.
· $t$ : the delay of transmitting a data item of unitsize 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 $d_j$ .

Heuristics 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 degree of cooperation affects both the data availability and the query delay.

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. 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.

Greedy Data Replication Scheme:

One naive greedy data replication scheme is to allocate the most frequently accessed data items until the memory is full. It considers data size difference between data items. It calculates data access frequency of a data item $d_k$ by normalizing it against the data size, i.e $a_{ik} / s_k$ This scheme 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. The following sections present schemes that apply different levels of cooperation between neighboring nodes following our heuristics.

The One-To-One Optimization 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. Node $N_i$ calculates the combined access frequency value of $N_i$ and $N_j$ to data item $d_k$ at $N_i$, denoted as $CAF^k_{ij}$ by using following function:

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

Similarly node $N_j$ calculates its combined access frequency to data item $d_k$ by using the following function:

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

If link failure probability between two nodes is low then data replication is less likely to be done but if it's high then replication is done. 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 neighbouring 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 is stable, they can have more co-operations 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, $\mathcal{N}i$ selects the next best candidate (data item) according to the replication scheme.

OTOO considers only one neighbouring node when making data replication choices. However, it still considers its own access frequency as the most important factor because the access frequency from a neighbouring node is reduced by a factor of the link failure probability.

The Reliable Neighbour 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 a node $N_i$ a neighboring node $N_j$ is considered to be $N_i$'s reliable neighbor if

$$(1- \ ) > \mathcal{T}_r$$

where $\mathcal{T}_r$ is a threshold value and $f_{ij}$ is link failure probability.

The total contributed memory size of $\mathcal{N}i$, denoted as $(i)$, is set to be

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

where $\alpha$ is a system tuning factor which affects the memory allocated to itself and its neighbours. Intuitively, $\mathcal{N}i$ contributes more memory if its links with neighboring nodes are more stable. The two extreme cases are: 1) when $(i) = \mathcal{C}$, $\mathcal{N}i$ contributes all its memory to hold data for neighboring nodes; 2) when $fij = 1$, $\forall \mathcal{N}j \in (i)$, $\mathcal{N}i$ does not contribute any memory. The reason behind the RN scheme is that when links to neighboring nodes of $\mathcal{N}i$ are stable, $\mathcal{N}i$ can hold more data for neighboring nodes as they also hold data for $\mathcal{N}i$. 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 $\mathcal{N}i$ first allocates its most interested data to its memory, up to
$\mathcal{C} - Cc\ (i)$ memory space. Then all the rest of the data are sorted according to $\mathcal{P}$ to a list called the neighbour's interest list. The $\mathcal{P}$ value of $\mathcal{N}i$ to $dk$ is defined as:

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

The memory space of $(i)$ is used to allocate data with the highest $\mathcal{P}$ values. There may be some overlap between $\mathcal{N}i$'s interested data and the allocated data interested by $\mathcal{N}i$'s neighbours. 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 neighbour's interest list is chosen instead. However, co-operation among the nodes is not exploited to the maximum extent in this scheme.

The Reliable Grouping Scheme: To achieve maximum degree of co-operation between the mobile nodes, the reliable grouping scheme shares replicas in large and reliable groups of 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 query delay within the group. There is no redundant replication until every data item is replicated at least once and so a maximum degree of cooperation within the reliable group is achieved. RG aims to allocate the data evenly among nodes in the network, thus it is also less dependent on the change of network topology. 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. Therefore, the proposed reliable grouping scheme not only helps to reduce the data access delay, but also ensures data availability when mobility increases.

The reliable grouping scheme works as follows:

1. At a relocation period, all nodes broadcast their ids and access frequencies.
2. Biconnected components with reliable links are detected and put into groups.
3. In each reliable group, the average access probability of each data item is calculated to get its priority value in the group and then it is normalised based on data size. It is denoted as

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

where $g$ is the total number of nodes in the group.
4. $T_{jk}$ denotes the demand-weighted access delay for a copy of data $d_k$ placed at a node $N_j$. It is calculated using following equation:

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

5. Starting from the data item with the highest priority, the best node in the group say $N_i$ with smallest delay to access $d_k$ is selected to replicate the data .i.e,

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

Here, if memory in $N_i$ becomes full, $d_k$ is given to node with 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 frequently accessed to the least frequently access data.

## IV SIMULATION RESULTS

Simulation is performed using Network Simulator 2.35. At the beginning of the simulation, $m$ nodes are placed randomly. The radio range is set to be $D$. If two nodes $Ni$ and are within the radio range (i.e., $(i, j) < D$), they can communicate with each other. The proposed schemes do not depend on the failure model and they are able to work as long as the failure probability between neighboring nodes can be estimated. 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 $Ni$: $offset_i$, which is between 1 and $n-1$.

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. The performance metrics used in the simulation are mainly data availability and query delay.

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 Fine tuning $\mathcal{Tr}$ and Fine tuning β.

### Fine-tuning $\mathcal{Tr}$

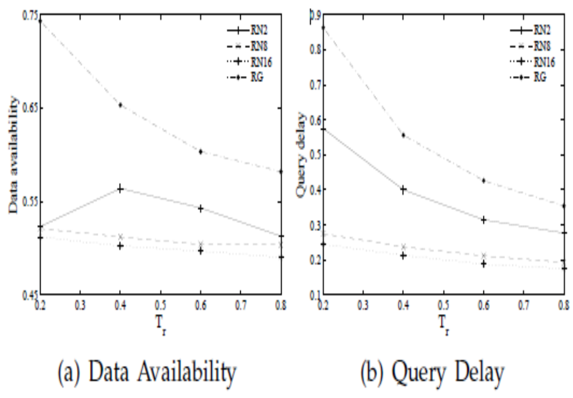


Fig5: Performance of RN and RG as a function of $\mathcal{Tr}$ when nodes have the same access pattern.

Larger $\mathcal{Tr}$ results in smaller number of cooperative neighbours, and vice-versa. We can see that $\mathcal{Tr}$ 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 neighbours. The performance of the RG scheme is also affected by $\mathcal{Tr}$, because the change of $\mathcal{Tr}$ affects the number of nodes in a reliable group. When $\mathcal{Tr}$ is around 0.6, all schemes have relatively stable performance, which means the change of $\mathcal{Tr}$ does not have significant effect on the relative performance of different data replication schemes. Thus, we use $\mathcal{Tr} = 0.6$ in the following.

### Fine-tuning β

By controlling $\beta$, the link failure probability can be adjusted. When the link failure probability deceases, data availability increases as shown in Figure 6. 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. 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.

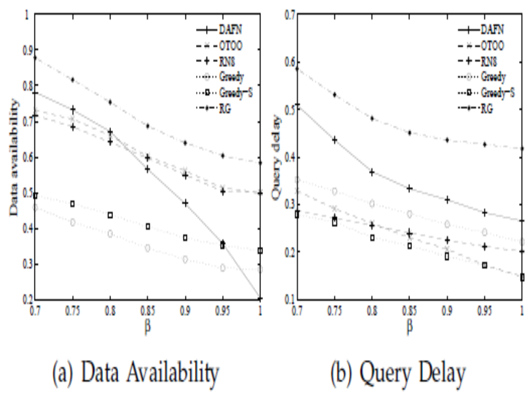


Fig6: Performance as a function of $\beta$ when nodes have the same access pattern

Comparing RN2, RN8, RN16, OTOO, and RG, we find that the relation of their data availability is as follows:
$RG > RN2 > RN8 > RN16 \approx OTOO$ (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.

## V CONCLUSION

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 neighbour's memory when the communication link to them is reliable. 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 trade-off between data availability and query delay.

## REFERENCES

[1] "MOBILE ADHOC NETWORKING", Carlos de Morais Cordairo and Dharma.P.Agarwal, OBR research center for distributed and mobile computing, University of Cincinnati.
[2] "Balancing the Tradeoffs between Query Delay and Data Availability in MANETs" IEEE transactions on parallel and distributed systems. January 2012.
[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, 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 adhoc 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.