

Applicability of sub graph centrality to improve data accessibility among peers in MANETs

M. Pushpalatha · T. Ramarao · Revathi Venkataraman

Received: 13 April 2012 / Accepted: 13 November 2012 / Published online: 4 December 2012
© Springer Science+Business Media New York 2012

Abstract Data sharing plays an essential role in many of the mobile ad-hoc network (MANET) applications that exhibits collaborative behavior. In such applications, replication is used as a foremost and fundamental technique to improve data availability. However, due to the dynamic nature of the network, data replication becomes more intricate in MANET. To alleviate this problem, we have proposed a mechanism which not only enhances data accessibility, replicates data in a minimum number of nodes, relocates shared data on the prediction of mobility of replica holder and in addition, data can be accessed by any node in a minimum number of hops. In our approach, we have prefaced mathematical concept known as minimum dominating set and sub graph centrality principle to decide the number of replicas both in static and dynamic environment. Simulation results when compared with the existing mechanisms shows that the response time or data access delay is reduced, client can access the data from the server in a minimum number of hops, and consequently the number of forwarded messages to access the data are greatly reduced thus making our network energy efficient.

Keywords Data replication · Mobility prediction · Minimum dominating set · Sub graph centrality

M. Pushpalatha (✉) · R. Venkataraman
Department of Computer Science and Engineering,
SRM University, Chennai, India
e-mail: lathamardappa@yahoo.co.in

R. Venkataraman
e-mail: revathivenkat@yahoo.com

T. Ramarao
Department of Telecommunication and Engineering,
SRM University, Chennai, India
e-mail: ramarao@ieee.org

1 Introduction

Mobile ad-hoc networks (MANETs) are termed as infrastructure less networks, which consist of a collection of mobile hosts moving in an arbitrary manner [1]. As per the mode of operation, ad hoc networks can also be defined as peer to peer multihop network [2–4] where nodes can communicate directly with other nodes within single hop or multihops and the nodes amid of hops can act as a router. Nodes are said to be peer to peer as they can act both as a client or server to provide or acquire service. The key characteristics such as self organization and decentralized nature had lead to several similarities between peer to peer (P2P) network and MANET. Such similarities are stated as follows: i) Dynamic network topology (node leave or join in P2P and mobility in MANET), ii) hop/hop communication and additional similarities with unstructured P2P networks are flooding based routing and limited scalability. Due to this synergy, many thoughts proposed for MANET are fit for P2P overlay or vice versa.

MANET finds its use in a variety of applications that include but not confined to military services, rescue and emergency operations, disaster relief and so on [5]. In all these applications, sharing data in a collective manner is vital and also acquiring data are time critical. However, the challenging feature of MANET, especially its dynamic behavior subsidence the data availability and increases the data access delay to accomplish a certain task.

To facilitate data sharing and to increase data availability, the data is replicated across multiple nodes [6]. However, this phenomenon becomes undesirable due to the following challenges as discussed in [7–8]:

- *Node Mobility:*

Server Mobility: As the server moves, it may partition the network and hence the client may not be able to access the data or it may travel several hops to access the data.

Client Mobility: Movement of a client from one location to another may lead to high data access delay because it has to traverse maximum number of hops to reach its server.

- *Power Consumption:* All mobile devices in the MANET are battery powered. If a node with less power is replicated with many frequently accessed data items, it soon gets drained and cannot provide services any more.
- *Resource Availability:* Since nodes participating in MANET are portable devices, memory capacity is limited.
- *Response Time:* Time is the crucial factor for many of the applications like rescue operations and military operations. Hence response time, i.e. the time taken for the the client to access the data must be minimum.
- *Replica Relocation:* This topic addresses the issues related to when, where, who and how replicas are allocated. Due to dynamic topology, static allocation of replicas is not possible.
- *Consistency Management:* If the data shared is read only, performance can be improved by fully replicating the data in all the nodes. But if a replica is frequently updated, other replicas becomes invalid.

To counteract the above-mentioned challenges, effective measures are mandatory to develop an efficient replication algorithm. The effectiveness of replication schemes highly be sustained by the number of replica to be created in the system, where to place the replica and when to relocate replica on nodes. Hence to develop a good data replication algorithm the following are the salient features:

- A replication mechanism must determine when the server node will move out of range and replicate data to appropriate node on beforehand.
- The replication algorithm should also consider client mobility and make the client to access the data from nearby servers.
- A replication procedure should be energy aware and relocate the replica accordingly.
- The algorithm has to check whether a node has sufficient memory capacity to hold the replica before sending the replica to that node.
- Data must be replicated to nodes that are nearer to the clients to improve the response time.
- Replicas must be relocated dynamically to improve data availability.
- A mechanism is required to manage the consistency of data in the network.

We have proposed a technique that consists of two phases. The first phase is the Initialization phase, and the second phase is the Maintenance phase. In Initialization phase, the dominating nodes are identified and minimized to hold the replica of the data using the minimum dominating set concept in

graph theory. This method is suitable for a static network, and the attractive feature of this concept is that each node can access the data in at the most one hop. The nodes that are chosen as dominators act as a server and the remaining nodes act as a client. However, the same concept cannot be seductive for dynamic networks because the unstable node may be designated as a new dominator to relocate the replica if the old dominator moves out of range. As an alternate, in maintenance phase, we have used sub graph centrality principle to identify a stable node and relocate the replica if a server node moves out of range from its client. To the best of our knowledge, our approach is the first approach to introduce a sub graph centrality principle to distribute replica in a dynamic network. Our mechanism solves most of the confrontations discussed here by replicating data closer to the clients, by considering the mobility of nodes and relocating the replica appropriately on prediction of mobility of a server node.

The structure of paper is as follows: Section 2 discusses the related work. In Section 3, we discuss some mathematical concepts related to our work; Section 4 gives the details of the proposed model. Section 5 represents simulation results and analysis and finally, we conclude our work.

2 Related works

In the paper proposed by [9], the goal of the data replication technique is to address the issue of data access delay, i.e. to have a minimum number of data servers and access the data in minimum number of hops. They have introduced three approaches for replicating data named as a centralized approach (CEN), distributed approach with no status information, distributed approach with status information. In CEN, the algorithm is run by a coordinator to identify the data server in the network. A coordinator can be any host which determines the number of hosts covered by each host in “k” hops using controlled flooding message. Any host who covers a maximum number of hosts will be the data server and thereafter the hosts who are in its coverage will not participate in data server selection. The remaining hosts are arranged in the decreasing order based on the number of hosts they cover. Data server selection process continues until there is no remaining host. This method of allocation reduces the access delay since the client can access the data in at most “k” hops, however, it is unsuitable for a dynamic behavior because the data accessibility will become low if the server moves away from its client. To resolve this, the authors have proposed two other approaches that take care of the dynamic behavior in the network. First method, consist of two phases: initialization phase and updating phase. Each host generates a random number and if the random number is less than a threshold the host can hold the data items; otherwise they remain as a regular host. In

updating phase, if each host moves away, they should get the data from the data server who are more than “ k ” hops away and announces itself as data servers to the entire host within “ k ” hops. The next method is slightly modified to avoid redundancy of servers and maintain minimum number of server. To accomplish this, each data server within the range can exchange their member information to other server. If data servers have same set of members, one data server can give up its role and can become a regular host. However, all these approaches discussed in this paper have no guarantee on the degree of replicas maintained. Lots of messages are flooded to identify the data servers and to avoid redundancy of servers. Due to these overheads, these approaches cannot perform well in scalable networks.

A distributed and decentralized algorithm for dynamic data replication has been proposed in [10] named as two phase replication approach (TPRA). The aim of this replication is to ensure high data availability. Two algorithms are proposed termed as Primary replication algorithm and Dynamic replication algorithm. In primary replication, the node that wish to replicate its data, distributes it data to a node that are three hops away and the new replica holder further distributes the data to a node that is three hops away. This process continues till no nodes are found in the network. In dynamic replication, the replica is redistributed based on a parameter called as access rate. If a node accesses a data frequently, the replica of the data is redistributed to it in order to save the energy and reduces the traffic. This node then notifies the other nodes in the range that it is designated as a new data server. If the access rate of a node is low then the client have to access data from the server that are $k > 1$ hops away. Although this method distributes the replica uniformly, the replica distribution is based on the location of primary server which could not be a fair approach in a situation where client may take several hops to reach a server. The distance estimation between the servers is three hops away which is a weak assumption and cannot be a good estimation for large scale or dense networks. Because this approach may increase the number of replicas. Moreover in dynamic replication, redistribution of replica is based only on the access rate of the data by a client and has not considered the other issues that exhibit the dynamic behavior of a node.

In this paper [11], authors have applied the Eigen vector centrality (EVC) principle to determine the best node for replication of shared data. Their goal is to find whether EVC calculations can give good measure for determining the host nodes to replicate data items. The authors have calculated EVC for three types of connectivity matrices - Weighted Connected matrix, unweighted connected matrix and weighted normalized connectivity matrix. They have proved that using different types of connectivity matrix to calculate EVC does not make much difference in the performance of the network. Also, they have proved that choosing a group of higher EVC nodes as replication points is not a good

measure because according to the definition of EVC, if a node has high EVC then the neighbor node also has high EVC. Therefore the replication points chosen are close to each other most of the times and hence decrease the data accessibility of the client that are far way.

Hara et al. [12] have introduced the concept of the stability of radio links to allocate replicas on connected mobile host in the whole network. The initial distribution of replica takes place by using static access frequency method. After applying SAF, replicas of every pair of neighboring nodes are compared to eliminate duplication. The authors have proposed two methods DAFSN -S1 and DAFSN-S2. In DAFSN-S1, for each pair of nodes, the link stability is computed and compared with a threshold. If the link stability is greater than a threshold, the link between the nodes is strong and replicas are compared for each pair of nodes. If duplicate exist and if the data is not a original it is replaced by new data whose access frequency is high compared to other data items. Else if the stability of link is not strong the node pair is ignored. In the second method, in addition to the stability of the link, another parameter is included to check the probability that the this host will access the data even if it is disconnected from the other host. If so, the duplication will not be eliminated else it will be removed. In this method, despite its efficiency in allocating replica, more computation is involved in eliminating the redundancy of replicas which may cause unnecessary delay in accessing the data.

The work proposed in [13] addresses the problem of selfishness in the context of replica allocation in MANET, i.e., selfish nodes may not share its own memory space to store replica for the benefit of other nodes. The proposed strategy consist of three parts: 1) Detecting selfish nodes, 2) Building the self centered friendship SCF-tree and 3) allocating replica. Each node in the network detects the selfish nodes based on credit risk scores and then makes its own topology (partial) and builds its own SCF-tree by excluding selfish nodes and finally allocates replica in a distributed manner. Despite the fact that the authors have proposed replica allocation algorithm in a new dimension, the algorithm incurs lot of computation in detecting selfish nodes and constructing SCF-tree. Such type of replication is not applicable for time critical applications.

The paper proposed in [14], discuss new data replication techniques to balance the trade-offs between data availability and query delay. They have proposed four replication schemes named as, the greedy data replication scheme, one to one optimization (OTOO), Reliable Neighbor scheme (RN) and Reliable Grouping scheme (RG). In greedy scheme, each node picks the data item with large access frequency and small data size. This scheme does not consider the cooperation between the neighboring nodes and hence its performance may be limited. In the next scheme, each mobile node only cooperates with at most one neighbor and calculates the combined access frequency to decide which data to replicate. If the probability

of link failure between two nodes is less and if the neighbor contains the frequently accessed data, the node is less likely to replicate this data. Otherwise it replicates the data locally. The next scheme is the RN scheme which further increases the cooperation to replicate data for more number of neighboring nodes. Each node checks its neighbor reliability by calculating the link failure probability. If the probability of link failure is less, then the node finds the neighbor data interest and replicate them. To further increase the cooperation of nodes, RG scheme has been proposed, in which replicas are shared in large groups. This scheme selects the most suitable data items and places them in the best node. The node selection is based on the function that considers access delay between the query node and the nearest replication node. This paper focus on allocation of frequently accessed data near to the clients, but has not considered the stability of a node to place the replica.

When compared to all the above methods, our proposed mechanism maintains minimum replica degree, no extra computation is involved in removing duplicates, uses best centrality measure to relocate the replica in a dynamic network by considering the issue discussed in [11] and the number of communication messages to access data are reduced. In addition, replica distribution is a client based approach i.e. allocating replica nearer to the clients, thus the data access latency is low where the client can access the data in at most one or two hops.

3 Preliminaries

3.1 Dominating set

In graph theory [15], dominating set for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is joined to at least one member of D by some edge. The *domination number* $\gamma(g)$ is the number of vertices in a smallest dominating set for G . These vertices are named as *dominators*. A minimum dominating set (MDS) is a dominating set of minimum cardinality.

Constructing a minimum dominating set will help the nodes to access the data in at most one hop. However, computing a minimal size dominating set is NP_hard in general, but efficient approximation algorithms make the problem NP_complete. In our model, we have used binary integer programming, an approximation technique to find the minimum dominating set to distribute the replica.

3.2 Centrality principles

Centrality measure is a substantial measure for network analysis. Centrality determines the significance of a node within the graph. Here, we have defined immensely used centrality measures named as *Degree centrality*, *Eigen*

vector centrality and also a new centrality measure introduced in our proposed model to identify the backbone nodes in the network.

3.2.1 Degree Centrality (DC)

It is defined as number of links made up to a node or simply the number of nodes attached to a node. It assumes every connection to a node is equal which is not true in mobile ad hoc networks.

3.2.2 Eigenvector Centrality (EC)

Eigen Vector Centrality is defined as the principle or dominant Eigen vector of the adjacent matrix A and it is better interpreted as a sort of extended degree centrality which is equivalent to the sum of the centrality of the node neighbors. Consequently, a node has a high value of EC either if it is connected to many other nodes, or if it is connected to others those themselves has high EC [16]. Due to its definition, high EC nodes are close to each other most of the times i.e. they are neighbors to each other. Hence, this principle cannot be used to select nodes to replicate data in order to increase the data accessibility.

3.2.3 Sub Graph Centrality (SC)

SC defined as a new centrality measure [16] that differentiates the participation of each node in all sub graphs in a network. Small sub graphs are given more significance than larger ones. The sub graph centrality can be obtained mathematically from the spectra of the adjacency matrix of the network. The advantage of this method is, it can discriminate nodes of a network than the alternative measures such as Degree centrality, Closeness, Betweenness and Eigen vector centrality. This centrality measure helps us to select the replication points that are extensively distributed in the network thus increasing the data accessibility.

3.2.4 Significance of sub graph centrality in accessing the data in ad hoc networks

To show that subgraph centrality measure play a significant role in accessing data in MANET, we empirically investigate by computing subgraph centrality and other two common measures such as Eigen vector centrality and Degree centrality of each node in the network. These measures are calculated for different cluster size at different instant of time by varying the connectivity of nodes. The nodes are then ranked according to the values computed. From the Fig. 1 it was observed that selecting top 10 % ranked nodes can serve upto 80 to 90 % of clients in SC, whereas EC and DC can only serve upto 40 % and 70 % of clients. Similarly,

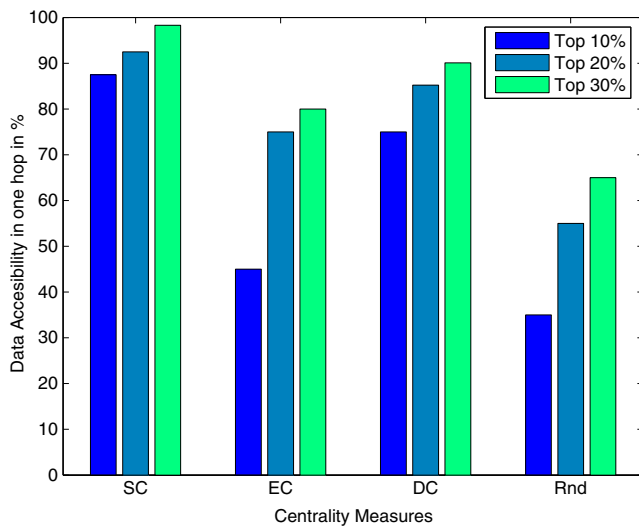


Fig. 1 Performance of Centrality measures in accessing the data in one hop in accessing the data

selecting top 20 % and 30 % ranked nodes can serve 90 – 100 % of clients in SC, while nodes selected using EC and DC can serve 70–80 % and 80–90 % respectively. Hence using subgraph centrality increases data accessibility because they select nodes that are evenly distributed and can cover much number of clients in the network in minimum number of hops. As a result, sub graph centrality will be a good estimate for selecting replication points in dynamic networks. The Table 1 demonstrates the centrality measures calculated for ten nodes at a particular time instant. All three measures show a distinct behavior and it is observed that both EC and DC serve less number of nodes compared to SC.

Besides, to prove that the subgraph centrality exhibit distinct behavior when compared to EC and DC, we have examined several data sets that consist of SC, EC and DC measures for each node computed for different network size and at different time epoch. The correlation between these measures are found which will be in the range of $-1 < 0 < +1$. The measures are said to be highly correlated if the correlation value is $+1$. Table 2 depicts the correlation among the centrality measures and it shows that SC is

uncorrelated with EC and moderately correlated with DC. This is due to the fact that, in some observations SC and DC are redundant i.e., the ranking of nodes are same.

3.3 System model

We model our topology as a finite connected undirected graph $G(V, E)$, a randomly deployed topology where V represent the number of nodes and E represent the number of edges between the nodes. Each node in the network is identified by unique identity number and designated as a client or server. The node is said to be a server if it holds the shared data or replica of the shared data. Otherwise the node is said to be a client to access the shared data.

3.4 Assumptions

The following are the assumptions made in our model:

- Each node in the system is assigned a unique node identifier $N = N_1, N_2 \dots N_n$ where N is the total number of nodes in the system
- Every node is assumed to be identical and there is no special node with different hardware or software capabilities.
- Mobile host can access the data periodically.
- Nodes move in a random fashion at constant speed.
- Our topology is not a complete graph.
- The topology remains static for a certain period of time once it is deployed.
- The shared data or information is unchangeable. Hence no updates to data items and related consistency are considered.

4 Proposed models

4.1 Initialization phase

In this phase, the dominating nodes (servers) in the connected topology are identified and the number of dominating nodes is deprecated to find the minimum number of

Table 1 Sample centrality measures taken for a network size 10

Node_Id	SC	EC	DC	RankNo:1	No: of nodes served
1	114	-3.42	7.8	SC=6	SC: 9 nodes
2	127	-3.61	8.13		
3	72	-2.7	5.9		
4	95	-3.11	6.53	EC=.9	EC: 5 nodes
5	108	-3.32	7.02		
6	130	-3.6	7.8		
7	91	-3.18	6.78	DC = 8	DC: 6 nodes
8	111	-3.37	8.14		
9	24.6	-1.5	3.69		
10	20.5	-1.59	3.9		

Table 2 Correlation between centrality measures

	T +Δ1		T +Δ2		T +Δ3		T +Δ4		Average Correlation						
	SC	EC	SC	EC	SC	EC	SC	EC	SC	EC					
Network Size 10	EC	-0.258	EC	-0.012	EC	-0.99	EC	-0.99	EC	-0.83					
	DC	-0.62	-0.76	DC	0.98	0.06	DC	0.97	-0.68	DC	0.03	-0.07	DC	0.34	-0.36
	SC		EC		SC		EC		SC		EC		SC		EC
Network Size 20	EC	0.8		EC	-0.74		EC	-0.8		EC	-0.78		EC	-0.38	
	DC	0.98	0.54	DC	0.99	-0.54	DC	-0.05	-0.07	DC	-0.07	-0.07	DC	0.46	-0.07
	SC		EC		SC		EC		SC		EC		SC		EC
Network Size 50	EC	0.009		EC	0.46		EC	0.17		EC	0.23		EC	0.21	
	DC	0.6	0.54	DC	0.25	-0.12	DC	0.86	-0.97	DC	-0.32	-0.36	DC	0.34	-0.91
	SC		EC		SC		EC		SC		EC		SC		EC

dominating nodes using binary integer programming technique. These nodes perform as replication points where the copy of the shared data is allocated. The dominators then, advertise themselves as a new data server to their immediate neighbors by sending their identity and the data identity. On receiving this message, the neighbors save this information at a table and access the data from the server whenever needed. Besides, we split the networks into several logical regions to ease the maintenance phase. Each region is defined as a set of nodes that consist of one server and multiple clients that are immediate neighbors to the server as shown in Fig. 2

4.1.1 Static data replica allocation algorithm

1. The data servers who desire to replicate data solicit the algorithm (4.1.2) to find the minimum dominating set for the current topology.
2. Identify the dominating nodes and dispense the replica to those nodes in graph G

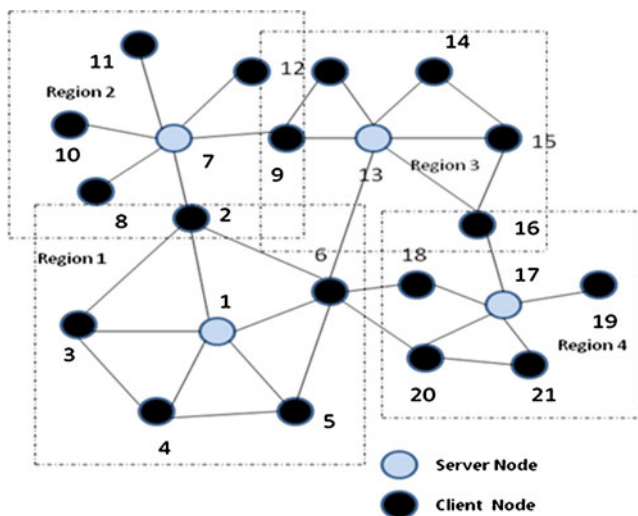


Fig. 2 Sample Topology

3. Dominating nodes then recognizes its immediate neighbors ($k=1$ hop) and broadcasts the identifier of the data it holds to its neighbors.
4. If the neighbor receives the broadcast message from one or more data servers, it stores the identifier of the data servers and the data identifiers in a table named as replica table.

4.1.2 Algorithm to find the minimum dominating set

1. Given a graph G , find the number of vertices (nodes) n in G .
2. Compute the adjacency matrix $M(G)$ for $k=1$ hop.
3. Initialize identity matrix $I(G)$, a $n \times n$ matrix.
4. Compute $A=M(G)+I(G)$ for $k=1$ hop.
5. Using binary integer programming (4.1.3), optimization technique, find a binary vector x that minimizes a linear function

$$\min \sum_{i=1}^n c_i \cdot f(x) \text{ such that } (-A) \cdot x \leq (-b)$$

Where f is a vector containing the coefficients of the linear objective function, A is the matrix containing the coefficients of the linear inequality constraints and b corresponds to the right hand side of the linear inequality constraints

6. Find the non_zero elements in the binary vector x . The indices of the non_zero elements represent the identifier of the dominating nodes.
7. Return the indices of non_zero elements.

4.1.3 Algorithm to find a binary vector x (Binary integer programming using branch and bound method)

Binary integer programming (BIP) [17] is a LP based branch and bound algorithm to solve the binary integer problems. Feasible solution is found by solving series of LP relaxation problems. The algorithm given below creates a search tree by repeatedly adding constraints to the problem that is branching.

At each node, the algorithm solves an LP relaxation problem using the constraints at that node and decides whether to branch or move to next node depending on the outcome.

1. Set $U = +\infty$ and mark all nodes in tree as active.
2. Select an active node k and solve LP relaxation

$$\min \sum_{i=1}^n c_i f(\hat{x}) \text{ such that } (-A) \cdot \hat{x} \leq (-b)$$

Let \hat{x} be the solution of relaxation.

3. If $c \cdot f(\hat{x}) \geq U$ mark all the nodes in the sub tree with root k as inactive.
4. If all the components of \hat{x} are 0 or 1 mark all nodes in the sub tree root k as inactive.

Else if $c \cdot f(\hat{x}) < U$ then set $U := c \cdot f(\hat{x})$ and save \hat{x} as the best feasible point so far.

5. Otherwise mark node k as inactive.
6. Go to Step 2. This process is repeated until feasible binary vector x is found
7. Return the binary vector x .

4.1.4 Algorithm description to find minimum dominating set (4.1.2)

This section explains working principle of the algorithm to identify minimum dominating set using a simple illustration. Let us assume a graph $G = (V, E)$ that consists of three (n) nodes (**Step 1 in 4.1.2**) and connected linearly as shown in Fig. 3. The adjacency matrix $M(G)$ of a graph is computed and added with identity matrix $I(G)$ to compute A where A is the matrix containing the coefficients of the linear inequality constraints in left hand side (LHS) and b is the $n \times 1$ identity matrix containing linear inequality constraints in right hand side (RHS) (**Steps 2 – 4 in 4.1.2**).

For this example, $M(G)$, $I(G)$, $A(G)$ are computed as $M(G)$

$$= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad A = M(G) + I$$

$$= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

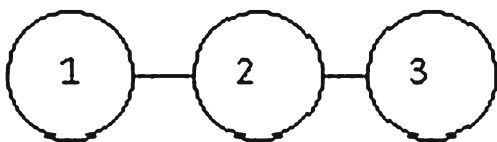


Fig. 3 The sample graph to identify minimum dominating set

The next step is to solve the equation $\min \sum_{i=1}^n c_i f(x)$ such that $(-A) \cdot x \leq (-b)$, using Binary integer programming optimization technique (**Algorithm 4.1.3**) and identify the best binary vector x and the non_zero elements of the vector x which will serve as dominating nodes (**Steps 5–7 in 4.1.2**).

Here the binary vector x is represented as $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ given that

the graph consist of 3 nodes and b is represented as $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$.

The equation $\min \sum_{i=1}^n c_i f(x)$ such that $(-A) \cdot x \leq (-b)$ can be represented as follows:

$$\text{Min } Z = x_1 + x_2 + x_3 \text{ such that } - \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Where $z = \sum_{i=1}^n c_i \cdot f(x)$

Further, this equation can be simplified into:
 $\text{min } Z = x_1 + x_2 + x_3$ such that

$$\begin{aligned} -x_1 - x_2 &\leq -1 \text{ (cond : 1);} \\ -x_1 - x_2 - x_3 &\leq -1 \text{ (cond : 2);} \\ -x_2 - x_3 &\leq -1 \text{ (cond : 3).} \end{aligned}$$

Using the above mentioned equation and the set of constraints, the binary integer programming (**4.1.3**) algorithm is invoked and the description of the algorithm is explained in subsection **4.1.4.1**

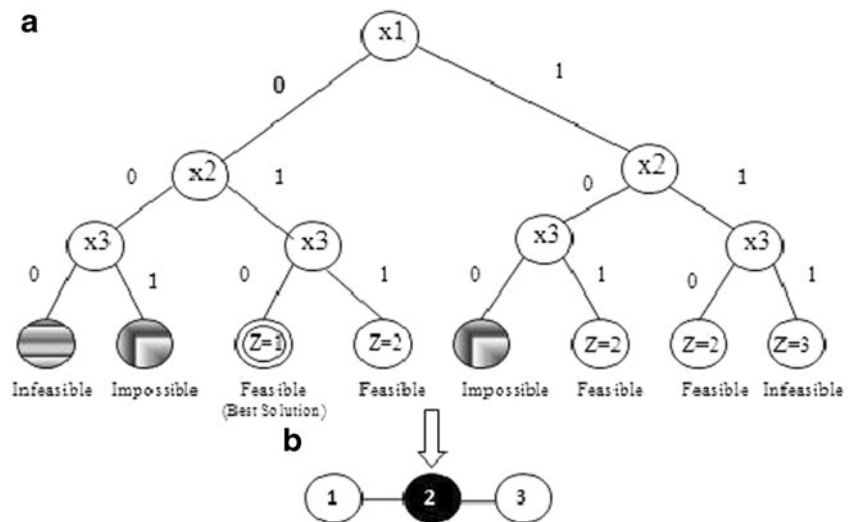
4.1.4.1 Branch and bound based BIP algorithm description (4.1.3)

To find the minimum dominating set, the LP based branch and bound technique is used. As per the algorithm, set $U = +\infty$ and mark all nodes as active (**Step 1 in 4.1.3**) and the tree shown in Fig. 4(a) and (b) will explain how the best feasible solution x (**Steps 2– 6 in 4.1.3**) is found.

LHS of a Tree: Branch x_1 and set $x_1 = 0$

1. Branch to x_2 and set $x_2 = 0$; No solution found.
 - a. Branch to x_3
 - i. Set $x_3 = 0$; where $\hat{x} = (000)$. **Infeasible** solution as per the step mentioned in the algorithm (**Step 4 part in 4.1.3**). Hence make the sub tree as **inactive**.
 - ii. Set $x_3 = 1$; where $\hat{x} = (001)$. **Impossible** solution as it does not satisfy constraints mentioned in (**cond: I**) (**Step 5 in 4.1.3**). Make the subtree as **inactive**

Fig. 4 (a) Branch and Bound Method, (b) A Graph with dominating node



2. Set $x_2 = 1$; No solution found

a. Branch to x_3

- i. Set $x_3 = 0$; where $\hat{x} = (0 1 0)$. **Feasible** solution since it satisfy all the constraints and the Z is computed as 1 which is less than U . Hence set $U := Z$ and make the subtree **active (Step 4 Else part in 4.1.3)**.
- ii. Set $x_3 = 1$; where $\hat{x} = (0 1 1)$. **Feasible** solution but $Z > U$ since Z value is 2. Hence make the subtree **inactive (Step 3 in 4.1.3)**.

RHS of a tree: Set $x_1 = 1$

1. Branch to x_2 and set $x_2 = 0$; No solution found.

a. Branch to x_3

- i. Set $x_3 = 0$; where $\hat{x} = (1 0 0)$. **Impossible** solution as it does not satisfy the constraints (**cond: 3**) (**Step 5 in 4.1.3**). Hence make the subtree **inactive**.
- ii. Set $x_3 = 1$; where $\hat{x} = (1 0 1)$. **Feasible** solution. But Z value ($Z=2$) when compared to U is greater. Hence make the subtree **inactive (Step 3 in 4.1.3)**.

2. Set $x_2 = 1$

a. Branch to x_3

- i. Set $x_3 = 0$; where $\hat{x} = (1 1 0)$. **Feasible** solution. But $Z > U$ since Z value is 2. Hence make the subtree **inactive (Step 3 in 4.1.3)**.
- ii. Set $x_3 = 1$; where $\hat{x} = (1 1 1)$. **InFeasible** solution as per the step mentioned in the algorithm (**Step 4 if part in 4.1.3**). Hence make the subtree **inactive**.

The subtree \hat{x} that is identified as active will be best possible binary vector x . The best feasible binary vector identified in this example is $x = (0 1 0)$ (**Step 7 in 4.1.3**). Hence the index of non-zero element, i.e. **2** is chosen as a

dominating node using binary integer programming technique as shown in Fig. 4(b).

4.2 Maintenance phase

In this phase, in a given region, if the mobility of the server is discerned, sub graph centrality is calculated for each node in the region to relocate the replica. In general, all centrality measures uses the adjacent matrix to compute the centrality of a node. However, in MANETs adjacency alone will not be sufficient to select the stable nodes for relocation of replica. Hence, we have introduced a normalized stability matrix as an input to compute the sub graph centrality where each element in the stability matrix determines the stability of each node with its corresponding neighbor. The node with maximum sub graph centrality will be considered as more stable and becomes a new replication point to assign the replica. The following algorithms will explain how replica is relocated on the movement of existing servers from their clients.

4.2.1 Dynamic data replica allocation algorithm

1. For every region, each node i periodically (Δt) monitors the RSS (Received Signal Strength) of the all its neighbors j where $j = 1$ to n and records them. This helps the node i to check the connectivity with its neighbors.
2. If the RSS of $node_{ij} \geq \delta_1$ then
 - 2.1. Set $Count_{ij} = Count_{ij} + 1$ /* Record the connectivity for the current and past time intervals */
 - 2.2. Set $Cm_{ij} = 1$ /* Record the connectivity only for the current time interval */
3. Else Set $Cm_{ij} = 0$, where δ_1 is the receiver sensitivity threshold
4. For every T seconds, Invoke the Mobility Prediction Algorithm (4.2.2) to predict the mobility of a client or the server.

- 4.1. If node i is a moving client and if $flag = 1$ it predicts that it will move out of range and do the following:
 - 4.1.1. Search for a new server within "k" hops and associate with it for data access.
 - 4.1.2. Otherwise fetch the most frequently or needed data from the old server and cache it.
- 4.2. Else If node i is a moving server and if $flag = 1$, check the $avgrss_T$ for all its neighbors j
 - 4.2.1.1. For $j = 1 : n$
 - 4.2.1.2. If $avgrss_T \leq \delta mov = mov + 1$;
 - 4.2.1.3. If $mov \geq \frac{n}{2}$, Invoke the relocation algorithm (4.2.3)
5. Nodes with maximum sub graph Centrality are selected as replication points to relocate the replica.

4.2.2 Mobility prediction algorithm

1. Compute the average of the RSS signal strength at every time window T where $T = \Delta t_1, \Delta t_2 \dots \Delta t_n$
2. Record the current and previous average RSS measured at T and $T - 1$.
3. If $avgrss_T \geq avgrss_{T-1} + \delta$, nodes are moving towards each other, δ value varies from 1 to 10.
 - 3.1. Set $flag = 0$;
4. If $|avgrss_{T-1} - \delta < avgrss_T < avgrss_{T-1} + \delta|$, nodes are moving around in a small area or they are stationary.
 - 4.1. Set $flag = 0$;
5. If $avgrss_T < avgrss_{T-1} - \delta$, nodes are moving away from each other and if the average $avgrss_T < \delta_I$ then
 - 5.1. Set $flag = 1$; where δ value varies from -1 to -10
6. Return the flag status

4.2.3 Replica relocation algorithm with sub graph centrality principle

1. Server node i broadcast its message to all its clients asking them to send the count value $Count_{ij}$ and the recent connectivity value cm_{ij} recorded for all their neighbors.
2. On receiving these values, server compute the weight stability matrix, an $n \times n$ matrix by calculating wm_{ij} for each node pair (i,j) where i and j varies from 1 to n in its region.

$$WSM = \begin{pmatrix} wm_{1,1} & \cdots & wm_{1,j} \\ \vdots & \ddots & \vdots \\ wm_{i,1} & \cdots & wm_{i,j} \end{pmatrix} \quad (1)$$

Where $wm_{ij} = \frac{Count_{ij}}{Maxcount}$, $Maxcount = \frac{T_c}{\Delta t}$, T_c is current time and Δt is the Time period.

3. Server then compute the connectivity matrix CM_{ij} , $n \times n$ matrix at current time interval T using cm_{ij} for all node pairs (i,j) where i and j varies from 1 to n in its region.

$$CM = \begin{pmatrix} cm_{1,1} & \cdots & cm_{1,j} \\ \vdots & \ddots & \vdots \\ cm_{i,1} & \cdots & cm_{i,j} \end{pmatrix} \quad (2)$$

The value of $cm_{ij} = 1$ if i and j are connected else $cm_{ij} = 0$

4. Using the WSM and CM , server compute the normalized weight stability matrix

$$NWSM = \alpha WSM + (1 - \alpha)CM \quad (3)$$

Where α is the normalization factor equal to 0.5.

5. Invokes the sub graph centrality Algorithm (4.2.4) by giving the $NWSM$ as an input matrix.
6. On receiving the Cs vector,
 - 6.1. Find the element with maximum sub graph centrality
 - 6.2. Remove the elements from Cs that are immediate neighbors to the element with maximum sub graph centrality
7. Repeat step 6, until Cs contains only the elements which can serve as a new replication points. The nodes with a maximum sub graph Centrality will serve as a new replication points.

4.2.4 Algorithm to find a sub graph centrality of a node

1. Compute the Eigen vector V and eigen value λ

$$[V, \lambda] = eig(NWSM_{ij}) \quad (4)$$

2. Compute matrix of squares of Eigen vector elements

$$V_2 = V^2 \quad (5)$$

3. Compute sub graph Centrality

$$Cs = V_2 \times \exp(\lambda) \quad (6)$$

Where Cs is the vector containing the subgraph centrality values for each node in a particular region.

4. Return the Cs Vector.

4.2.5 Algorithm description for maintenance phase

The work flow of Dynamic data replication algorithm (4.2.1) is represented as a flow chart and shown in Fig. 5.

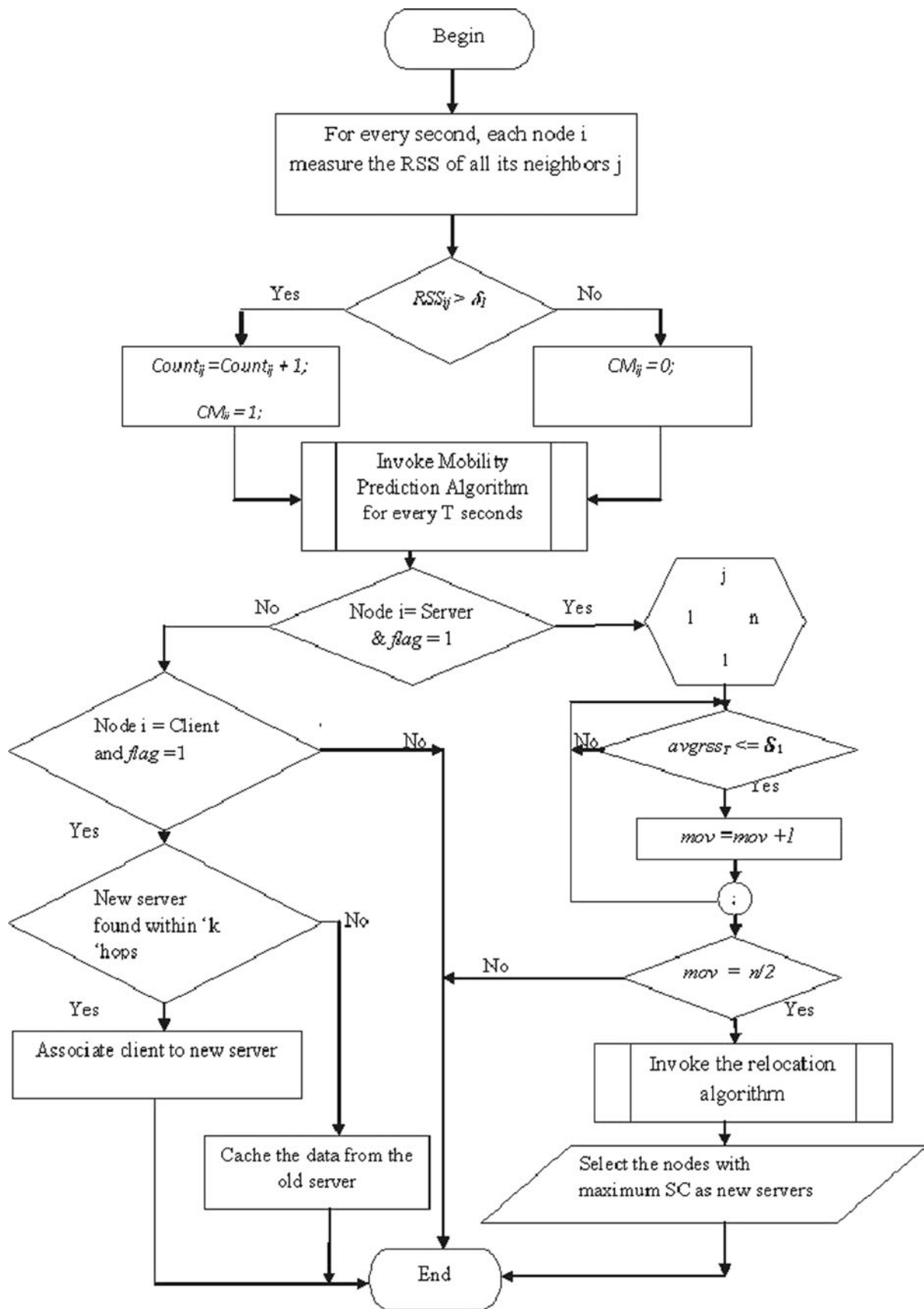


Fig. 5 Flowchart to execute maintenance phase

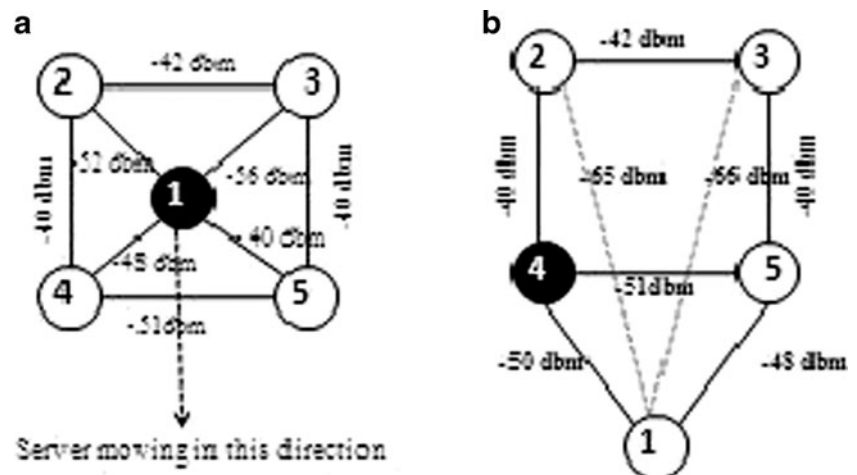
Table 3 Notations used in the maintenance phase algorithms

Notation	Description
$Count_{ij}$	Incremented by 1 for every time units when two nodes are in range
CM_{ij}	Set to 1 if two nodes are in range at current time T_c ; Otherwise set to 0
$Flag$	Set to 1 if a node move away from its neighbor; Otherwise set to zero
mov	Incremented by one for every neighbor if a node moves away from them
T	Time window
Δt	Time units in time window
T_c	Current Time
WSM	Weight stability matrix for past connectivity and current connectivity
δI	Receiver sensitivity threshold
CM	Current Connectivity matrix
$NWSM$	Normalized Stability matrix
C_s	Subgraph Centrality Vector
α	Normalized factor

The notations used in the maintenance phase algorithms are also described in the Table 3. Finally, the algorithm steps are explained with illustration.

The Fig. 6 shows the present state of the topology consisting of five nodes at time window $T-1$ and T seconds. The value mentioned in each edge represent the $avgrss$ recorded at time window $T-1$ and T . The size of the time window T is assumed to be 10 seconds. For every second, the RSS of each connected node pair i and j is measured and recorded (**Step1 in 4.2.1**). If RSS is within the threshold δ_1 , two nodes are within the range and hence the count $Count_{ij}$ is incremented by one and the current connectivity CM_{ij} is recorded as one (**Step2 in 4.2.1**). Otherwise if RSS is not within the threshold δ_1 , current connectivity CM_{ij} is set to zero (**Step3 in 4.2.1**). From the Fig. 5 it is obvious that the node1 is a dominator and hence it is identified as a server by using the minimum dominating set algorithm (4.1) and the other nodes are designated as clients. **The steps 4 and 5 of 4.2.1** are executed based on the outcomes of algorithm 4.2.2 and 4.2.3

For every T seconds, invoke the mobility prediction algorithm (4.2.2) and compute the $avgrss$ for each node pair (**Step 1 in 4.2.2**). Average received signal strength for current window T is recorded as $avgrss_T$ and for previous window $T-1$ as $avgrss_{T-1}$ (**Steps 2–3 in 4.2.2**). Then the $avgrss$ of each node pair i and j for current time window T and past time window $T-1$ is compared (**Step 4–6 in 4.2.2**). For example, let us assume that the server starts to move in a particular direction as shown in Fig. 6(a) and the current time T_c is 50 s. For every 10 s, the mobility prediction algorithm is invoked, and the $avgrss$ calculated for current time window T (40–50 s) will be compared with $avgrss$ for past time window $T-1$ (30–40 s) for every node pair i and j . From the Fig.6(b), it is observed that the $avgrss_T$ computed for the server with its neighbors 2 and 3 is less than the $avgrss_{T-1} - \delta$ i.e. $-65 < -62(-52 -10)$ and $-66 < -54 (-56-10)$ where $\delta =10$ and less than the receiver sensitivity threshold δ_1 and hence set the flag to one for node pair 1–2 and 1–3 (**Step 6 in 4.2.2**). However the $avgrss_T$ computed for the server with its neighbors 4 and

Fig. 6 Topology state at time window $T-1$ (a) and Time window T (b)

5 is less than the $avgrss_{T-1} - \delta$ but greater than the receiver sensitivity threshold δ_1 . Hence the flag is set to zero. For other node pairs the flag remain zero (Steps 4–5 in 4.2.2) as the nodes remain stationary. The flag status is returned to algorithm 4.2.1.

Henceforth the Step 4 in algorithm 4.2.1 is executed. As the moving node is a server (Step 4.2 in 4.2.1) the *mov* variable is incremented to 2 (Step 4.2.1.1 in 4.2.1) since it has moved away from two clients. The server calls up the replica relocation algorithm (4.2.3) since *mov* is equal to 2 (Step 4.2.1.2 in 4.2.1). As clients are stationary Step 4.1 will not be executed.

To illustrate the work flow of replica relocation algorithm (4.2.3), assume the current time T_c as 50 s, clients are stationary and the server is mobile. The received signal strength measured for every second will make the *Count* value of each node pair as 50 if they remain connected. Otherwise the count value may fluctuate between 40 and 50 if the nodes try to move away from their neighbors at time $T_c = 40$ s. The server collects the count value and current connectivity of each node pair *i* and *j* (Step 1 in 4.2.3) and compute the matrices as follows:

$$Count_{ij} = \begin{pmatrix} 0 & 43 & 42 & 50 & 50 \\ 43 & 0 & 50 & 50 & 0 \\ 42 & 50 & 0 & 0 & 50 \\ 50 & 50 & 0 & 0 & 50 \\ 50 & 0 & 50 & 50 & 0 \end{pmatrix} \text{ and}$$

$$CM_{ij} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

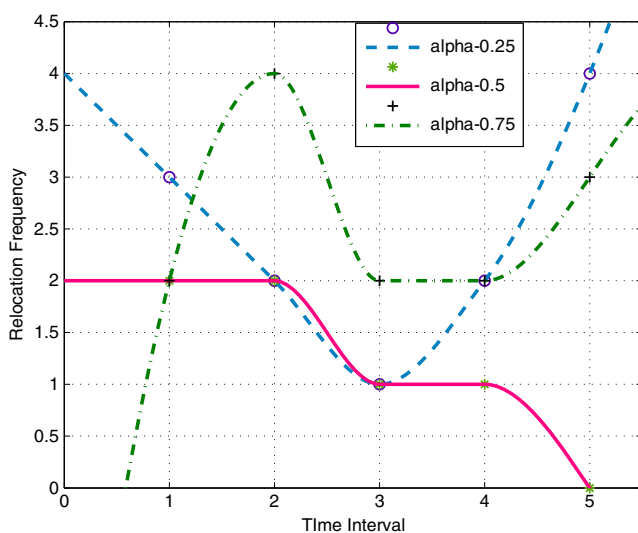


Fig. 7 Fine Tuning of α

As per the example, the node pairs 1–2 and 2–3 will be in the range of 40–50 because the server has moved from them. Using the past connectivity matrix $Count_{ij}$, the weight stability matrix *WSM* (Step 2 in 4.2.3) is

calculated as: $WSM = \begin{pmatrix} 0 & 0.86 & 0.84 & 1 & 1 \\ 0.86 & 0 & 1 & 1 & 0 \\ 0.85 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$. The illus-

tration in the next line shows how the elements of *WSM* are computed. If $Count_{12} = 43$ for node pair 1–2, current time $T_c = 50$ s and $\Delta t_1 = 1$ s, the $max\ count = \frac{50}{1} = 50$. Therefore $wm_{12} = \frac{43}{50} = 0.86$. Then the server computes *NWSM* (Step 4 in 4.2.3) by multiplying α to *WSM* and $\alpha-1$ to *CM* (Step 3 in 4.2.3). Here α is the normalization factor equal to 0.5 and *WSM* and *CM* are as follows:

$$\alpha \times WSM = \begin{pmatrix} 0 & 0.43 & 0.42 & 0.5 & 0.5 \\ 0.43 & 0 & 0.5 & 0.5 & 0 \\ 0.42 & 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0.5 & 0 \end{pmatrix} \text{ and}$$

$$(\alpha - 1) \times CM = \begin{pmatrix} 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0.5 & 0 \end{pmatrix}$$

Table 4 Simulation parameters

Simulation parameters	
Node density	50
Cluster Size	10, 20, 30
Link layer Implementation	IEEE 802.11b
Simulation Area Size	200 × 200
Topology	Random Graph
Mobility Model	Random Way Point
Transmit Power	0.2 watts
RSS Threshold	-72 dbm
Mobility prediction Threshold $\delta 1$	-62dbm
Transmission Range	10 – 30 m
Max Speed	5,10,15 m/sec
Max pause	1 sec
Date rate	11 Mbps
Antenna Type	Omni directional
Propagation Model	Shadowing
Routing protocol	AODV
Packet Loss	None
Simulation Time	500 sec
α	0.5



Fig. 8 Mobility of a server with respect to its clients

$$NWSM = \alpha \times WSM + (\alpha - 1) \times CM$$

$$= \begin{pmatrix} 0 & 0.43 & 0.42 & 1 & 1 \\ 0.43 & 0 & 1 & 1 & 0 \\ 0.42 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

The matrix *NWSM* is normalized as the input matrix to compute the subgraph centrality should be in the range (0, 1). Henceforth *NWSM* will serve as the input to calculate the subgraph centrality of a node (**Step 5 in 4.2.3**)

To compute the sub graph centrality, the algorithm mentioned in 4.2.4 called upon, the eigenvector *V* and Eigen values λ of *NWSM* (**Step 1 in 4.2.4**) are calculated and given as follows:

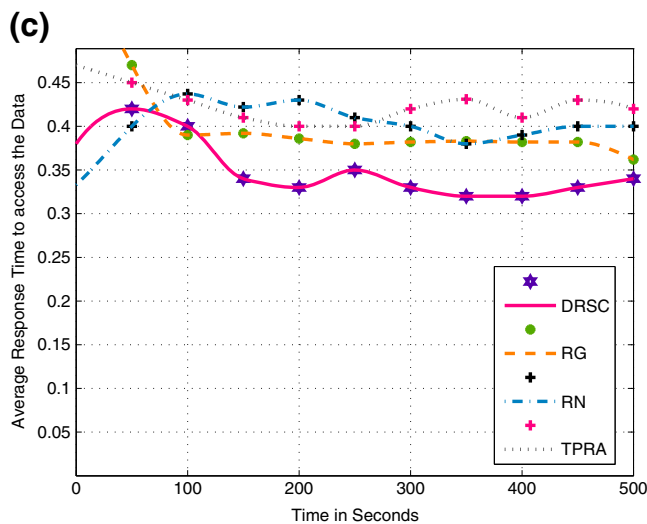
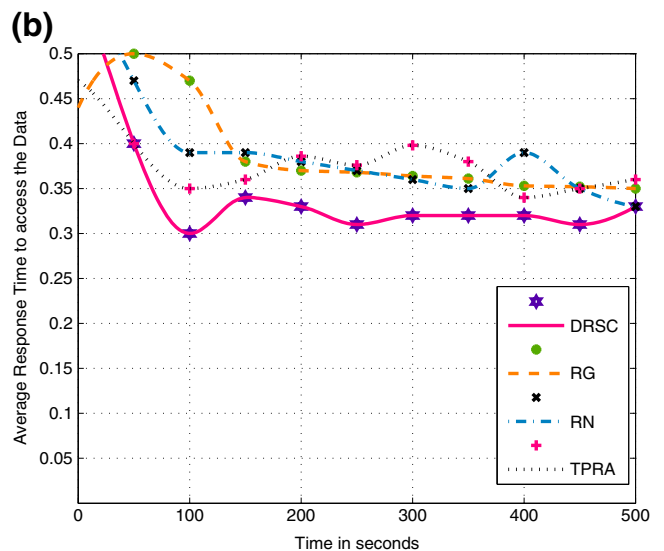
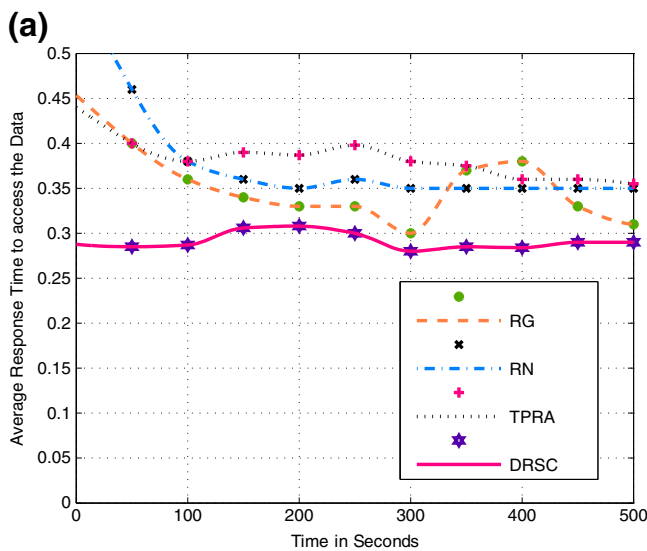


Fig. 9 Average response time to access the data from the server moving at a speed of (a) 5 m/sec, (b) 10 m/sec, (c) 15 m/sec

$$V = \begin{pmatrix} 0.00 & -0.8 & 0. & 0.36 & -0.47 \\ -0.5 & -0.03 & -0.5 & 0.5 & -0.38 \\ 0.49 & -0.03 & 0.49 & 0.59 & -0.38 \\ 0.49 & 0.42 & -0.49 & -0.29 & -0.48 \\ -0.5 & 0.41 & 0.5 & -0.29 & -0.48 \end{pmatrix} \text{ and } \lambda = \begin{pmatrix} -2 \\ -1 \\ 0 \\ 0.2 \\ 2.7 \end{pmatrix}$$

The next step (**Step 2 in 4.2.4**) is to compute V^2 and exponential $\exp(\lambda)$ which are calculated and given as:

$$V^2 = V_2 = \begin{pmatrix} 0 & 0.6 & 0 & 0.13 & 0.2 \\ 0.25 & 0.0 & 0.25 & 0.34 & 0.15 \\ 0.24 & 0.0 & 0.24 & 0.34 & 0.15 \\ 0.24 & 0.17 & 0.24 & 0.08 & 0.23 \\ 0.25 & 0.17 & 0.25 & 0.08 & 0.23 \end{pmatrix} \text{ and } \exp(\lambda) = \begin{pmatrix} 0.13 \\ 0.36 \\ 1 \\ 1.2 \\ 15.9 \end{pmatrix}$$

Finally, the subgraph centrality of node is calculated using the formula, $Cs = V_2 \times \exp(\lambda)$ (**Step 3 in 4.2.4**)

and represented as $Cs = \begin{pmatrix} 3.9 \\ 3.1 \\ 3.13 \\ 4.2 \\ 4.2 \end{pmatrix}$. The vector Cs is returned

to algorithm **4.2.3 (Step 4 in 4.2.4)**. On receiving the Cs the element with the maximum subgraph centrality is determined and the neighbors in one or two hops are removed from the vector Cs i.e. the entry is made as zero. In the given example, node 4 or 5 will become a new server since node 4 and 5 has maximum subgraph centrality and other nodes can reach them in one or two hops. Therefore node 4 or 5 will become a new server and the indices are returned to algorithm 4.2.1 (**Step 5 in 4.2.1**).

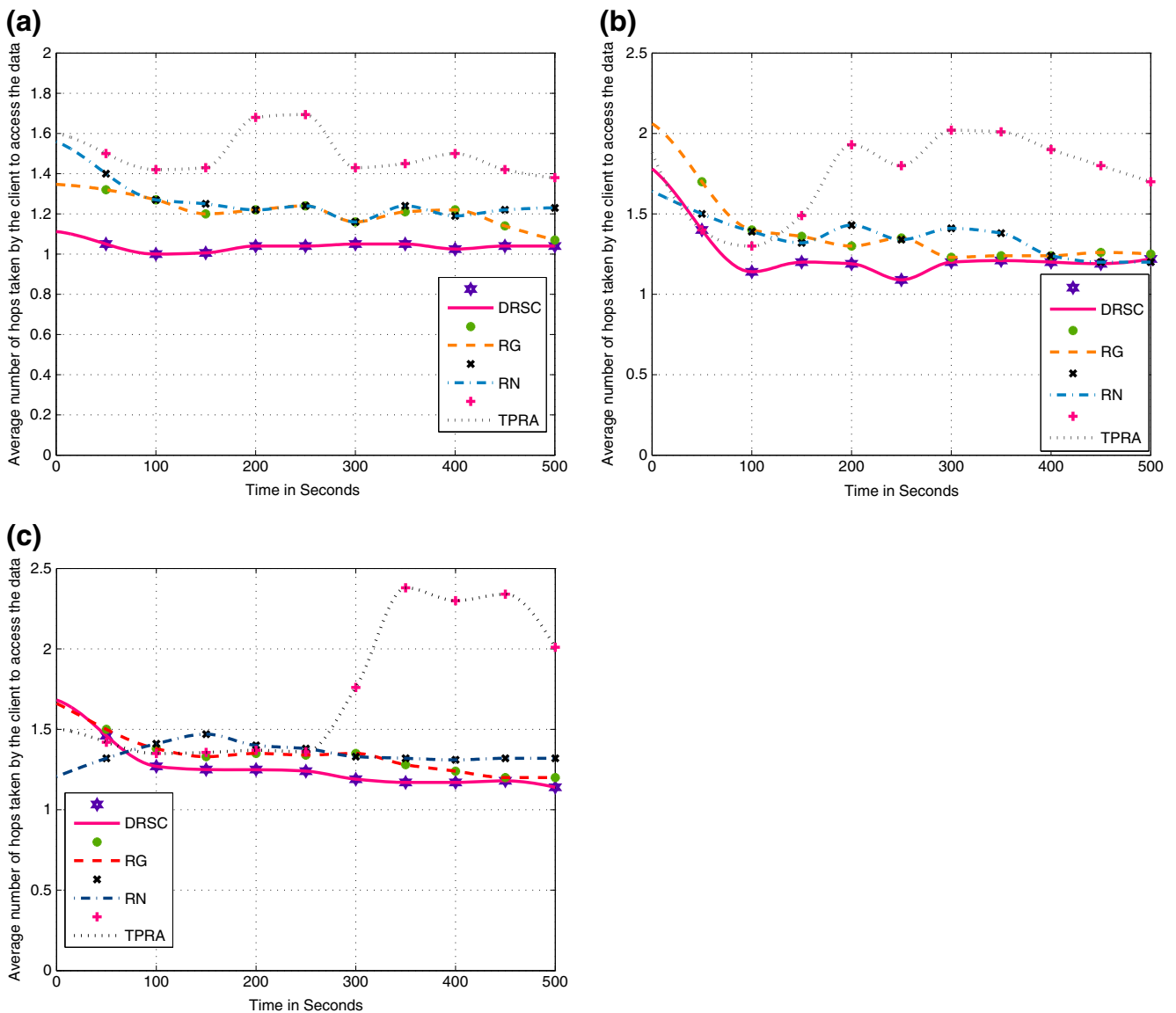


Fig. 10 Average number of hops to access the data from the server moving at a speed of (a) 5 m/sec, (b) 10 m/sec, (c) 15 m/sec

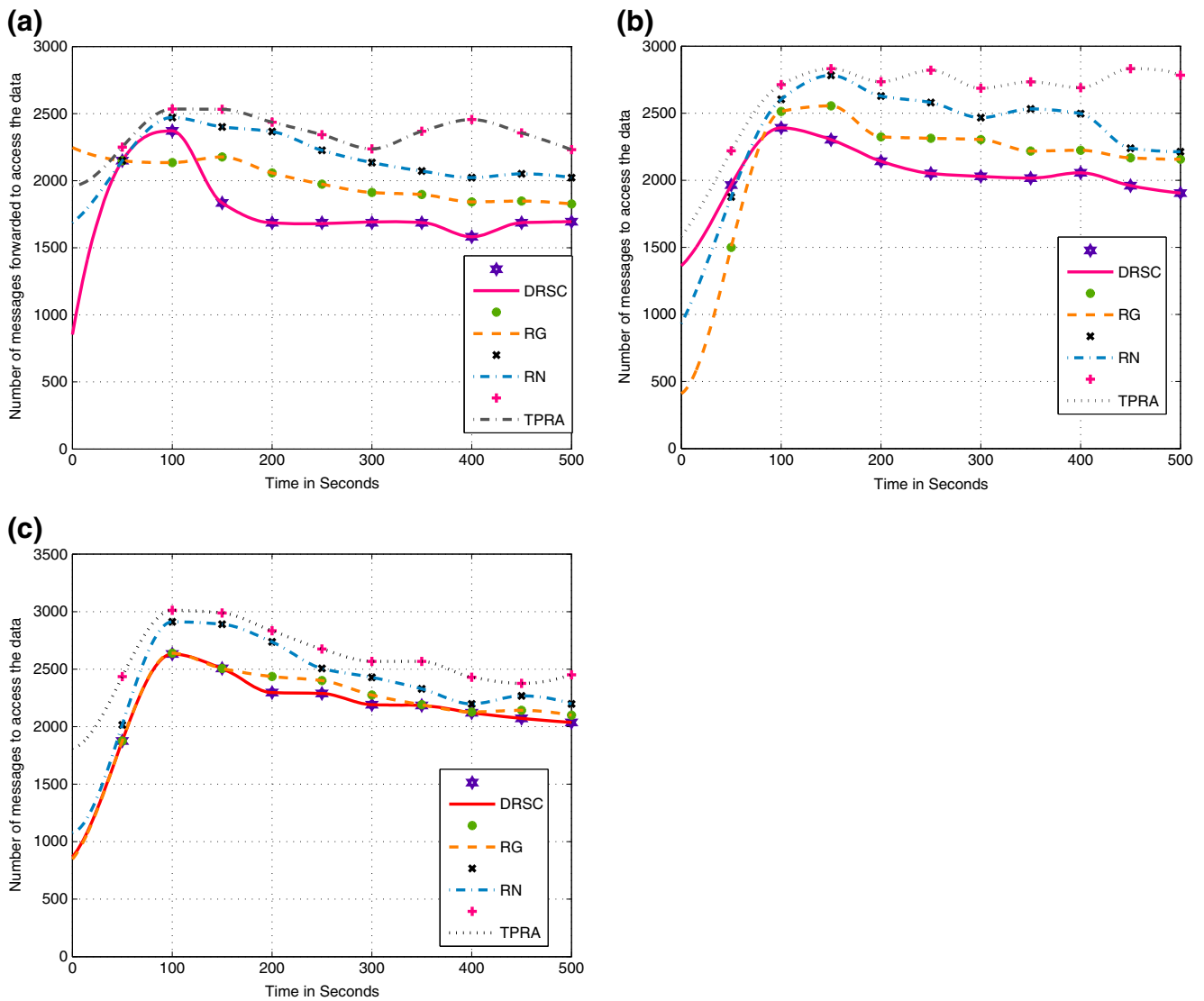


Fig. 11 Communication cost to access the data at a maximum speed of (a) 5 m/sec, (b) 10 m/sec, and (c) 15 m/sec

4.2.6 Fine tuning of α

To select the optimal value of α , the experiments were conducted with different network connectivity patterns and varying the size of the network. From Fig. 7 it was observed that tuning $\alpha=0.5$ was best possible because the relocation frequency (number of times the data relocated at a particular time interval) was consistent at different time intervals with average relocation frequency of 1.5 for every 100 s i.e. the network remain stable at an average of 80 s. Tuning $\alpha=0.25$, i.e., giving more importance to recent connectivity lead to an average relocation frequency of 2.4 for every 100 s i.e. the network remain stable on an average of 40 s. Assuming $\alpha=0.75$, i.e., giving more importance to past connectivity also lead to fluctuations in relocation of data with the average frequency of 2.6 for every 100 s and the network remain stable on an average of 43 s.

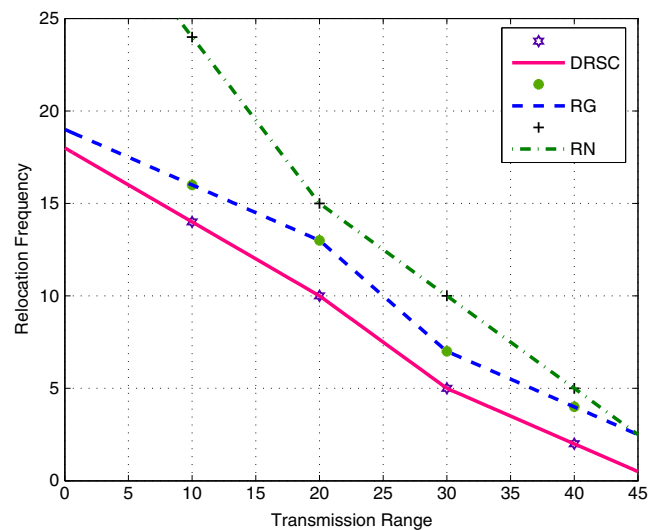


Fig. 12 Relocation Frequency varying the transmission ranges

4.3 Time complexity of the proposed algorithms

We have proposed two graph theory concepts to locate a node for holding the replica. During the initialization phase, the replication points are identified using minimum dominating set. Since Binary Integer programming is used as an optimization technique, branch and bound strategy is used to find minimum number of dominating nodes. Using branch and bound technique, the time complexity grows exponentially when the search becomes exhaustive and if the number of variables x_i increases in the optimization function. To limit the search, the algorithm limits the number of iterations, execution time and number of nodes. Hence the time complexity of calculating minimum dominating set can be represented as $O(2^n)$ where n is the number of decision variables x_i where $i = 1$ to n . Due to its complexity, this algorithm is used only in the initial stage i.e. when the network is stationary. On the other hand, using sub graph centrality measure substantially reduces the time complexity to $O(n^3)$ to identify the replication nodes. Hence this measure will serve as improved measure to identify the replication points in a dynamic environment.

5 Simulation results and analysis

We have simulated our model using wireless network simulator-MATLAB [18–21]. The simulation parameters are listed in Table 4. Our approach is compared with TPRA, RN and RG schemes [14] [10]. The following are the performance metrics [6] used to evaluate our model with existing models:

- *Response time*: The time taken for the client to place the data request and get the data as a reply.
- *Query cost*: The average number of hops taken by the client to access the data.
- *Data Communication cost of the Forwarder*: Number of messages forwarded by the forwarders to access the data.
- *Relocation Frequency*: This metric is defined as number of times the data relocated from one node to another
- *Data Accessibility*: The probability of accessing the data in one hop. The probability is 1, if the client can access the data in one hop. 0 Otherwise.
- *Replication degree*: The average number of replicas in a cluster.

To exhibit the dynamic behavior of the network, as a sample, the server1 mobility (speed 5 m/sec) with respect to its clients is shown in Fig. 8. We observed that the network remains static for

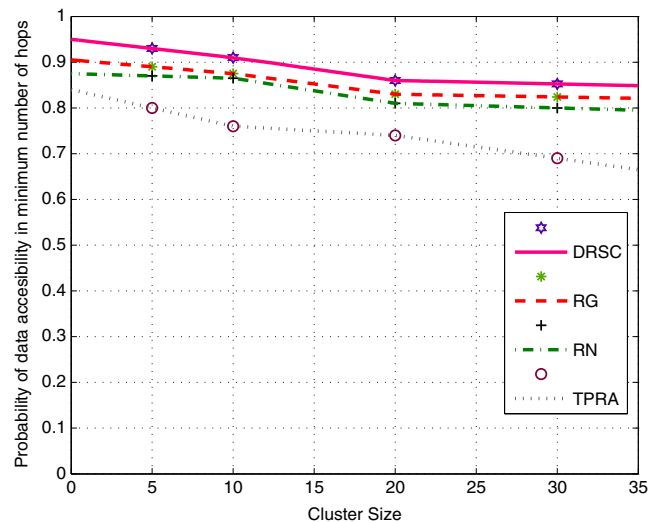


Fig. 13 Data accessibility with respect to its cluster size

a period of 10 s. After 10 s, the server starts moving in a random fashion at a constant speed. At time 50, it is observed that, the server is moving away from client 2, 5, 6 and moving closer to client 3 and 4 but still all are in communication range. However, at time 110 the server is moving away from all its clients and may lose its communication in future. At this point, sub graph centrality of each node is calculated to relocate the replica.

Figure 9 shows the response time for a client to access the data at different moving speed of the server for all four methods. The response time of our approach (DRSC) is minimal at all speeds when compared to the existing methods because if most of the clients access the data at most 2 hops, data relocation is invoked immediately. It is also observed that in reliable

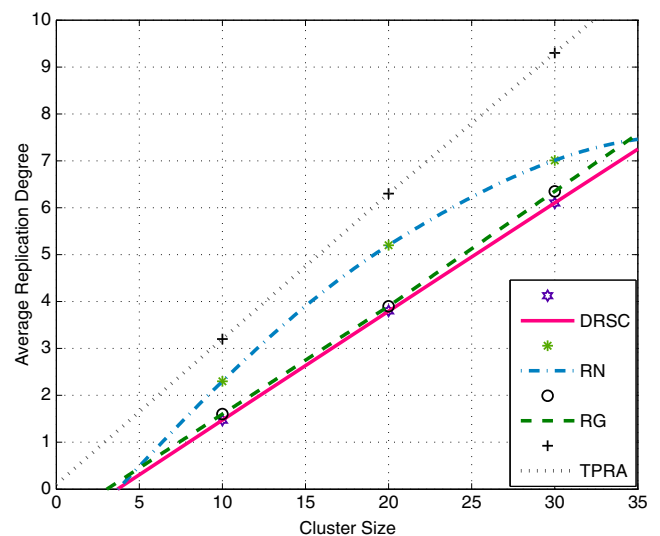


Fig. 14 Average replication degree

group (RG) method response time is less when compared to reliable neighbor (RN) scheme. This due the fact that the RG method always selects the best node that is nearer to many clients. But in RN method, server is selected based on the number of immediate neighbors it can serve. As the server starts to move from immediate neighbors, the response time will increase. In TPRA method, when the server starts to move, the clients access the data from it in multihops or from nearby servers and thus increasing the response time. Further observation we made in all three graphs are, there is a slight increase in response time for different moving speeds. The reason behind this is as server move faster, distance between the client and server increases thus increasing the response time

Figure 10 shows the average number of hops traversed by a client to access the data at different moving speeds of the servers. In our approach minimum number of hops i.e. $k = 1$ to 1.5 hop is maintained. Little increase in the number of hops is due to the server mobility from few of its clients. In addition, RG and RN schemes also maintain minimum number of hops as both the methods replicate data near to clients. However, in two phase replication approach, maximum number of hops ranges from 2 to 2.5 because the mechanism replicates data in such a way that the distance between the two servers is equal to three hops. If the client cannot reach the existing server it searches for the server that is more than two hops away.

Figure 11 represents the communication cost for different moving speed of the servers. This cost is defined as the number of messages forwarded by the source and intermediate nodes to access the data. The message includes data request and data reply messages from and to the clients. Figure 11 clearly depicts that the number of forwarded messages in our approach are less because the client can access the data at one hop for most of the time. However, in other three approaches, communication cost will be moderately high because due to dynamic behavior, the client traverses multiple hops to access the data and thus increasing the forwarded messages. Furthermore fast movement of servers also rises the communication cost.

Figure 12 depicts the relocation frequency for different transmission range. This graph explains how many times data is relocated by varying the transmission range. It was observed that for a simulation run of 500 s and for moving speed of 15 m/sec, the relocation frequency decreases as the transmission range between two nodes increases. If the transmission range between two nodes is limited to 10 m, the network formed will be a sparse network and number of relocations will be high. On the other hand, if the transmission range is extended to 40 m relocation number declines. However, 30 m will be the optimum range as this the maximum range supported by IEEE 802.11.b in an indoor environment. From the graph, it is clear that the number of

relocations in our method is better than RG and RN, since we consider the most stable node to relocate the data.

Figure 13 shows the probability of data accessibility in one hop versus the cluster size. In our approach if the cluster size is small, the probability of accessing data in one hop is in the range 0.9 to 1. As cluster size increases, the probability goes below since all nodes cannot be connected in one hop. However our method when compared to other methods has higher probability in accessing the data in one hop.

Figure 14 represents average replication degree for different cluster size measured at different time instants. From the graph it is observed that the replication degree is less in our approach compare to RN, RG and TPRA. As the cluster size increases the number of replica increases in all the above method. Replication degree of reliable group is close to our method as it picks up the best node than are in the range of many clients.

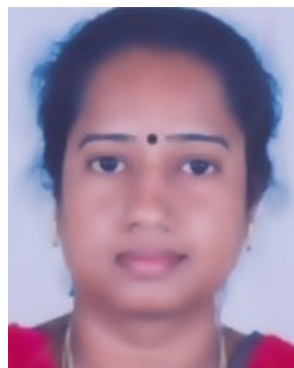
6 Conclusions

Data Replication is a key technique in many of the applications of ad-hoc networks that exhibits collaborative behavior. We have proposed a replication method that minimizes the replication points; decreases the data access delay, improves the response time and reduces the communication cost to access the data. Our mechanism is well accepted for the dynamic behavior of networks by emigrating data to a node that is highly stable and associated with the maximum number of nodes by using a different approach termed as a sub graph centrality principle. Our simulation results when compared to an existing methods shows that the response time is less, data access by a client to a server is at most one to two hops, minimum replication degree and the number of forwarding messages to access the data are minimal. As a consequence, our mechanism is energy-efficient and well applicable to both for sparse and dense networks and even in a complex network.

References

1. Siva Ram Murthy C, Manoj BS (2007) “Ad hoc Wireless networks, Architecture and Protocols”, Second edition Pearson Edition
2. Marcel C Castro et al. (2010) “Peer to Peer Overlay in mobile ad hoc networks”, Handbook of Peer-to-Peer Networking,, Volume . ISBN 978-0-387-09750-3. Springer Science + Media, LLC, p. 1045
3. Charli Y, Saumitra M.Das, Himabindu Purcha (2002) “Exploiting the synergy between peer to peer and mobile ad hoc networks”, ECE Technical Reports. Paper 167

4. De Moralis C, Agarwal DP (2002) “Mobile ad hoc networking”, Course material, OBR Research Center for Distributed and Mobile Computing.
5. Derhab A, Badache N (2009) “Data replication protocols for mobile ad hoc networks, a survey and taxonomy”. *IEEE Commun Surv Tutor* 2(2):33–51, Addison Wesley, Massachusetts
6. Padmanaban, Gruenwald le (2008) “A survey of data replication technique for mobile ad hoc networks databases”, *Springer link. VLDB J* 17:1143–1164
7. Padmanabhan P, Gruenwald le (2006) “Managing data replication in mobile ad hoc networks”, In *Proceedings of IEEE Xplore, International conference on Collaborative computing: network, Applications and Work sharing*
8. Moussaoui S, Gueuomi M, Badache N (2006) “Data Replication in Mobile Ad hoc Networks”, *Second International Conference on Mobile ad hoc and sensor networks*. *Proc Lect Notes Comput Sci* 4325:685–697, Springer link-verlag
9. Xiao C (2007) “Data replication approaches for ad hoc networks satisfying the time constraints”. *Int J Parallel Emergent Distrib Syst* 22(3):149–161
10. Moussaoui S, Badache N, Gueuomi M (2005) “Two phase replication approach for MANETs”. *Int J Adhoc Ubiquit Comput* 4 (5):292–303
11. Atsan E, and Ozkasap O (2007) “Applicability of eigen vector centrality principle to data replication in MANET”, In *Proc. IEEE Xplore, International symposium on Computer and Information Sciences*
12. Hara T, Loh YH, Nishio S (2003) “Data replication methods based on the stability of radio links in ad hoc networks”, In *Proc. IEEE Xplore, International workshop on database and Expert Systems*
13. Zhang Y, Yin LZ, Zhao J, Cao G (2012) “Balancing the trade-off between query delay and data availability in MANETs”. *IEEE Trans Parallel Distrib Syst* 23(4):643–650, Online:18 August 2011
14. Choi JH, Shim KS, Lee S, Wu KL (2012) “Handling selfishness in Replica allocation over a mobile ad hoc networks”. *IEEE Trans Mob Comput* 11(2):278–291
15. Narsingh Deo (1974) “Graph Theory with Applications to Engineering and Computer Science”, Prentice Hall
16. Estrada E, Rodriguez JA (2005) “Subgraph Centrality in Complex Networks”, *Phys Rev E Am Phys Soc J Vol.no71: Issue 5*
17. Morgan MJ, Grout Vic (2008) “Finding Optimal Solutions to Backbone Minimization Problems Using Mixed Integer Programming”, *Seventh International Conference (INC 2008)*, Plymouth, UK
18. *Wireless Network Simulator in Matlab*, <http://www.Wireless-matlab.sourceforge.net>.
19. Ed. Scheineerman, *Matgraph*, “A graph Theory Toolbox for MATLAB” <http://www.ams.jhu.edu>
20. MATLAB, “The Language of Technical Computing”, <http://www.mathworks.com>
21. Estrada’s sub graph centrality, communicability, community detection <http://intersci.ss.uci.edu/wiki/index.php/>



M Pushpalatha received her M.E degree in Madras University, Tamil Nadu, India in the Year 2001. She is currently working towards her Ph.D in Computer Science and Engineering. Her research interest includes. Wireless ad hoc networks, Distributed systems and Sensor Networks.



T. Ramarao currently, working as ‘Professor & Head’, Department of Telecommunication Engineering, Faculty of Engineering & Technology, SRM University, India. He received his PhD degree on ‘Radio Wave Propagation studies for Fixed and Mobile Communications over Southern India’ from Sri Venkateswara University, Tirupati, India in the year 2000. He has a long-standing research history on Radiowave Propagation Studies for Wireless Communications. His research interests are, Radio Channel Measurements & Modeling, Broadband Wireless Communications/Networks and Wireless Information Networks. He authored papers in reputed journals/transactions and international/national conferences.



Revathi Venkataraman is pursuing her PhD. Currently working as Assistant Professor in the Department of Computer Science and Engineering, SRM University. Her research interests are wireless ad hoc and sensor network testbed developments which are ongoing research activities funded by Indian Government. Her other research interests include wireless networks and security, trust computing and routing in ad hoc networks.