Replica Allocation in MANETs for Eliminating Selfish Node

Mr. Ritesh Dhanare, Dr. Sunita Varma

Abstract—in wireless ad hoc networks, all nodes cooperate to provide network services. Due to the limited radio transmission range, data packets are usually forwarded through multiple relay nodes before they reach the destinations. If a node always serves as a relay to transmit the packets, it may quickly use up its own energy and other resources. Therefore, some nodes use a selfish approach, they try to avoid forwarding the packets. Such selfish behavior would probably cause the network to break down. Selfish nodes are common within ad hoc networks because they are managed by different resources. The performance of ad-hoc network degraded because of selfish node which causes large packet loss, low throughput, decrease data accessibility, high query delay, high communication cost, higher chances for network cresses etc.

In a mobile ad hoc network, the mobility and resource constraints of mobile nodes may lead to network partitioning or performance degradation. Several data replication techniques have been proposed to minimize performance degradation. Most of them assume that all mobile nodes collaborate fully in terms of sharing their memory space. In reality, however, some nodes may selfishly decide only to cooperate partially, or not at all, with other nodes. These selfish nodes could then reduce the overall data accessibility in the network. In this paper, we examine the impact of selfish nodes in a mobile ad hoc network from the perspective of replica allocation. We term this selfish replica allocation. In particular, we use a selfish node detection algorithm that considers partial selfishness and novel replica allocation techniques to properly cope with selfish replica allocation. The conducted simulations demonstrate the proposed approach outperforms traditional cooperative replica allocation techniques in terms of communication cost, throughput, packet delay and packet delivery ratio.

Index Terms— Mobile ad-hoc network, credit risk, SCF-Tree, DCG, SAF, degree of selfishness, selfish replica allocation.

1 INTRODUCTION

A "mobile ad hoc network" (MANET) is an autonomous system of mobile routers connected by wireless links. A MANETs is a peer-to-peer multi-hop mobile wireless network that has neither a fixed infrastructure nor a central server. Each node in a MANET acts as a router, and communicates with each other. MANET plays an important role in many environments and applications, especially, in critical settings that lack fixed network infrastructure, such as emergency rescue, humanitarian aid, as well as military and law enforcement.

The routers are free to move randomly and organize themselves arbitrarily, thus, the network's wireless topology may change rapidly. Nodes can appear, disappear and reappear as the time goes on and all the time the network connections should work between the nodes that are part of it. The network topology in an ad hoc wireless network is highly dynamic due to the movement of nodes; hence an on-going session suffers frequent path breaks. This situation often leads to frequent route changes. And there is one more constrain is there which is battery power of node which is limited that form a major constraint for the nodes in an ad hoc network. Devices used in these networks have restrictions on the power source in order to maintain portability, size and weight of the device. By increasing the power and processing ability makes the nodes bulky and less portable. So only MANETs nodes has to optimally use this resource.

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, this type of behavior of node is called selfish behavior. A node may act selfishly, i.e., using its limited resource only for its own benefit, since each node in a MANETs has resource constraints, such as battery and storage limitations. The performance of ad-hoc network degraded because of selfish node which causes large packet loss, low throughput, decrease data accessibility, high query delay, high communication cost, higher chances for network cresses etc. In this paper, we address the problem of selfishness in the context of replica allocation in a MANET, i.e., a selfish node may not share its own memory space to store replica for the benefit of other nodes. We can easily find such cases in a typical peer-to-peer application. For example, in Gnutella [1], nearly 70 percent of users do not share their storage for the benefit of others. The number of selfish users has increased to 85 percent of all Gnutella users over five years [2]. In this paper, we shall refer to such a problem as the selfish replica allocation. Simply, selfish replica allocation refers to a node's noncooperative action, such that the node refuses to cooperate fully in sharing its memory space with other nodes. To our knowledge, this work is one of few works [3] [4] to cope with selfish nodes in the context of replica allocation over a MANETs.

Fig. 1 illustrates an existing replica allocation scheme, DCG [5], where nodes N1,N2, . . .,N6 maintain their memory space M1,M2, . . .,M6 respectively, with the access frequency information in Table 1 (In Fig. 1, a straight line denotes a wireless link, a gray rectangle denotes an original data item, and a

- *Ritesh Dhanare is currently pursuing masters of engineering in computer engineering in S.G.S.I.T.S. Indore, India, PH-+91-9893851802. E-mail: riteshdhanare@yahoo.com.*
- *Dr. Sunita Varma is head of department of computer technology and application in S.G.S.I.T.S. Indore, India, PH-+91-9425056970. E-mail: sunita.varma19@gmail.com.*

white rectangle denotes a replica allocated. In Table 1, the gray colored area shows three data items that are accessed frequently by N3 and N4). As shown in Fig. 1, DCG seeks to minimize the duplication of data items in a group to achieve high data accessibility.

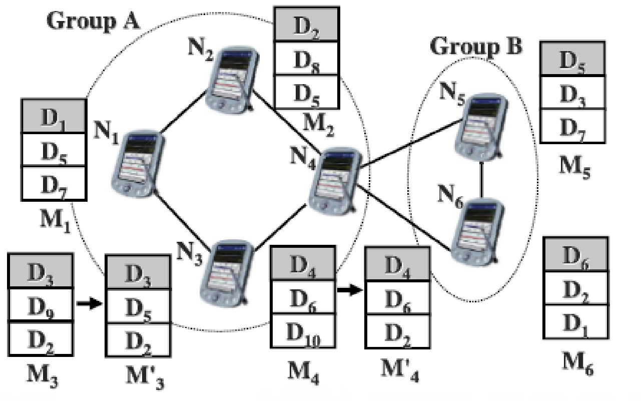
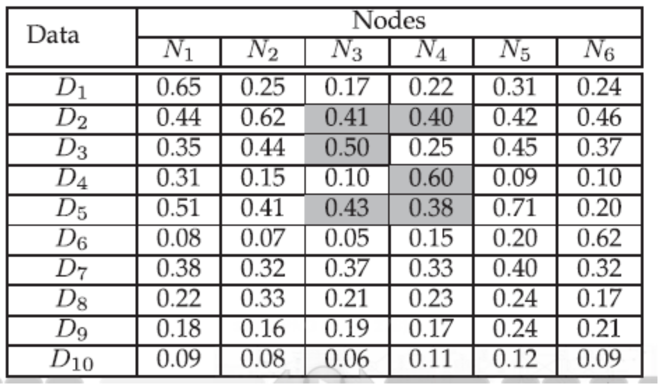


Figure 1: Example of selfish allocation replica [5].

Let us consider the case where N3 behaves "selfishly" by maintaining M'3, instead of M3, to prefer the locally frequently accessed data for low query delay. In the original case, D3, D9, and D2 were allocated to N3. However, due to the selfish behavior, D3, D5, and D2, the top three most locally frequently accessed items, are instead maintained in local storage. Thus, other nodes in the same group, i.e., N1, N2, and N4, are no longer able to access D9. This showcases degraded data accessibility, since N1, N2, and N4 cannot fully leverage N3's memory space as intended in cooperative replica sharing.
As another example, a node may be only "partially selfish" in a MANET. For instance, node N4 may want to locally hold D2, one of the locally frequently accessed data items. In this case, N4 uses only a part of its storage for its own frequently accessed data, while the remaining part is for the benefit of overall data accessibility. Thus, N4 may decide to maintain M'4 , instead of M4. Even with only partial selfishness, data accessibility is still degraded, since the other nodes in the same group, i.e., N1, N2, and N3, cannot access D10.

Table 1: Access frequency of node [5]



Data	Nodes					
	N_1	N_2	N_3	N_4	N_5	N_6
D_1	0.65	0.25	0.17	0.22	0.31	0.24
D_2	0.44	0.62	0.41	0.40	0.42	0.46
D_3	0.35	0.44	0.50	0.25	0.45	0.37
D_4	0.31	0.15	0.10	0.60	0.09	0.10
D_5	0.51	0.41	0.43	0.38	0.71	0.20
D_6	0.08	0.07	0.05	0.15	0.20	0.62
D_7	0.38	0.32	0.37	0.33	0.40	0.32
D_8	0.22	0.33	0.21	0.23	0.24	0.17
D_9	0.18	0.16	0.19	0.17	0.24	0.21
D_{10}	0.09	0.08	0.06	0.11	0.12	0.09

We believe that the partially selfish nodes (e.g., N4 in Fig. 1) should also be taken into account, in addition to the fully selfish nodes (e.g., N3 in Fig. 1), to properly handle the selfish replica allocation problem. We therefore need to measure the "degree of selfishness" to appropriately handle the partially selfish nodes. Motivated by this concept of "partial selfishness," we borrow the notion of credit risk (CR) [22] from economics to detect selfish nodes. Since the credit risk is calculated from several selfishness features in this paper, it can measure the degree of selfishness elaborately. In our scheme, a node can measure the degree of selfishness of another node, to which it is connected by one or multiple hops in a MANET.
We use replica allocation techniques with the developed selfish node detection method. They are based on the concept of a self-centered friendship tree (SCF-tree) and its variation to achieve high data accessibility with low communication cost in the presence of selfish nodes. The SCF-tree is inspired by our human friendship management in the real world. In the real world, a friendship, which is a form of social bond, is made individually [6]. For example, although A and B are friends, the friends of A are not always the same as the friends of B. With the help of SCF-tree, we aim to reduce the communication cost, while still achieving good data accessibility. The technical contributions of this paper can be summarized as follows:

- **Recognizing the selfish replica allocation problem:** We view a selfish node in a MANET from the perspective of data replication, and recognize that selfish replica allocation can lead to degraded data accessibility in a MANET.
- **Detecting the fully or the partially selfish nodes effectively:** We devise a selfish node detection method that can measure the degree of selfishness.
- **Allocating replica effectively:** We propose a set of replica allocation techniques that use the self-centered friendship tree to reduce communication cost, while achieving good data accessibility.
- **Verifying the proposed strategy:** The simulation results verify the efficacy of our proposed strategy.

The remainder of this paper is organized as follows: Section 2 describes the overview related work ……system model and the node behavior model from the viewpoint of selfish replica allocation. The proposed detection method and the replica allocation techniques are presented in Section 3. Section 4 evaluates the performance of our strategy. We briefly overview related work, and conclude the paper in Sections 5 and 6, respectively.

2 Related Work

Various algorithms have been designed in recent years to resolve the issue of selfish nodes. Each algorithm takes a different approach to the problem, but the majority of these algorithms can be broken into three general categories that is reputation based, credit based and game theory based.
2.1. Reputation Based: Marti et al. [7] are the first to introduce detection-based routing protocol enhancements for wireless ad hoc networks. They use a watchdog that identifies

misbehaving nodes and a pathrater that helps routing protocols avoid these nodes.

Buchegger and Le Boudec [8] propose a protocol, called CONFIDANT, to make misbehavior unattractive. They add observation, detection and reaction mechanisms to a routing protocol to exclude uncooperative nodes from the network. The security architecture is based on a distributed trust manager running on each node.

Michiardi and Molva [9] show a generic mechanism based on reputation to enforce cooperation among the nodes of a MANET to prevent selfish behavior. Each network entity keeps track of other entities' collaboration using a technique called reputation. Simple denial of service attacks are prevented by the collaboration technique.

B. Credit payment based: Buttyan and Hubaux [10] present a scheme to ensure cooperation among nodes in wireless ad hoc networks. They introduce a virtual currency called Nuglet, which is used to charge for the transmission of packets and to reward the forwarding process.

Zhong et al. [11] make one of the first proposals, which uses rewards to encourage cooperation among nodes in wireless ad hoc networks. The authors propose a virtual currency called Credits and a centralized account management via a Credit Clearance Service for all nodes.

Chen et al. [12] propose an auction-based incentive scheme (called iPass) to enable cooperative packet forwarding behavior in MANET. Each flow pays the market price of packet forwarding service to the intermediate routers. The resource allocation mechanism in ipass is based on the generalized Vickrey auction with reserve pricing.

C. Game theory based: Urpi et al. [13] develop a general model which formally describes the characteristics of wireless ad hoc networks. They analyze different cooperation enforcement mechanism from the literature and propose a simple strategy resulting in equilibrium. This indicates that in their model, cooperation is possible out of a node self's self-interest. Srinivasan et al. [14] obtain similar results. They use an algorithm based on the generous tit-for-tat (GTFT) strategy.

L. Anderegg et.al in [15] introduce a game-theoretic setting for routing in a wireless ad hoc network that consists of greedy, selfish agents who accept payments for forwarding data for other agents if the payments cover their individual costs incurred by forwarding data. In this setting, the authors propose Ad hoc-VCG, a reactive routing protocol that achieves the design objectives of truthfulness and cost-efficiency in a game theoretic sense by paying to the intermediate nodes a premium over their actual costs for forwarding data packets.

Felegyhazi et al. [16] investigate whether cooperation can exist in wireless ad hoc networks without incentive mechanisms. They propose a model based on game theory and graph theory to investigate equilibrium conditions for packet forwarding strategies. Their model is the first to consider the network topology. They find that in theory conditions for cooperation out of self-interest exist, but their simulation show that in practice these conditions are almost never satisfied and there will always be nodes which need an incentive to cooperate.

3 Background Study

3.1. System Model: In this, it is assume that each node has limited local memory space and acts as a data provider of several data items and a data consumer. Each node holds tables of data items, and maintains the tables in local memory space. The tables are relocated in a specific period. There are m nodes, N_1, N_2 . . . Nm and no central server determines the allocation of replica of table. Any node freely joins and organizes an open MANET. Using an undirected graph $G = (IN, IL)$ that consists of a finite set of nodes, IN, and a finite set of communication links, IL, where each element is a group (N_j, N_k) of nodes in the network, to model a MANET. Following assumptions are there:

- Each node in a MANET has a unique identifier. All nodes that are placed in a MANET are denoted by $N = \{ N_1, N_2, \ldots N_m\}$ 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 set of all data items is denoted by $D = \{D_1, D_2, \ldots D_n \}$, where n is the total number of data items.
- Each node N_i ($1 \le i \le m$) has limited memory space for table and original data items. The size of the memory space is S_i. Each node can hold only C, where $1 < C < n$, replica in its memory space.
- Each node N_i ($1 \le i \le m$) has its own access frequency to data item $D_j \in D$ ($1 \le j \le n$), AF_i^j . The access frequency does not change.
- Each node moves freely within the maximum velocity.

When a node N_i makes an access request to a data item (i.e., issuing a query), it checks its own memory space first. The request is successful when N_i hold the original or replica of the data item in its local memory. If it does not hold the original or replica of the data item, the request will be broadcasted. The request is also successful when N_i receives any reply from at least one node connected to N_i with single hop or multiple hops. The node holds the original or replica of the targeted data item. Otherwise, the request, or query processing, fails. When a node N_i receives a data access request, it performs any one of the following function.

- Serves the request by sending its original or replica if it holds the target data item (the data may go through multiple hops before reaching the requester).
- Forward the request to its neighbors if it does not hold the target data item.

3.2. Node Behavior: There are three types of behavioral states for nodes from the viewpoint of selfish replica allocation.

- **Type-1 node:** The nodes are non-selfish nodes. The nodes hold replicas allocated by other nodes within the limits of their memory space.
- **Type-2 node:** The nodes are fully selfish nodes. The nodes do not hold replicas allocated by other nodes, but allocate replicas to other nodes for their accessibility.
- **Type-3 node:** The nodes are partially selfish nodes. The nodes use their memory space partially for allocated replicas by other nodes. Their memory space may be divided logically into two parts: selfish and public area. These nodes allocate replicas to other nodes for their accessibility.

The detection of the type-3 nodes is complex, because they are not always selfish. In some sense, a type-3 node might be considered as nonselfish, since the node shares part of its memory space. In this paper, however, we have considered it as (partial) selfish, because the node also leads to the selfish replica allocation problem.

4 Proposed Strategy

Our strategy consists of three parts: 1) detecting selfish nodes, 2) building the SCF-tree, and 3) allocating replica. At a specific period, or relocation period [5], each node executes the following procedures:

I. Each node detects the selfish nodes based on credit risk scores.
II. Each node makes its own (partial) topology graph and builds its own SCF-tree by excluding selfish nodes.
III. Based on SCF-tree, each node allocates replica in a fully distributed manner.

The CR score is updated accordingly during the query processing phase. We borrow the notion of credit risk from economics to effectively measure the "degree of selfishness." In economics, credit risk is the measured risk of loss due to a debtor's nonpayment of a loan. A bank examines the credit risk of an applicant prior to approving the loan. The measured credit risk of the applicant indicates if node is creditworthy. We take a similar approach. A node wants to know if another node is believable, in the sense that a replica can be paid back, or served upon request to share a memory space in a MANET.

4.1 Detection of Selfish Node: The credit risk [17] can be described by the following equation.

In mathematical form it can be written as

Algorithm 1 describes how to detect selfish nodes. At each relocation period, node Ni detects selfish nodes based on credit risk. The estimated values are adjusted at query processing

$$Credit\ Risk = \frac{expected\ risk}{expected\ value}$$

time, according to Algorithm 2.

According to 1 first node will check the credit risk value by

$$nCR_i^k = \frac{P_i^k}{\alpha * \frac{SS_i^k}{S_i} + (1-\alpha) * \frac{ND_i^k}{n_i}}$$

expected value of risk, if it is greater less than the expected risk then the node will mark as nonselfish node else the node is selfish node, then node waits for replica allocation to be done. And if node is nonselfish then for each connected node it replicates the replica of the node its shared memory space and shared data item, else it does node replicate the shared data item and the shared memory space. And in algorithm 2, it updates the selfish node during route discovery, if any new node comes in the network then first it will treat as a nonselfish node and in route discovery if it serves the query then according to algorithm the shared data item added into the network and also shared memory size added into the network. If new node does not serves the query then increase the expected risk value of the node and remove the shared data item and memory size.

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 ($nCR_i^k < \delta$) $N_k$ is marked as non-selfish;
                else $N_k$ is marked as selfish ;}
                wait until next reallocation period;
        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;
}}}
```

Algorithm 2: Pseudo code to update selfish features

```
At every query processing time
/* When $N_i$ issues a query */
update () {
        while (during the predefined time ω) {
                if (an expected node $N_k$ serves the query)
                        decrease $P_i^k$ ;
                if (an unexpected node $N_j$ serves the query){
                        $ND_i^j = ND_i^j + 1$;
                        $SS_i^j = SS_i^j$ + (the size of a data
                                      item);
        }}
        if (an expected node $N_k$ does not serve the query) {
                increase $P_i^k$ ;
                $ND_i^k = ND_i^k$ - 1;
                $SS_i^k = SS_i^k$ - (the size of a data item);
}}
```

4.2 Building SCF-Tree: The SCF-tree [13] based replica allocation techniques are inspired by human friendship management in the real world, where each person makes his/her own friends forming a web and manages friendship by himself/herself. He/she does not have to discuss these with others to maintain the friendship. The decision is solely at his/her discretion. The main objective of our novel replica allocation techniques is to reduce traffic overhead, while achieving high data accessibility. If the novel replica allocation techniques can allocate replica without discussion with other nodes, as in a human friendship management, traffic overhead will decrease. Algorithm 3 describes how to generate SCF-Tree.

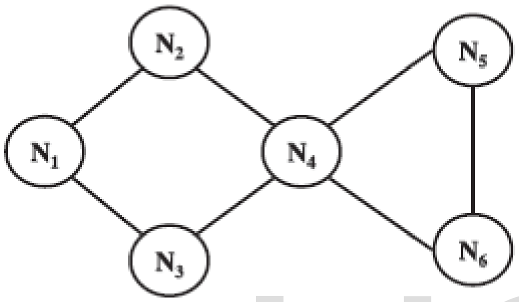


Figure 2: Sample topology G [17]

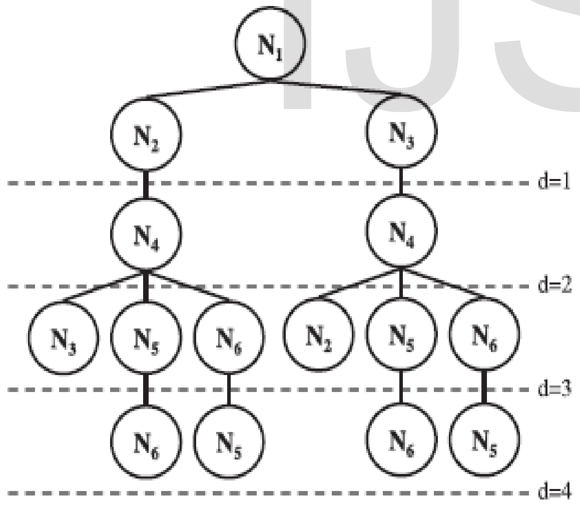


Figure 3: SCF-Tree for N1 [17]

Algorithm 3: Pseudo code to build SCF-tree

```
/* $N_i$ makes SCF-tree with a parameter, depth $d$ */
ScfTree () {
    append $N_i$ to SCF-tree as the root node;
    checkChildnodes ($N_i$);
    return SCF-tree ;}
Procedure checkChildnodes($N_i$){
/* $IN_i^a$ is a set of nodes that are adjacent nodes to $N_i$ */
for (each node $N_a \in IN_i^a$ ) {
    if (distance between $N_a$ and the root > $d$)
        continue;
    else if ($N_a$ is an ancestor of $N_i$ in $T_i^{SCF}$ )
        continue;
    else {append $N_a$ to $T_i^{SCF}$ as a child of $N_i$;
        checkChildnodes($N_a$);
}}}
```

4.3. Allocating Replica: After building the SCF-tree, a node allocates replica at every relocation period. Each node asks nonselfish nodes within its SCF-tree to hold replica when it cannot hold replica in its local memory space. 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. Algorithm 4 describes, how to allocate replica in the network.

Algorithm 4: Pseudo code for replica allocation

```
/* $N_i$ executes this algorithm at relocation period */
replica_allocation(){
    $L_i$ = make priority ($T_i^{SCF}$($N_i$ build its own SCF-
                                        Tree));
    for (each data item $\in ID_i$) {
        if ($M_s$ is not full)
            allocate replica of the data to $M_s$ ;
        else {/* $M_s$ is full */
            allocate replica of the data to the
            target node;
    /* the target node is selected from $L_i$ */
        if ($M_p$ is not full)
            allocate replica of the data to $M_p$; }
    }
while (during a relocation period) {
    if ($N_k$ requests for the allocation of $D_q$)
        replica_allocation_for_others ($N_k$, $D_q$) ;}}
Procedure make_priority ($T_i^{SCF}$) {
    for (all vertices in $T_i^{SCF}$ ) {
        select a vertex in $T_i^{SCF}$ in order of $BFS$;
        append the selected vertex id to $L_i$; }
    return $L_i$; }
Procedure replica_allocation_for_others ($N_k$, $D_q$) {
    if ($N_k$ is in $T_i^{SCF}$ and $N_i$ does not hold $D_q$) {
        if ($M_p$ is not full) allocate $D_q$ to $M_p$ ;
        else {/* $M_p$ is full */
            if ($N_i$ holds any replica of local
               interest in $M_p$)
            replace the replica with $D_q$ ;
            else {
                /* $N_h$ is the node with the highest $nCR_i^h$ among the
                nodes which allocated replica to Mp */
                if ($nCR_i^h$ >$nCR_i^k$)
            replace the replica requested by $N_h$
            with $D_q$;
}}}}
```

5 Simulation Parameter

Simulation Environment:

Table 2: Simulation Parameter

Parameter	Value
NS2 Version	2.35
Network Area	250000 m^2
No. of nodes	70
Node Density	7/25000 node per m^2
Transmission Range	250 m
Routing Protocol	AODV
Simulation Time	10 s
Data Type	CBR
Bandwidth	512-2048 kbps

Performance Metrics:

- **Communication Cost:** This is the total hop count of data transmission for selfish node detection and replica allocation/relocation, and their involved information sharing.
- **Throughput:** It the average rates of successful message delivery over a communication channel.
- **Packet delay:** Total delay added by intermediate node during transmission.
- **Packet Delivery Ratio:** Ratio of total number of received packet to total number of sended packets.

Communication cost can be determined by following technique.

- **Static Access Frequency (SAF) [5]:** Each node allocates replica based only on its own access frequency, without considering or detecting selfish nodes. This allocation technique is expected to show the optimal performance in terms of communication cost, because the technique does not communicate with others to allocate replica.
- **Dynamic Connectivity-based Grouping (DCG) [5]:** DCG creates groups of nodes that are biconnected components in a network, without considering or detecting selfish nodes. In each group, the node, called coordinator, allocates replicas based on the access frequency of the group. This technique is known to have high data accessibility.
- **Dynamic Connectivity-based Grouping with detection (DCG+):** The technique combines DCG with our detection method. Initially, groups of nodes are created according to the DCG methodology. Subsequently, in each group, selfish nodes are detected based on our detection method. For the detection, each node in a group sends its *n*CR scores to the coordinator with the lowest suffix of node identifier in the group [18]. The coordinator excludes selfish node(s) from the group for replica allocation. As a result, only nonselfish nodes form a group again. The replica allocation is only performed within the final group without any selfish nodes. After replica allocation, the coordinator shares the information of replica allocation with group members for the subsequent selfishness detection.

6 Simulation Result

6.1 Communication Cost: We evaluate several replica allocation techniques in terms of communication cost. Our intuition was that our techniques outperform SAF, while being inferior to SCF. This intuition is confirmed by the results in figure 4.SAF shows the worst performance in all cases, since group members need to communicate with each other in detecting selfish nodes and allocating/relocating replica. We report that, on average, about 70 percent of total communication cost in the SAF technique is caused by replica allocation/relocation, while about 30 percent is caused by selfish node detection. As expected, DCG shows the best performance, since no detection of selfish nodes or group communication is made. Although SCF and DCG techniques show better performance than SAF in communication cost, they are expected to show poor performance in data accessibility in the presence of selfish nodes. Interestingly, our analysis reveals that our techniques, which detect selfish nodes, considerably outperform DCG, which does not perform the selfishness detection procedure. This verifies the efficacy of our fully distributed way of detecting selfish nodes and allocating replica, i.e., no group communication.

6.2 Throughput: In figure 5, the graph shows that the difference of normal network, when selfish node comes in the network and after detection and elimination of selfish node from the network. It clearly shows that the performance improvement in the network.

6.3 Packet delay: In figure 6, the graph shows that the packet delay in all three cases that is normal network, when selfish node comes in the network and after detection of selfish node in the network.

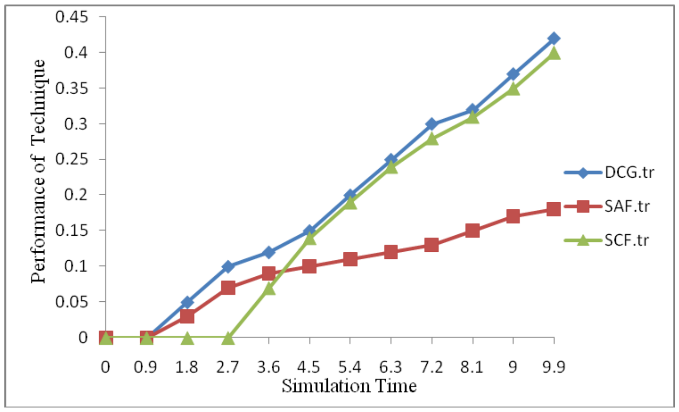


Figure 4: Evaluation of communication cost

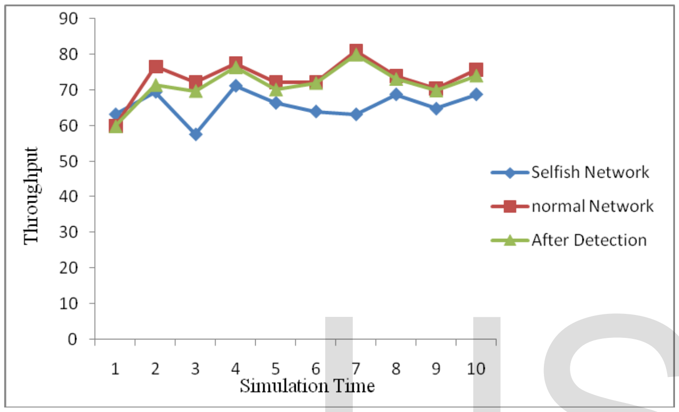


Figure 5: Evaluation of Throughput

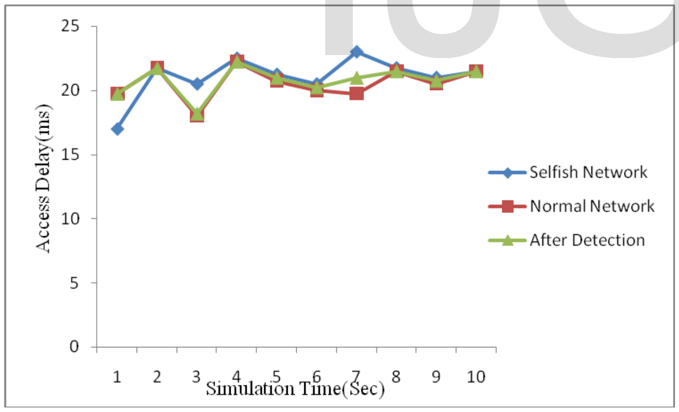


Figure 6: Access Delay

Packet Delivery Ratio: In figure 7, the packet delivery ratio in all three cases can be seen. In that graph in normal network the ratio of packet delivered is high as compare to other two cases, and the when selfish node comes into the network then the packet delivery ratio is very low as compare to other two cases.

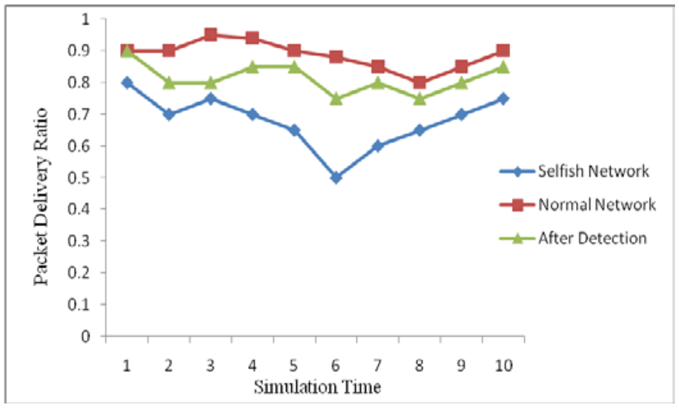


Figure 7: Packet Delivery Ratio

7 Conclusion

In contrast to the network viewpoint, we have addressed the problem of selfish nodes from the replica allocation perspective. We term this problem selfish replica allocation. Our work was motivated by the fact that a selfish replica allocation could lead to overall poor data accessibility in a MANET. We have proposed 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. We applied the notion of credit risk from economics to detect selfish nodes. Every node in 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.

References

[1] E. Adar and B.A. Huberman, "Free Riding on Gnutella," First Monday, vol. 5, no. 10, pp. 1-22, 2000.

[2] M. Feldman and J. Chuang, "Overcoming Free-Riding Behavior in Peer-to-Peer Systems," SIGecom Exchanges, vol. 5, no. 4, pp. 41-50, 2005.

[3] H. Li and M. Singhal, "Trust Management in Distributed Systems," Computer, vol. 40, no. 2, pp. 45-53, Feb. 2007.

[4] A. Mondal, S.K. Madria, and M. Kitsuregawa, "An Economic Incentive Model for Encouraging Peer Collaboration in Mobile- P2P Networks with Support for Constraint Queries," Peer-to-Peer Networking and Applications, vol. 2, no. 3, pp. 230-251, 2009.

[5] T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," Proc. IEEE INFOCOM, pp. 1568- 1576, 2001.

[6] R.F. Baumeister and M.R. Leary, "The Need to Belong: Desire for Interpersonal Attachments as a Fundamental Human Motivation," Psychological Bull., vol. 117, no. 3, pp. 497-529, 1995.

[7] S. Marti, T.J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks", Proc. of Mobicom, August 2000.

[8] S. Buchegger, J.-Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness In Distributed Ad-hoc NeTworks", Proc. of MobiHoc, June 2002.

[9] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks", Proc. of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security, pp.107-121, September 2002.

[10] L. Buttyan and J.P. Hubaux, "Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks", Technical Report No. DSC/2001/001, Swiss Federal Institute of Technology (EPFL), January 2001.

[11] S. Zhong, J. Chen, and Y.R. Yang, "Sprite: A Simple, Cheat-Proof, Credit Based System for Mobile Ad-Hoc Networks", Proc. of IEEE INFOCOM, April 2003.

[12] K. Chen and K. Nahrstedt, "iPass: an Incentive Compatible Auction Scheme to Enable Packet Forwarding Service in MANET", Proc. of the 24th International Conference on Distributed Computing Systems, March 2004.

[13] A.Urpi, M.Bonuccelli, and S.Giordano, "Modeling cooperation in mobile ad-hoc networks: A formal description of selfishness", Proc. of Modeling and Optimal in Moile, Ad Hoc and Wireless Networks, April 2003.

[14] N.B. Salem, L. Buttyyan, J.P. Hubaux, and M. Jakobsson, "A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks", Proc. of MobiHoc, June 2003.

[15] L. Anderegg and S. Eidenbenz, "Ad hoc-VCG: a Truthful and Cost-Efficient Routing Protocol for Mobile Ad hoc Networks with Selfish Agents", Proc. Of Mobicom, September 2003.

[16] M. Felegyhazi, L. Buttyan, and J.P.Hubaux, "Equilibrium Analysis of Packet Forwarding Strategies in Wireless Ad Hoc Networks- the static Case", Proc. of Personal Wireless Communication, October 2003.

[17] Jae-Ho Choi, Kyu-Sun Shim, SangKeun Lee, and Kun-Lung Wu, "Handling Selfishness in Replica Allocation over a Mobile Ad Hoc Network " ieee transactions on mobile computing, vol. 11, no. 2, feb-ruary 2012.

[18] T. Hara and S.K. Madria, "Data Replication for Improving Data Ac-cessibility in Ad Hoc Networks," IEEE Trans. Mobile Computing, vol. 5, no. 11, pp. 1515-1532, Nov. 2006.