HANDLING SELFISHNESS IN REPLICA ALLOCATION OVER A MANET

K.N.Anandhapriya PG Scholar, B.A.Sapna Assistant Professor

Abstract— In a MANET (Mobile Ad Hoc Network), the mobility along with source constraints of mobile nodes may escort to network partitioning or performance squalor. There are several data replication techniques that have been proposed to curtail the performance squalor. Most of them assume that all mobile nodes collaborate fully in terms of sharing their memory space. But, somehow some nodes may act selfishly and decide only to cooperate partially or not at all with other nodes. These selfish nodes could then lessen the overall data accessibility in the network. Here scrutinize the bang of selfish nodes in a Mobile Ad Hoc Network from the perspective of replica allocation. Hence this term is named as selfish replica allocation. In particular, budding a selfish node detection algorithm that considers partial selfishness and novel replica allocation techniques to properly hack it with selfish replica allocation. The conducted simulations reveal the anticipated loom outperforms traditional cooperative replica allocation techniques in terms of data accessibility, cost of communication and average query delay.

Key Words— Mobile Ad-Hoc networks, degree of selfishness, selfish replica allocation, credit risk.

1 Introduction

Mobile Ad Hoc Networks (MANETs) have attracted a lot of attention due to the popularity of mobile devices and the advances in wireless communication technologies [13], [14], [31]. A MANET is a peer-to-peer multihop mobile wireless network that has neither a fixed infrastructure nor a server at the center that is central server. Each node in a MANET acts as a router, and communicates with each other. A large variety of MANET applications have been developed. A MANET can be used in special situations, where installing infrastructure may be difficult, or even infeasible. A mobile peer-to-peer file sharing system is another interesting MANET application [9], [19].

Network partitions can occur often, since nodes move freely in a MANET, causing some data to be frequently difficult to get to some of the nodes. Hence, data accessibility is often an vital concert metric in a MANET [12]. Data are more often than not replicated at nodes, other than the original owners, to enhance data accessibility to cope with frequent network partitions. A substantial amount of research has recently been projected for replica allocation in a MANET [12] [13] [32].

In general, replication can concurrently improve data accessibility and condense query delay, i.e., query response time, if the mobile nodes in a MANET mutually have suffi cient memory space to hold both all the replicas and the original data. For example, the response time of a query can be significantly abridged, if the query accesses a data item that has a locally stored replica. However, there is habitually a trade-off between data accessibility and query delay, since most nodes in a MANET have only restricted memory space [32]. For example, a node may embrace a part of the frequently accessed data items locally to trim down its own query delay. However, if there is only inadequate memory space and many of the nodes hold the same replica locally, then some data items would be replaced and gone astray. Thus, on the whole 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 behavior can potentially lead to a wide range of problems for a MANET. Existing research on selfish behaviors in a MANET mostly focus on network issues [2], [11], [20]. 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 this paper, they 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

- *K.N.Anandhapriya PG Scholar, Department of Electronics and Communication Engineering,RVS College of Engineering and Technology, Coimbatore.*
  E-mail : priyanages@gmail.com
- *B.A.Sapna Assistant Professor, Departmaent of Electronics and Communication Engineering, RVS College of Engineering and Technology,Coimbatore.*

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 [10]. 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 [18] [25] to cope with selfish nodes in the context of replica allocation over a MANET.

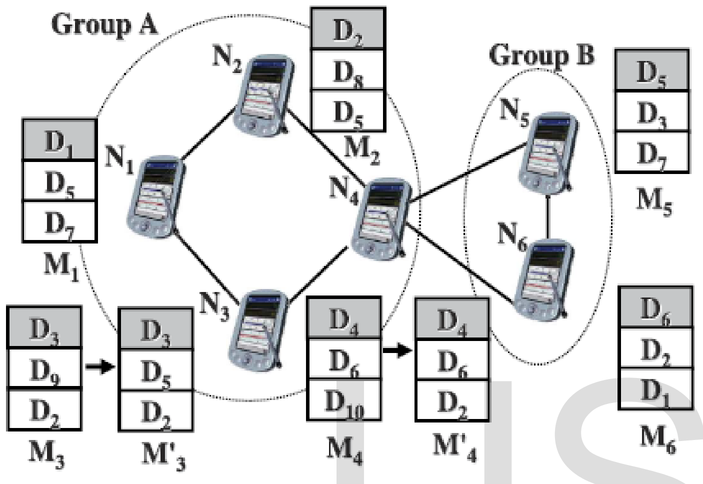

Fig 1: Example of selfish replica alloacation

Fig. 1 illustrates an existing replica allocation scheme, DCG [12], where nodes $N_1, N_2, \ldots, N_6$ maintain their memory space $M_1, M_2, \ldots, M_6$, 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 white rectangle denotes a replica allocated. In Table 1, the gray colored area shows three data items that are accessed frequently by N_3 and N_4). As shown in Fig. 1, DCG seeks to minimize the duplication of data items in a group to achieve high data accessibility.

Let us consider the case where N_3 behaves "selfishly" by maintaining M'_3, instead of M_3, to prefer the locally frequently accessed data for low query delay. In the original case, D_3, D_9, and D_2 were allocated to N_3. However, due to the selfish behavior, D_3, D_5, and D_2, the top three most locally frequently accessed items, are instead maintained in local storage. Thus, other nodes in the same group, i.e., N_1, N_2, and N_4, are no longer able to access D_9. This showcases degraded data accessibility, since N_1, N_2, and N_4 cannot fully leverage N_3'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 N_4 may want to locally hold D_2, one of the locally frequently accessed data items. In this case, N_4 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, N_4 may decide to maintain M'_4, instead of M_4. Even with only partial selfishness, data accessibility is still degraded, since the other nodes in the same group, i.e., N_1, N_2, and N_3, cannot access D_{10}.They believe that the partially selfish nodes (e.g., N_4 in Fig. 1) should also be taken into account, in addition to the fully selfish nodes (e.g., N_3 in Fig. 1), to properly handle the selfish replica allocation problem.

Therefore need to measure the "degree of selfishness" to appropriately handle the partially selfish nodes. Motivated by this concept of "partial selfishness," here borrowing 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.

TABLE:1
Access Frequency of Nodes

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

Here devise novel 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 lowcommunication 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 [4]. 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 SCFtree, and the aim is 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:They view a se fish 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: They devise a selfish node detection method that can measure the degree of selfishness.
- Allocating replica effectively: They propose a set of replica allocation techniques that use the selfcentered 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 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.

2 Literature Review

(Bo Zhu, Sanjeev Setia, Sushil Jajodia, Sankardoas Roy, 2010) proposed Localized Multicast for detecting node replication attacks. It is more efficient in terms of Communication and memory costs in large-scalar networks. Also achieves a higher probability of detecting node replicas. (Abdelouahid and Derhab and Nadjib Badache, 2009) proposed a classification scheme that categorizes the protocols into various classes. Also achieves both high availability and scalability. (Stephan Eidenbenz, Giovanni Resta and Paolo Santi, 2008) proposed the COMMIT based on the VCG payment to achieve truthfulness for sensor node. COMMIT is to achieve truthful and energy-efficient routing in ad hoc networks. (Prasanna Padmanaban, Le Gruenwald and Mohammed atiquzzaman, 2008) proposed data replication techniques for MANET databases.It is to maximize the number of transactions meeting their deadline constraints while minimizing the energy consumption of all mobile hosts. (Lakshmi Santhanam, Bin Xie and Dharma P.Agarwal, 2008) proposed credit based, reputation based and game theory based techniques to provide high bandwidth and reliable service. Also to provide security for the replication process. (Takahiro Hara and Sanjay K. Madria, 2006) proposed three replica allocation methods in Ad Hoc networks and achieves high data accessibility. Also here each data item is randomly updated and the mobile user behaves based on their schedules. (Srdjan Capkun and Jean-Pierre Hubaux, 2006) proposed Verifiable Multialteration (VM) mechanism and achieves secure positioning in wireless networks and also for sensor networks. VM enables for the secure computation and verification opf node position in the presence of attackers. (Michal Feldman and John Chuang, 2005) proposed incentive mechanisms to achieve factors affecting the degree of free riding and to encourage user cooperation. Also achieves the design of distributed systems consisting of rational participants with diverse and selfish interests. (Byung-Gon Chun and John Kubiatowicz, 2004) proposed selfish caching techniques to detect the selfish nodes. Also here introduced the non-cooperative game model to characterize the caching problem among selfish serverswithout any central coordination. (Charles E.Perkins, Elizabeth M.Royer and Samir R.das and Mahesh K.Marina, 2001) proposed the performance of dynamic source routing and Ad Hoc On-demand Distance Vector Protocols. Here both of them used on-demand Route discovery, but with different mechanisms. (Eytan Adar and Bernardo A.Huberman, 2001) proposed a novel technique and analyzed user traffic in gnutella and concluded that there is asignificant amount of free riding in the system. Also found that nearly 70% of Gnutella users share no files, and nearly 50% of all responses are returned by the top 1% of sharing hosts and that free riding is distributed evenly between domains, so that no one group contributes significantly more than others.

3 Materials and Methodology

3.1 Node Behavior Model

This work considers only binary behavioral states for selfish nodes from the network routing perspective: selfish or not (i.e., forwarding data or not). Therefore, they define three types of behavioral states for nodes from the viewpoint of selfish replica allocation:

- **Type-1 node**: The nodes are nonselfish 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.

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, each node executes the following procedures:

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

3.2 DETECTING SELFISH NODE

The notion of credit risk can be described by the following equation:

$$CreditRisk = \frac{\exp ectedrisk}{\exp ectedvalue}$$

In our strategy, each node calculates a CR score for each of the nodes to which it is connected. Each node shall estimate the "degree of selfishness" for all of its connected nodes based on the score. Here first describing the selfish features that may lead to the selfish replica allocation problem to determine both expected value and expected risk. Selfish features are divided into two categories: nodespecific and query processing-specific. Node-specific features can be explained by considering the following case: A selfish node may share part of its own memory space, or a small number of data items, like the type-3 node. In this case, the size of shared memory space and/or the number of shared data items can be used to represent the degree of selfishness. In our approach, the size of N_k's shared memory space, denoted as SS^k_i, and the number of N_k's shared data items, denoted as ND_i^k, observed by a node N_i, are used as node-specific features.3 Note that both SS^k_i 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 node-specific features can be used to represent the expected value of a node. For instance, when node N_i observes that node N_k shares large SS^k_i and ND_i^k , node N_k may be treated as a valuable node by node N_i. As the query processing-specific feature, we utilize the ratio of selfishness alarm of N_k on N_i, denoted as P_i^k, which is the ratio of N_i's data request being not served by the expected node N_k due to N_k's selfishness in its memory space (i.e., no target data item in its memory space).4 Thus, the query processing-specific feature can represent the expected risk of a node. For instance, when P_i^k gets larger, node N_i will treat N_k as a risky node because a large P_i^k means that N_k cannot serve N_i's requests due to selfishness in its memory usage. To effectively identify the expected node (s), N_i should know the (expected) status of other nodes' memory space. Our SCF-tree-based replica allocation techniques, fortunately, support this assumption.

3.3 BUILDING SCF-TREE

The SCF-tree 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.

Based on Gns i , Ni builds its own SCF-tree, denoted as TSCF i. Algorithm 3 describes how to construct the SCF-tree. Each node has a parameter d, the depth of SCF-tree.

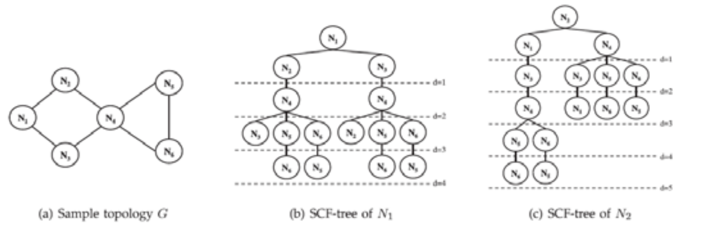

Fig : 2 Example of a self-centered friendship tree

When Ni builds its own SCF-tree, Ni first appends the nodes that are connected to Ni by one hop to Ni's child nodes. Then, Ni checks recursively the child nodes of the appended nodes, until the depth of the SCF-tree is equal to d. Fig. 2 illustrates the network topology and some SCF-trees of N1 and N2 in Fig. 1. In this example, assuming that all nodes are non-selfish nodes for simplicity. As can be seen in Figs. 2b and 2c, the SCF-tree may have multiple routes for some nodes from the root node. For example, in Fig. 2b, N1 has two routes to N2 when N1 sets its own parameter d to be 4. Since the multiple routes confer high stability [12], they allocate more replicas to the nodes that have multiple routes from the root node. At every relocation period, each node updates its own SCF-tree based on the network topology of that moment.

3.4 ALLOCATING REPLICA

After building the SCF-tree, a node allocates replica at every relocation period. Each node asks non-selfish 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.

Since every node has its own SCF-tree, it can perform replica allocation at its discretion. For example, in Fig. 2, after building the SCF-tree in Fig. 2b, N_1 may ask N_2 to hold some replicas. Note that the decision, whether to accept the replica allocation request or not, will be made at N_2's discretion (if N_2 is selfish, it may not accept the replica allocation request). Afterward, node N_1 may issue a query for the replicas. At this time, N_1 is likely to recognize whether the expected N_2 serves the query (i.e., non-selfish) or not (i.e., selfish). By observing the behavior of N_2, N_1 updates ND_1^2, SS_1^2, and P_1^2 accordingly.

Since they assume that a node can use some portion of its memory space selfishly, they may divide memory space M_i for replica logically into two parts: selfish area M_s and public area M_P. Each node may use its own memory space M_i freely as M_s and/or M_P. In each node, M_s will be used for data of local interest (i.e., to reduce query delay), while M_P for public data is asked to hold data by other node(s) (i.e., to improve data accessibility). A type-2 node uses M_i for only M_s, whereas a type-3 node uses M_i for M_s and M_p. Type-1 node's M_i will be equal to M_p.

4 RESULTS AND DISCUSSIONS

4.1 PERFORMANCE EVALUATION

Our simulation model is similar to that employed. In the simulation, the number of mobile nodes is set to 40. The data access frequency is assumed to follow Zip*f* distribution. The default relocation period is set to 256 units of simulation time which we vary from 64 to 8,192 units of simulation time. Table 2 describes the simulation parameters.

They evaluate our strategy using the following three 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.
- **Average query delay:** This is the number of hops from a requester node to the nearest node with the requested data item. If the requested data item is in the local memory of a requester, the query delay is 0. We only consider successful queries, i.e., it is the total delay of successful requests divided by the total number of successful requests.
- **Data accessibility:** This is the ratio of the number of successful data requests to the total number of data requests.

During 50,000 units of simulation time, we simulate and compare the proposed replica allocation strategies (i.e., SCF, SCF-DS, SCF-CN, and eSCF) with the following techniques:

- **Static Access Frequency (SAF) :** 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):** 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 nCR scores to the coordinator with the lowest suffix of node identifier in the group. The coordinator excludes selfish node(s) from the group for replica allocation. As a result, only non-selfish nodes form a group again. The replica allocation is only performed within the final group without any selfish nodes.

TABLE : 2
SIMULATION PARAMETER

Parameter (Unit)	Value (Default)
Number of Nodes	40
Number of Data items	40
Radius of Communication range (m)	1~19 (7)
Size of the network (m)	50 * 50
Size of memory space (data items)	2~40 (10)
Percentage of selfish nodes (%)	0~100 (70)
Maximum velocity of a node (m/s)	1
Relocation period	64~8,192 (256)
Zip*f* parameter	0.8

4.1.1 Communication Cost

There is no decisive difference among our techniques, except that the e-SCF technique shows the worst behavior. The other techniques (SCF, SCF-DS, and SCF-CN) show very similar communication cost, since they are all based on the same SCF-tree structure. Note that the consideration of selfishness degree in the SCF-DS technique does not affect the performance significantly, since nodes in the SCF-tree are sufficiently non-selfish to hold allocated replicas in many cases.

Communication cost decreases in every technique, except SAF, as the relocation period gets longer, since the frequency of selfish node detection and replica allocations decreases with a large relocation period. As communication cost increases as local memory size increases at first, but it decreases from a certain memory size (around 20 in our analysis) in every technique, except SAF. When the memory size is larger than a certain memory size, each node holds replicas of many data items and thus replica relocation rarely occurs.

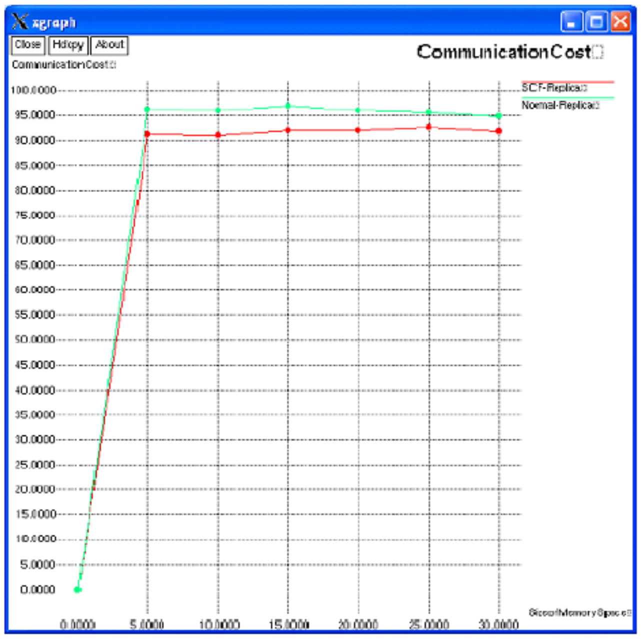

Fig : 3 Communication Cost

4.2.2 Average Query Delay

Average query delay for various parameters. As expected, the SAF technique shows the best performance in terms of query delay, since most successful queries are served by local memory space. Our techniques show slightly better query delay thandoes the DCG technique. The DCG+ technique shows the worst performance. This can be explained as follows: the distance in hop counts among group members in the DCG+ technique is longer than that in the DCG technique. Since most successful queries are served by group members in these techniques, the long distance among group members affects query delay negatively. Among our techniques, the eSCF technique shows the best average query delay. In the eSCF technique, nearby selfish nodes can be added to the eSCF-tree. Consequently, some queries are possibly served by the nearby (partially) selfish nodes, whereas only non-selfish nodes, which maybe far away, serve queries in other techniques.

Our intuition was that query delay decreases as the size of memory space increases. As the size of memory space increases, many nodes will accept replica allocation/relocation requests, since the size of public memory space increases as well. As a result, more queries are served by nearby nodes or locally. Here they have done an in-depth analysis for this situation. We found that, in the DCG and DCG+ techniques, the number of successful queries being served by some (non-selfish) nodes out of groups increases with more selfish nodes. That is, the profit of DCG is considerably hampered by many selfish nodes, since the biconnected component becomes non-effective. However, in our techniques, the number of successful queries being locally served increases slightly. This because when the number of nodes in the SCF-tree is very small, the local public memory space may be used for data items of local interest temporarily.

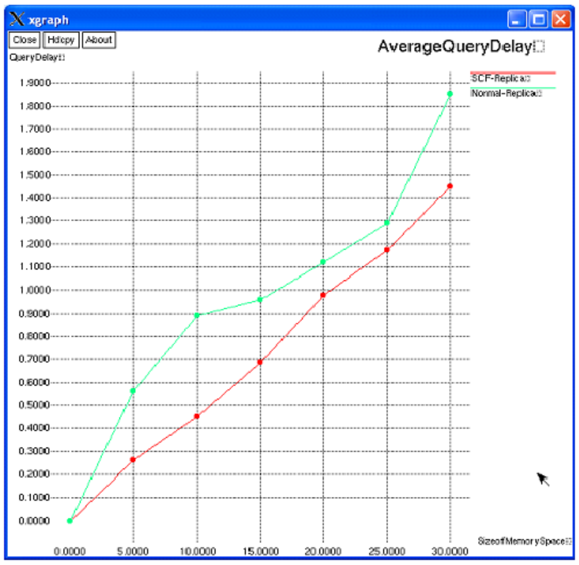

Fig : 4 Average Query Delay

4.2.3 Data Accessibility

Here evaluating the data accessibility of replica allocation methods under consideration. We expect that our techniques perform significantly better than other techniques in the presence of selfish nodes. The strength of our methodology: in all cases, our techniques outperform SAF, DCG, and DCG+ considerably, since our techniques can detect and handle selfish nodes in replica allocation effectively and efficiently. Among our techniques, the eSCF technique shows a slightly poorer performance.

Our initial intuition was that, data accessibility is stable with relocation periods. This is confirmed by the results in shows that data accessibility is proportional to the size of memory space, as expected. The performance of our techniques improves faster than do others, since our techniques fully utilize the memory space of nodes. The robustness of our techniques with respect to varying percentage of selfish nodes. The profit of DCG technique is considerably hampered by selfish nodes, whereas the SAF technique is insensitive at all.

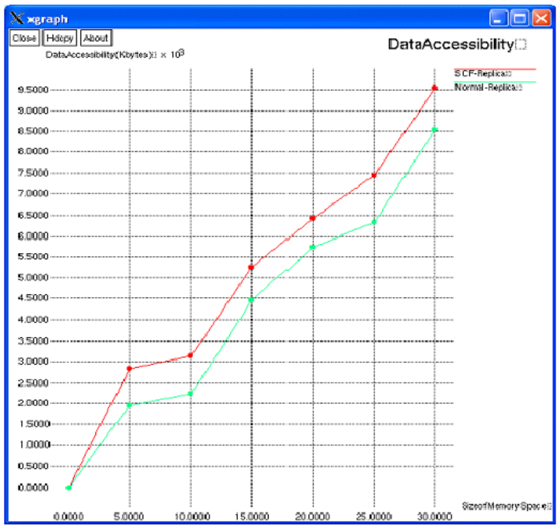

Fig : 5 Data Accessibility

5 RELATED WORK

5.1 Selfish Nodes from a Network Perspective

MANETs are divided into two categories: closed and open in the work. In a closed MANET, all nodes voluntarily participate in and organize the network. However, in an open MANET, which we consider in this paper, however, individual nodes may have different objectives. In this case, some nodes can be selfish to preserve their own resources. Various techniques have been proposed to handle the problem of selfish behavior from the network perspective. As described in, techniques handling selfish nodes can be classified into three categories: reputation based, credit-payment, and game theory-based techniques. In reputation-based techniques, each node observes the behaviors of others and uses the acquired information for routing. In credit-payment techniques, each node gives a credit to others, as a reward for data forwarding. The acquired credit is then used to send data to others. The game theory-based techniques assume that all rational nodes can determine their own optimal strategies to maximize their profit. The game theory-based techniques want to find the Nash Equilibrium point to maximize system performance. All these techniques focused on packet forwarding. In contrast, this paper focuses on the problem of selfish replica allocation.

5.2 Replica Allocation and Caching Techniques

In the pioneering work, some effective replica allocation techniques are suggested, including static access frequency, dynamic access frequency and neighborhood (DAFN), and dynamic connectivity-based grouping. It has been reported that DCG provides the highest data accessibility, while SAF incurs the lowest traffic, of the three techniques. Although DCG performs best in terms of data accessibility, it causes the worst network traffic. Moreover, DCG does not consider selfish nodes in a MANET. The work proposes data replication techniques that address both query delay and data accessibility in a MANET. The work demonstrates such a trade-off and proposes techniques to balance it. The work introduces the cooperative caching-based data access methods, including CachePath, CacheData, and Hybrid. Differing from all the above-mentioned replica allocation or caching techniques, they consider selfish nodes in a MANET.

6 FUTURE WORK and CONCLUSION

The SPRT can be thought of as one dimensional random walk with lower and upper limits. Before the random walk starts, null and alternate hypothesis are defined in such a way that the null hypothesis is associated with the lower limit while the alternate one is associated with the upper limit. A random walk starts from a point between two limits and move toward the lower or upper limit in accordance with each observation. If the walk reaches (or exceeds) the lower or upper limit, it terminates and the null or alternate hypothesis is selected, respectively. They believe that the SPRT is well suited for tackling the mobile replica detection problem since we can construct a random walk with two limits in such a way that each walk determined by the observed speed of a mobile node. The lower and upper limits can be configured to be associated with speeds less than and in excess of Vmax, respectively the SPRT to the mobile replica detection problem as follows.

Once the base station decides that a mobile node has been replicated it revokes the replica nodes from the network:

- A novel mobile replica detection scheme based on the Sequential Probability Ratio Test (SPRT).
- The fact that an uncompromised mobile node should never move at speeds in excess of the System-configured maximum speed
- A benign mobile sensor node's measured speed will nearly always be less than the system-configured maximum speed as long as we employ a speed measurement system with a low error rate.
- Replica nodes are in two or more places at the same time. This makes it appear as if the replicated node is moving much faster than any of the benign nodes, and thus the replica nodes' measured speeds will often be over the system-configured maximum speed.
- If they observe that a mobile node's measured speed is over the system-configured maximum speed, it is then highly likely that at least two nodes with the same identity are present in the network.

REFERENCES

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

[2] L. Anderegg and S. Eidenbenz, "Ad Hoc-VCG: A Truthful and Cost-Efficient Routing Protocol for Mo-

bile Ad Hoc Networks with Selfish Agents," Proc. ACM MobiCom, pp. 245-259, 2003.

[3] K. Balakrishnan, J. Deng, and P.K. Varshney, "TWOACK: Preventing Selfishness in Mobile Ad Hoc Networks," Proc. IEEE Wireless Comm. and Networking, pp. 2137-2142, 2005.

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

[5] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," Proc. ACM MobiCom, pp. 85-97, 1998.

[6] G. Cao, L. Yin, and C.R. Das, "Cooperative Cache-Based Data Access in Ad Hoc Networks," Computer, vol. 37, no. 2, pp. 32-39, Feb. 2004.

[7] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C.H. Papadimitriou, and J. Kubiatowicz, "Selfish Caching in Distributed Systems: A Game-Theoretic Analysis," Proc. ACM Symp. Principles of Distributed Computing, pp. 21-30, 2004.

[8] E. Damiani, S.D.C. di Vimercati, S. Paraboschi, and P. Samarati, "Managing and Sharing Servents' Reputations in P2P Systems," IEEE Trans. Knowledge and Data Eng., vol. 15, no. 4, pp. 840854, July/Aug. 2003.

[9] G. Ding and B. Bhargava, "Peer-to-Peer File-Sharing over Mobile Ad Hoc Networks," Proc. IEEE Ann. Conf. Pervasive Computing and Comm. Workshops, pp. 104-108, 2004.

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

[11] D. Hales, "From Selfish Nodes to Cooperative Net works - Emergent Link-Based Incentives in Peer-to-Peer Networks," Proc. IEEE Int'l Conf. Peer-to-Peer Computing, pp. 151-158, 2004.

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

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

[14] T. Hara and S.K. Madria, "Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks," IEEE Trans. Mobile Computing, vol. 8, no. 7, pp. 950-967, July 2009.

[15] S.U. Khan and I. Ahmad, "A Pure Nash Equilibrium-Based Game Theoretical Method for Data Replication across Multiple Servers," IEEE Trans. Knowledge and Data Eng., vol. 21, no. 4, pp. 537553, Apr. 2009.

[16] N. Laoutaris, G. Smaragdakis, A. Bestavros, I. Matta, and I. Stavrakakis, "Distributed Selfish Caching," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 10, pp. 1361-1376, Oct. 2007.

[17] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed Selfish RepLication," IEEE Trans. Parallel and Distributed Systems, vol. 17, no. 12, pp. 1401-1413, Dec. 2006.

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

[19] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Efficient Peer-toPeer Information Sharing over Mobile Ad Hoc Networks," Proc. World Wide Web (WWW) Workshop Emerging Applications for Wireless and Mobile Access, pp. 2-6, 2004.

[20] Y. Liu and Y. Yang, "Reputation Propagation and Agreement in Mobile Ad-Hoc Networks," Proc. IEEE Wireless Comm. and Networking Conf., pp. 1510-1515, 2003.

[21] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad hoc Networks," Proc. ACM MobiCom, pp. 255-265, 2000.

[22] L.J. Mester, "What's the Point of Credit Scoring?" Business Rev., pp. 3-16, Sept. 1997.

[23] P. Michiardi and R. Molva, "Simulation-Based Analysis of Security Exposures in Mobile Ad Hoc Networks," Proc. European Wireless Conf., pp. 1-6, 2002.

[24] H. Miranda and L. Rodrigues, “Friends and Foes: Preventing Selfishness in Open Mobile Ad hoc Networks,” Proc. IEEE Int’l Conf. Distributed Computing Systems Workshops, pp. 440-445, 2003.

[25] 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.

[26] M.J. Osborne, An Introduction to Game Theory. Oxford Univ., 2003.

[27] P. Padmanabhan, L. Gruenwald, A. Vallur, and M. Atiquzzaman, “A Survey of Data Replication Techniques for Mobile Ad Hoc Network Databases,” The Int’l J. Very Large Data Bases, vol. 17, no. 5, pp. 1143-1164, 2008.

[28] K. Paul and D. Westhoff, “Context Aware Detection of Selfish Nodes in DSR Based Ad-Hoc Networks,” Proc. IEEE Global Telecomm. Conf., pp. 178-182, 2002.

[29] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao, “Cooperation in Wireless Ad Hoc Networks,” Proc. IEEE INFOCOM, pp. 808-817, 2003.

[30] W. Wang, X.-Y. Li, and Y. Wang, “Truthful Multicast Routing in Selfish Wireless Networks,” Proc. ACM MobiCom, pp. 245-259, 2004.

[31] L. Yin and G. Cao, “Balancing the Tradeoffs between Data Accessibility and Query Delay in Ad Hoc Networks,” Proc. IEEE Int’l Symp. Reliable Distributed Systems, pp. 289-298, 2004.