ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Improving Data Accessibility by Using Selfish Nodes in Mobile Ad-hoc Networks

[I]A.U AabithaNasreen, [II]P.Gnanasekaran
[I,II]B.S Abdur Rahman University, Chennai, India

Abstract
MANET has a collection of autonomous mobile nodes that can move freely and unpredictably to form a temporary network, without the need of any infrastructure. In a mobile ad hoc network, the mobile nodes will have the characteristics of mobility and constraints in resources. Since, the mobility is high, the nodes may move fast and randomly, which lead to network partitioning. The resource constraints lead to a big problem as decrease in performance and the network partitioning leads to poor data accessibility. To improve the data accessibility many data replication techniques have been proposed to minimize performance degradation caused by network partitioning in a mobile ad hoc network. Most of them assume that all mobile nodes collaborate fully in terms of sharing their memory space. However, in reality, some nodes may selfishly decide to only cooperate partially, or not at all, with other nodes. Recently, a new approach to selfish replica allocation has been proposed to handle node selfishness. The existing selfish replica allocation strategy suffers from long query delay and poor data accessibility, because it utilizes only non-selfish nodes that may be faraway nodes. The proposed method is a novel replica allocation strategy in the presence of selfish nodes, that takes into account both selfish behaviour and node distance. Moreover, through a novel node levelling technique is to utilize the memory space of all connected nodes, including selfish nodes. The conducted simulations demonstrate that the proposed strategy outperforms existing replica allocation techniques in terms of data accessibility, Query delay.

Keywords
Mobile Ad-hoc Networks, Integrated degree of selfishness, Node distance, Credit Risk, Replica Allocation

I. Introduction

A mobile ad-hoc network (MANET) is attracting considerable attention with the advance in wireless technologies and mobile devices [11]. A MANET has a collection of autonomous mobile nodes that can move freely and unpredictably to form a temporary network, without the need of any infrastructure, such as a base station. Each node in a MANET acts as a router and communicates with each other. Since a MANET requires minimal configuration and is quickly deployed, it has been applied in a variety of applications [3]. For example, a MANET can be used in special situations where the installation of infrastructure may be difficult, or even infeasible, such as natural disasters or military conflicts. A mobile peer-to-peer (P2P) file sharing system is yet another interesting application[6,13].

Network partitions can occur frequently, since nodes move freely in a MANET, causing some data to be often inaccessible to some of the nodes. Hence, data accessibility is often an important performance metric in a MANET. Data are usually replicated at nodes, other than the original owners, to increase data accessibility to cope with frequent network partitions. A considerable amount of research has recently been proposed for replica allocation in a MANET. In general, replication can simultaneously improve data accessibility and reduce query delay, i.e., query response time, if the mobile nodes in a MANET together have sufficient memory space to hold both all the replicas and the original data. For example, the response time of a query can be substantially reduced, if the query accesses a data item that has a locally stored replica.

However, there is often a trade-off between data accessibility and query delay, since most nodes in a MANET have only limited memory space. For example, a node may hold a part of the frequently accessed data items locally to reduce its own query delay. However, if there is only limited memory space and many of the nodes hold the same replica locally, then some data items would be replaced and missing. Thus, the overall data accessibility would be decreased. Hence, to maximize data accessibility, a node should not hold the same replica that is also held by many other nodes. However, this will increase its own query delay.

A node may act selfishly, i.e., using its limited resource only for its own benefit, since each node in a MANET has resource constraints, such as battery and storage limitations. A node would like to enjoy the benefits provided by the resources of other nodes, but it may not make its own resource available to help others. Such selfish behaviour can potentially lead to a wide range of problems for a MANET. Existing research on selfish behaviours in a MANET mostly focus on network issues. For example, selfish nodes may not transmit data to others to conserve their own batteries. Although network issues are important in a MANET, replica allocation is also crucial, since the ultimate goal of using a MANET is to provide data services to users. In general, if mobile nodes in a MANET have sufficient memory space to hold the replicas of all data items, then data replication can simultaneously increase data accessibility and reduce query delay. However, since most mobile nodes have limited memory space for replicas of data items, there is often a trade-off between data accessibility and query delay. This trade-off may lead to the selfish behaviour of mobile nodes in a MANET, e.g., using their limited memory space only for their own benefit. A node would like to enjoy the benefits provided by the resources of other nodes, but does not make its own resources available to others.

In this paper, we assume a mesoscale ad hoc network consisting of a few dozen mobile hosts, which is similar to a MANET employed in [9]. Thus, our methodology is mainly designed for such a mesoscale network. We verify the effectiveness of our proposed methods by simulation experiments. The technical contributions of this paper can be summarized as follows:

A. Degree of selfishness

We measure the integrated degree of selfishness by taking into account selfish behaviour and node distance.

B. SCF+ tree by utilizing all connected node

We devise a node levelling technique that takes into account the integrated degree of selfishness. During replica allocations, a node regards all connected nodes as its friends. Consequently, the proposed replica allocation strategy can utilize all connected nodes, including selfish nodes as well as non-selfish nodes.

C. Allocating replicas based on SCF+ Trees

We devise a novel replica allocation technique that uses SCF+ trees to improve query delay and data accessibility, while still incurring low communication cost.

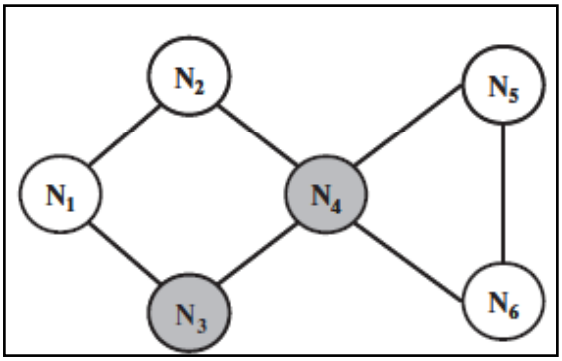
D. Verifying the Proposed Strategy

The conducted simulation results verify the efficacy of our proposed strategy.

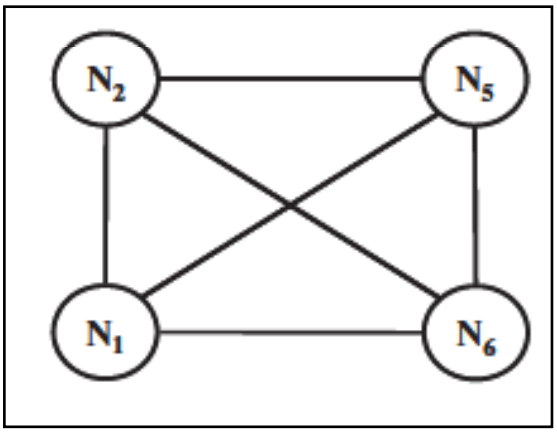
II. Preliminaries

A. System Model

Our system model is similar to that employed in [1] [9]. Each node has limited local memory space and simultaneously acts as a data provider and data consumer. Each node holds replicas of data items, and maintains the replicas in local memory space. Replicas are relocated during at a specific period, or relocation period [9]. There is no central server that determines the allocation of replicas. Any nodes freely join and organize a MANET. We model a MANET using an undirected graph G=(N, E) that consists of a finite set of nodes, N, and a finite set of communication links, E, where each element is a tuple of nodes in the network.



(a). Original Topology Network



(b). Partial Network Topology with Selfish Nodes

Similar to the work [5, 9], the following assumptions has been made:

Each node in a MANET has a unique identifier. The collection of nodes placed in a MANET is denoted as N = {N1,N2, . . . ,Nm}, where m is the total number of nodes.
All data items are of equal size, and each data item is held by a particular node as its original node. Each data item has a unique identifier, and the collection of all data items is denoted as D = {D1,D2, .. . ,Dn}, where n is the total number of data items.

Each node Ni ($1 \leq i \leq m$) has limited memory space for replica and original data items. The size of the memory space is denoted as Si.

Each node can hold only C, where $1 < C < n$, replicas in its memory space.

Selfishness in data forwarding is not considered, i.e., only selfishness in replica allocation is focused.

Each node has its own access frequency to each data item. The access frequency does not change.

Data items are not updated. This assumption is made for the sake of simplicity, i.e., we do not have to address data consistency or currency issues.

Similar to the work [1], we consider two types of selfish nodes: fully selfish and partially selfish nodes. We assume that the fully selfish nodes do not hold replicas allocated by other nodes, but allocate replicas to other nodes for their accessibility. The partially selfish nodes use some portion of their memory space for replicas allocated by othernodes.

B. SCF Techinque

The work builds on the SCF technique [1]. Thus, the outline of SCF technique in this section has been briefly described.

1. Detection of Selfish Node

To detect selfish nodes, the work [1] proposed the concept of credit risk (CR) score. A node computes CR score for every connected node. Based on the CR score, a node estimates the degree of selfishness for all connected nodes.
Prior to the CR score, we first need to compute the selfishness alarm of N_k on N_i, denoted as P_i^k.

$$P_i^k = \frac{\text{the number of } N_i\text{'s data requests not served by the expected node } N_k}{\text{the total number of } N_i\text{'s requests for data allocated to } N_k} \quad (1)$$

With the estimated selfishness alarm, the CR score is computed by Eq. (2).
In Eq. (2), SS_i^k and ND_i^k denote size of N_k's shared memory space and the number of N_k's shared data items for N_i, respectively.2 Note that both SSik and ND_i^k are N_i's estimated values, since N_k, which may be selfish or not, does not necessarily let N_i know the number of shared data items or size of the shared memory space. The system parameter, α, is used to adjust the relative importance of SS_i^k and ND_i^k

$$CR_i^k = \frac{P_i^k}{\alpha * SS_i^k + (1-\alpha) * ND_i^k}, \text{ where } 0 \leqslant \alpha \leqslant 1 \quad (2)$$

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Algorithm 1. Pseudo code to detect selfish nodes

```
At every relocation period
/*N_i detects selfish nodes with this algorithm*/
detection(){
    for (each connected node N_k){
        if (nCR_i^k < δ) N_k is marked as non-selfish;
        else N_k is marked as selfish;}
    wait until replica allocation is done;
    for (each connected node N_k){
        if (N_i has allocated replica to N_k){
            ND_i^k = the number of allocated replica;
            SS_i^k = the total size of allocated replica;}
        else{
            ND_i^k = 1;
            SS_i^k = the size of a data item;}}}
```

$$nCR_i^k = \frac{P_i^k}{\alpha * \frac{SS_i^k}{S_i} + (1-\alpha) * \frac{ND_i^k}{n_i}}, \text{ where } 0 \leqslant \alpha \leqslant 1 \qquad (3)$$

Algorithm 1 describes how to detect selfish nodes. At every relocation period, node Ni detects selfish nodes based on nCRik. Each node may have its own initial value of Pik as a system parameter. After replica allocation, Ni sets SSik and NDik accordingly. The two values are adjusted at every query processing time.

2. Building the SCF tree

Prior to building the SCF tree, each node makes its own partial topology graph G_i, which is a component of the graph G. Before constructing/updating the SCF tree, node N_i removes selfish nodes from G_i, using the nCR_i^k score. Thus, N_i changes G_i into its own partial Gins by a smoothing out operation in graph theory.
Based on Gins, N_i builds its own SCF tree. Each node has the depth of SCF tree. When N_i builds its own SCF tree, N_i first appends the nodes that are connected to N_i by one hop to N_i's child nodes. Then, N_i checks and appends the child nodes of the appended nodes recursively. At every relocation period, each node updates its own SCF tree based on the network topology of that moment.

3. Allocating replicas based on the SCF tree

After building SCF tree, a node allocates replica

Algorithm 2 Pseudo code to build SCF tree

```
00:/*N_i makes SCF-tree with a parameter, depth d*/
01: constructScfTree(){
02: append N_i to SCF-tree as the root node;
03: checkChildnodes(N_i);
04: return SCF-tree;}
05: Procedure checkChildnodes(Nj){
06: /*IN_j^a is a set of nodes that are adjacent to N_j */
07: for (each node N_a ϵ IN_j^a) {
08: if (distance between N_a and the root > d)
09: continue;
10: else if (N_a is an ancestor of N_j in T_i^SCF)
11: continue;
12: else {append N_a to T_i^SCF as a child of N_j;
13: checkChildnodes(N_a);}}
```

at every relocation period. Since the SCF tree based replica allocation is performed in a fully distributed manner, each node determines replica allocation individually without any communication with other nodes. Each node asks non-selfish nodes within its SCF tree to hold replica, when it cannot hold replica in its local memory space. Each node allocates replicas in descending order of its own access frequency.

III. Proposed Approach

A. Overview

In a MANET, the work [1] adopts the notion of selfishness alarm to estimate the degree of selfishness. Recall that the selfishness alarms is the ratio of some node's data requests being not served by the expected node. The selfishness alarms consist of true alarms caused by selfish behaviour and false alarms3 caused by disconnections [1]. Since the disconnection probability of a faraway node is higher than that of a near node [7], faraway nodes are more likely to cause false alarms than near nodes. Consequently, the work [1] suffers from many false alarms, since it fails to consider the probability of disconnections. To address this situation, we incorporate node distance (in terms of the number of hops) into the degree of selfishness.
By considering node distance and selfish behavior in an integrated manner. The expectation is to reduce false alarms significantly.
Based on the estimated integrated degree of selfishness, each node categorizes all connected nodes into some levels. Each node builds its own SCF+ trees for all connected nodes without explicit binary classification of (partial) selfish and non-selfish nodes. This is quite different from the previous work [1] where each node builds its own SCF tree with only nodes classified as non-selfish ones.
Next, each node allocates replicas to nodes in SCF+ trees by preferring higher level nodes, i.e., near and/or less selfish, to lower level nodes, i.e., faraway and/or more selfish. This deviates from the work [1] where each node allocates replicas in descending order of its own access frequency to non-selfish nodes in SCF tree. The expectation is that our new technique will reduce query delay and achieve high data accessibility, since it prefers near nodes, which are likely to support queries quickly without disconnections, and utilizes all connected nodes.

B. Measuring Degree of Selfishness

1. Extended CR (xCR) score

Each node uses the extended CR (xCR) score to estimate the integrated degree of selfishness for all connected nodes, including selfish nodes. Since faraway nodes are likely to be frequently disconnected and cause many false alarms, the distance has been incorporated in terms of number of hops, denoted as H, into the degree of selfishness. Thus, the following Eq. (4) is used to measure the integrated degree of selfishness:

$$xCR_i^k = nCR_i^k * \left(\frac{H_i^k}{max\ H_i}\right)^2, \text{where } 0 \leqslant \alpha \leqslant 1 \qquad (4)$$

C. Building SCF Tree

1. Node Leveling Technique

Based on the estimated integrated degree of selfishness, each node categorizes all connected nodes into some levels. The number of nodes per level is referred as the fit number of node Ni denoted as

fi. If fi is too small, Ni will suffer from the lack of the trustworthy nodes. Since many important data items, i.e., frequently used data items, will be held by untrustworthy nodes, it may lead to low data accessibility. Similarly, if fi is too large, Ni will also suffer from poor data accessibility, since many selfish nodes will not be handled appropriately. Specifically, a greedy choice has been made, i.e., used to keep the size of the first level (the most trustworthy level) sufficiently large to contain all data items, and try to make all other level sizes equal to the first level size plus one. Ni computes fi based on the estimated average of SSik for all connected nodes and total number of data items. Consequently, Ni calculates its fi and number of levels, denoted as ti, as follows:

$$f_i = \lceil \frac{total\ number\ of\ data\ items}{average\ of\ SS_i^k\ for\ every\ connected\ N_k} \rceil \quad (5)$$

$$t_i = \lceil \frac{number\ of\ all\ connected\ nodes}{f_i} \rceil \quad (6)$$

2. SCF+ TREE

To build SCF+ trees, each node uses its own partial network topology graph, Gi, and a collection of pairs (node ID, corresponding level). In the proposed strategy, a node builds its own SCF+ trees at every relocation period. In particular, NI builds its own per-level SCF+ tree.

The SCF+ trees is defined of N_i as $ST_i = \{ST_i^l \mid ST_i^l$ is a SCF+ tree of l^{th} level $1 \le l \le t_i\}$. The root node of ST_i^l is N_i for every l.

Algorithm 3 describes how to construct SCF+ trees of N_i, ST_i. In line 5, Ni initializes "visited" marks of all nodes in Gi as unvisited. In line 6 and 7, N_i marks itself as a visited and appends itself as the root node of ST_i^l. In line 8, the procedure checkChildNodes() is called to determine whether to append the next node to ST_i^l. In line 14 and 16, the procedure checks whether N_a is suitable for being appended to ST. In line 14, N_i checks whether N_a is visited or not. In line 15 and 16, if N_a is suitable for being appended to the current ST_i^l, N_i appends N_a to its own ST_i^l. If N_a is not a suitable node, N_i calls checkChildNodes () recursively with parameter N_a, l, and N_k.

In line 9, N_i appends ST_i^l to ST_i. Lastly, in line 10, the algorithm returns ST_i.

IV. Related Work

A most related work is [1] that identify the problem of selfish replica allocation in a MANET. The work proposes a suit of SCF tree based selfish replica allocation techniques. However, when it allocates replicas to other nodes, it uses SCF tree including only non-selfish nodes.

Algorithm 3: Pseudo code to SCF+ tree

```
/* Ni builds SCF+ trees with this algorithm */
/* Nja is a set of nodes that are adjacent to Nj */
/* Hja is the distance between Nj and Na */
Input: Gi and a collection of pairs (node ID,
  corresponding level)
Output: STi
01: constructSCF+Trees(){
02:   fi = ⌈total number of data items / average of SSik for every connected Nk⌉;
03:   ti = ⌈number of all connected nodes to Ni / fi⌉;
04:   for (l = 1; l ≤ ti; l++){
05:     initialize all nodes as unvisited;
06:     mark Ni as the visited node;
07:     append Ni to STil as the root node;
08:     checkChildnodes (Ni, l, Ni);
09:     append STil to STi;}
10:   return STi};
11: Procedure checkChildNodes (Nj, l, Nk){
12:   if (Nja is not an empty set){
13:     for (each node Na ∈ Nja){
14:       if (Na is unvisited){
15:         if (Na is the l-th level node of Ni){
16:           if (Hik < Hia and Hia ≥ Hka){
17:             mark Na as visited;
18:             append Na to STil as a child of Nk;
19:             checkChildnodes (Na, l, Na);}
20:         else{
21:           mark Na as visited;
22:           checkChildnodes (Na, l, Nk);}}}}}
```

Thus, although SCF-tree based strategy can handle selfish replica allocation effectively, SCF tree based replica allocation techniques lose the original network distance information. Through the performance evaluation, we empirically observe that the proposed solutions of [1] could be substantially improved. To handle the selfish nodes from the network perspective, various techniques [4] have been proposed. As described in [4], techniques handling selfish nodes can be classified into three categories: reputation-based [14, 15, 16], credit-payment [2,12], and game theory-based [8,17]. The work [5] proposes cooperative caching-based data access methods, including CachePath, CacheData, and Hybrid. However, all these works focus on the selfish behaviour from the network perspective, such as dropping or refusing to forward packets. Since disconnection probability of a faraway node is higher than that of a near node [7], the work [9] presents replica allocation techniques for a MANET, including static access frequency (SAF), dynamic access frequency and neighbourhood (DAFN), and dynamic connectivity based grouping (DCG). This work reports that DCG provides the highest data accessibility, while SAF incurs the lowest traffic among the three techniques.

V. Conclusion

The problem of selfish nodes has been addressed from the replica allocation perspective which is known as selfish replica allocation. The work was motivated by the fact that a selfish replica allocation could lead to overall poor data accessibility in a MANET. The proposed method is a selfish node detection method and novel replica allocation techniques to handle the selfish replica allocation appropriately. The proposed strategies are inspired by the real-world observations in economics in terms of credit risk and in human friendship management in terms of choosing one's friends completely at one's own discretion. Also the notion of credit risk applied from economics to detect selfish nodes. Every node in

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

a MANET calculates credit risk information on other connected nodes individually to measure the degree of selfishness. Since traditional replica allocation techniques failed to consider selfish nodes, the novel replica allocation technique proposed.

VI. Acknowledgment

This work was supported by department of information technology, B.S Abdur Rahman University, Vandaloor, Chennai.

References

[1] *J.-H. Choi, K.-S. Shim, S. Lee, K.-L. Wu, Handling selfishness in replica allocation over a mobile ad hoc network,*

[2] *E. Adar, B.A. Huberman, Free riding on Gnutella, First Monday 5 (10) (2000) 1–22.*

[3] *P. Padmanabhan, L. Gruenwald, A. Vallur, M. Atiquzzaman, A survey of data replication techniques for mobile ad hoc network databases, The International Journal on Very Large Data Bases 17 (5) (2008) 1143–1164.*

[4] *Y. Yoo, D.P. Agrawal, Why does it pay to be selfish in a MANET, IEEE Wireless Communications 13 (6) (2006) 87–97. routing protocols, in: Proceedings of ACM MobiCom, 1998, pp. 85– 97.*

[5] *G. Cao, L. Yin, C.R. Das, Cooperative cache-based data access in ad hoc networks, IEEE Computer 37 (2) (2004) 32–39. IEEE Transactions on Mobile Computing 11 (2) (2012) 278–291.*

[6] *G. Ding, B. Bhargava, Peer-to-peer file-sharing over mobile ad hoc networks, in: Proceedings of IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 104– 108.*

[7] *H.T. Friss, A note on a simple transmission formula, Proceedings of the Institute of Radio Engineers 34 (5) (1946) 254–256.*

[8] *D. Hales, From selfish nodes to cooperative networks – emergent link-based incentives in peer-to-peer networks, in: Proceedings of IEEE International Conference on Peer-to-Peer Computing, 2004, pp. 151–158,.*

[9] *T. Hara, Effective replica allocation in ad hoc networks for improving data accessibility, in: Proceedings of IEEE International Conference on Computer Communications, 2001, pp. 1568–1576.*

[10] *T. Hara, S.K. Madria, Data replication for improving data accessibility in ad hoc networks, IEEE Transactions on Mobile Computing 5 (11) (2006) 1515–1532.*

[11] *T. Hara, S.K. Madria, Consistency management strategies for data replication in mobile ad hoc networks, IEEE Transactions on Mobile Computing 8 (7) (2009) 950–967.*

[12] *W. Wang, X.-Y. Li, Y. Wang, Truthful multicast routing in selfish wireless networks, in: Proceedings of ACM MobiCom, 2004, pp. 245– 259.*

[13] *M. Li, W.-C. Lee, A. Sivasubramaniam, Efficient peer-to-peer information sharing over mobile ad hoc networks, In: Proceedings of WWW Workshop on Emerging Applications for Wireless and Mobile Access, 2004, pp. 2–6.*

[14] *Y. Liu, Y. Yang, Reputation propagation and agreement in mobile adhoc networks, in: Proceedings of IEEE Wireless Communications and Networking Conference, 2003, pp. 1510–1515.*

[15] *S. Marti, T. Giuli, K. Lai, M. Baker, Mitigating routing misbehaviour in mobile ad hoc networks, in: Proceedings of ACM MobiCom, 2000, pp. 255–265.*

[16] *K. Paul, D. Westhoff, Context aware detection of selfish nodes in DSR based ad-hoc networks, in: Proceedings of IEEE Global Telecommunications Conference, 2002, pp. 178–182.*

[17] *V. Srinivasan, P. Nuggehalli, C. Chiasserini, R. Rao, Cooperation in wireless ad hoc networks, in: Proceedings of IEEE International Conference on Computer Communications, 2003, pp. 808–817.*