INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Replica Allocation Over MANET Based On The Prediction Of Selfish Behavior

Lincy A.V
II Year M.Tech, CSE, K.M.C.T. Engineering College, India
Adarsh T.K
Assistant Professor, CSE, K.M.C.T. Engineering College, India

Abstract:
In MANET data accessibility is reduced by the network partition. To minimize the degradation, most of them assume that all mobile nodes collaborate fully in terms of sharing their memory space. 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. 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 selfish node may not share its own memory space to store replica for the benefit of other nodes. By allocating replicas we can improve the data accessibility, and performance. Among all nodes in network some of them are not ready to co-operate each other i.e. they behave selfishly. This paper discusses one replica allocation method that overcomes the selfishness of mobile nodes with minimum communication cost and time. In this paper we also consider the trade off between energy saved by selfish nodes and services provided by cooperative nodes.

1. Introduction

A mobile ad hoc network (MANET) is an autonomous system of mobile routers (and associated hosts) connected by wireless links. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. This type of network may operate in a standalone fashion, or it may be connected to the larger Internet. Sensor nodes consist of sensing, data processing, and communication components and typically form ad hoc networks. Fixed infrastructure is absent, and each node acts as a router, transfer data packets for other nodes.

MANET is used where infrastructure is not possible. Each node/device in MANET moves randomly. Due to this movements network partitions occur and will the performance. Network partitions will reduce the data accessibility also. Each node contains its original data and replicas of other nodes. There are number of replica allocation techniques are present and these methods will improve the data accessibility.

In this paper an efficient replica allocation method for MANET are proposed. Here at the time of joining, each device/node will get the details of all the selfish nodes existing in the system which is sent by the data allocator present in the network. Data allocator in the SCF tree [1] will send a table including behaviour of each node. Also each node can predict the movements of other nodes so that every node can allocate necessary replicas to improve data accessibility and reduce query response time. Data updates are also possible here. This paper also implements smooth trade off between energy saved by selfish nodes and services provided by the cooperative nodes.

The nodes in MANET have number of constraints like memory, energy etc. Due to these constraints many of the nodes behave selfishly. MANET contains number of issues also. The issues include node mobility, network partitioning, energy consumption etc

2.Related Work

In MANET number of data replication methods is present. And it includes SAF, DCG, and DAFN, SCF. In the Static Access Frequency Method [2], frequency table is present. Each node has some access frequency to each data items. Based on this access frequency the replicas are allocated. In this case same replicas are there for different nodes

The DAFN [2,3] method eliminates the same replicas present in the neighboring mobile hosts. First, this method allocates replicas in the same way as the SAF method. If there is duplication of data item between two neighboring mobile hosts, then the mobile host with lower access frequency to the data item changes the replica to another replica. Since the neighboring status changes as mobile hosts move, the DAFN method is executed at every relocation period.

At a relocation period, a mobile host may not connect to another mobile host which has an original or a replica of a data item that the host should allocate. In this case, the memory space for the replica is temporary filled with one of replicas that have been allocated since the previous relocation period but are not currently selected for allocation.

The DCG method [2, 3] shares replicas in larger groups of mobile hosts than the DAFN method that shares replicas among neighboring hosts. In order to share replicas effectively, each group should be stable, i.e., the group is not easily divided due to changes of network topology. From this viewpoint, the DCG method creates groups of mobile hosts that are biconnecteds

components [4] in a network. Here, a biconnected component denotes a maximum partial graph which is connected (not divided) if an arbitrary node in the graph is deleted.
SCF-tree [1] based replica allocation technique is to achieve good data accessibility with low communication cost in the presence of selfish nodes. Since our replica allocation technique appropriately handles the (partially) selfish nodes, the technique is expected to achieve the objective. 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.
The SCF tree has to be generated for nodes. SCF tree contains the path taken by the nodes. The selfish node and partially selfish nodes are separated by using the colours. Particular colour can indicate the selfish node and another particular can indicate the partially selfish node. The flow speed of the nodes can be limited to slow, fast and delay. The node processing from source to destination can be visualized in the network.

2.1.System Model

- Set of all mobile nodes in the system represented by Mi where i = {1....n}.
- Set of all data item is represented by Dj where j= {1...n}.
- Each mobile host has memory space for storing their original and replica data items.

On the basis of replica allocation three states are present for a node and are

- Selfish node
- Partially selfish
- Normal node

3.Replica Allocation Over MANET Based On The Prediction Of Selfish Behavior

3.1.Overview
Our strategy consists of four parts:

1) Detecting selfish nodes,
2) Building the SCF-tree,
3) Allocating replica, and
4) Check trade off between nodes individual welfare versus global welfare.

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.
- Using orbitery concept determine the locations of mobile nodes.

The CR score is updated accordingly during the query processing phase. The credit risk from economics is used to effectively measure the degree of selfishness. In economics, credit risk is used to measure risk of loss due to a debtor's non payment 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 he/she is creditworthy. We take a similar approach. If a node wants to know whether another node is believable or not, means that a replica can be paid back, or served upon request to share a memory space in a MANET. With the measured degree of selfishness, we propose a novel tree that represents relationships among nodes in a MANET, for replica allocation, termed the SCF-tree. The SCF-tree is based on human friendship management in the real world.
The SCF tree based replica allocation techniques will minimize the communication cost by maintaining high data accessibility. It happens because here each node identifies which among them are selfish nodes at its own discretion, without forming groups with other nodes.

3.2.Detecting Selfish Node
Here selfishness is detected by using credit risk .The credit risk can be determined by the following equation:

$$\text{Credit Risk} = \frac{\text{Expected Risk}}{\text{Expected Value}} \dots\dots\dots\dots\dots\dots\dots (1)$$

Each node calculates a CR score for its connected nodes. Based on these score each node can estimates the degree of selfishness. We first describe selfish features that may lead to the selfish replica allocation problem to determine both expected value and expected risk.
Selfish features can be classified into two categories: node specific and query processing-specific. Node-specific features can be defined by considering the following case: A selfish node may share part of its own memory space, or a small number of data items. In this case, the degree of selfishness is determined by the size of shared memory space and/or the number of shared data items. In our case, the node Nk's shared memory spaceis represented by SSki, and the number of shared data items present in Nk is denoted by NDki , and this is observed by a node Ni .This is used as node-specific features. SSki and NDki are estimated values, since Nk, may be selfish or not, does not necessarily let Ni 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 Ni observes that node Nk shares large SSki and NDki , node Nk may be treated as a valuable node by node Ni.
As the query processing-specific feature, we utilize the ratio of selfishness alarm of Nk on Ni, denoted as Pki, which is the ratio of Ni's data request being not served by the expected node Nk due to Nk's selfishness in its memory space (i.e., no target data item in its memory space).Thus, the query processing-specific feature can represent the expected risk of a node. For instance, when Pki

gets larger, node Ni will treat Nk as a risky node because a large Pki means that Nk cannot serve Ni's requests due to selfishness in its memory usage. To effectively identify the expected node (s), Ni should know the (expected) status of other nodes memory space.

The SCF-tree-based replica allocation techniques, fortunately, support this assumption. This will be explained in the following section. Using the described features, we can modify (1) into (2):

$$\text{CRki} = \frac{\text{Pki}}{\propto \times \text{SSki} + (1-\propto) \times \text{NDki}} \quad \text{where } 0 \leq \alpha \leq 1 \text{----------(2)}$$

The system parameter, α, is used to adjust the relative importance of SSki and NDki. Node Ni updates CRki at every query processing and looks it up for the connected node Nk at every relocation period. In addition, each node also has its own threshold δ of CRki. If the measured CRki exceeds δ,node Nk will be detected as a selfish node by Ni. The value of c (as well as SSki and NDki is updated at every query processing of some item that Ni allocates to other node(s) during the replica allocation phase.

The effect of parameters SSki and NDki on CRki can be weighted by taking into consideration the size of memory space at node Ni, Si, and the total number of data items accessed by Ni, ni. The rationale is that CRki may be strongly affected by Si and ni if CRki is not normalized. By normalizing, we obtain (3), where nCRki stands for the normalized CRki.

$$\text{nCRki} = \frac{\text{Pki}}{\propto * \frac{\text{SSki}}{\text{Si}} + (1-\alpha) * \text{NDki/ni}} \text{ where } 0 \leq \propto \leq 1 \ldots\ldots\ldots\ldots (3)$$

Algorithm 1 describes how to detect selfish nodes. Node Ni detects selfish nodes at each relocation period based on the value nCRki. Each nodes system parameter is considered as Pki's initial value.The initial value of Pki can represent the basic attitude toward strangers. For instance, if the initial value equals zero, node Ni always treats a new node as a non selfish node. Therefore, Ni can cooperate with strangers easily for cooperative replica sharing.

Algorithm 1.pseudo code to detect selfish nodes

```
At every relocation period
 detect (){
      for (each connected node Nk) {
        if(nCRki < δ)  Nk is marked as non-selfish
         else Nk is marked as selfish;}
        wait until replica allocation is done;
        for(each connected node Nk){
       if(Ni has allocated replica to Nk){
           NDki=the number of allocated replica;
           SSki=the total size of allocated replica;}
          else{NDki=1;
          SSki=size of data items;
      }}}
```

As described in Algorithm 2, Ni maintains its NDki , SSki ,and Pki during each query processing phase.

Algorithm 2.Pseudocode to update selfish features

```
Update(){
      while (during the predefined time Δ){
     If(an expected node Nk serves the query)
     Decrease Pki;
      If (an unexpected node Nj serves the query){
       NDij=NDij+1;
       SSij=SSij+(sizeof a data item);
         }}
       if(an expected node Nk does not serve the query){
       increase Pki;
       NDki=NDki-1;
     SSki=SSki-(size of data item);
     }}
```

When Ni issues a query, Ni awaits the response from the expected node Nk during the predefined wait time, Δ where Δ is the expected maximum time taken to exchange one round of request response message across the entire network.

Whenever Ni detects the selfish behavior of Nk, it modifies Pki , NDki, and SSki accordingly. If Nk serves the query as expected, however, only Pki will be decreased, while NDki and SSki remain unchanged. Note that, in case an unexpected node Nj replies to Ni's request, Ni will modify NDij and SSij accordingly, while not affecting s. That is, the reply from unexpected nodes does not affect the selfish features of expected nodes.

3.3.Building Scf-Tree

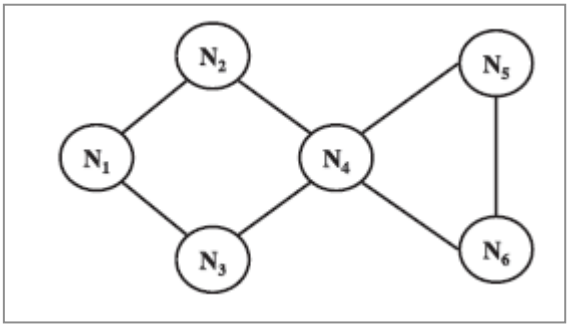

Figure 1(a): Sample Topology G

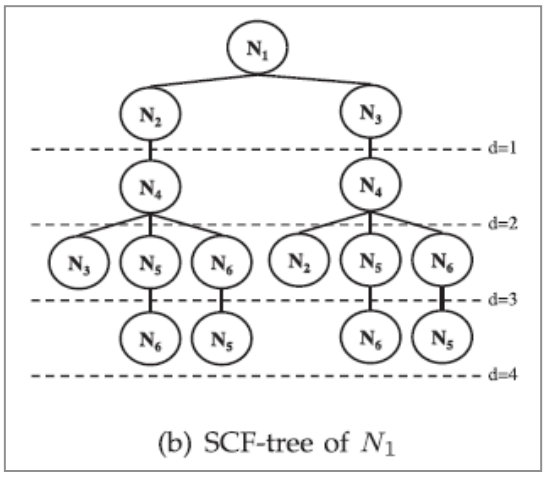

Figure 1(b): SCF-Tree Of N1

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.

Prior to building the SCF-tree, each node makes its own partial topology graph

Gi=(INi,ILi) which is a component of the graph G. Gi consists of a finite set of the nodes connected to Ni and a finite set of the links. At every relocation period, each node updates its own SCF-tree based on the network topology of that moment.

Algorithm 3.Pseudo code to build SCF tree

```
Constructscftree(){
   append Ni to SCF-tree as the root node;
   checkChildnodes(Ni);
   return SCF-tree;}
  Procedure checkChildnodes(Nj){
    for (each node Na Є INja ){
    if(distance between Na and the root>d)
   continue;
   else if(Na is an ancestor of Nj in Tiscf);
   continue;
   else{append Na to Tiscf as a child of Nj;
   checkChildnodes(Na);
  }}}
```

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.

Since we assume that a node can use some portion of its memory space selfishly, we may divide memory space Mi for replica logically into two parts: selfish area Ms and public area Mp. Each node may use its own memory space Mi freely as Ms and/or

Mp. In each node, Ms will be used for data of local interest (i.e., to reduce query delay), while Mp for public data is asked to hold data by other node(s) (i.e., to improve data accessibility).
Algorithm 4 describes how to allocate replica, where IDi and Li denote an ordered set of all data items to be allocated by Ni and the list of node ids, respectively. Note that, IDi is sorted in descending order of Ni's access frequency. Consequently, each node allocates replicas in descending order of its own access frequency.

Algorithm 4.Pseudo code for replica allocation

```
replica_alloc( ){
    Li=make_priority(Tiscf);
    for(each data item Є IDi){
     if(Ms is not full)
     allocate replica of the data to Ms;
     else{
     allocate replica of the data to the target node;
     if(Mp is not full)
     allocate replica of the data to Mp;}}
     while(during a relocation period){
      if(Nk requests for the allocation of Dq)
      replica_alloc_forothers(Nk,Dq);}}
Procedure make_priority(Tiscf){
      for(all vertices in Tiscf){
     select a vertex in Tiscf in order of BFS;
     append the selected vertex id to Li;}
     return Li;}
Procedure replica_alloc_forothers(Nk,Dq){
    if(Nk is in Tiscf and Ni does not hold Dq){
    if(Mp is not full) allocate Dq to Mp;
    else{if(Ni holds any replica of local interest in Mp)
    replace the replica with Dq;
    else{
     if(nCRih>nCRik)
     replace the replica requested by Nh with Dq;
}}}}
```

During a relocation period, Ni may receive requests for replica allocation from any nodes within its SCF-tree. If Ni is not a fully selfish node, Ni shall maintain its memory space Mp for the requests from other nodes, say Nk. In this case, Ni should determine whether to accept the replica allocation request.

3.5.Using Orbitery Concept Determine Location Of Mobile Nodes

Smooth trade algorithm is the process for finding the location of the node. This smooth trade technique is based on the orbitery concept. That is this smooth trade first set the orbit region for the center of the node travelling space and set the continuous square shaped orbits based on the center of the region.

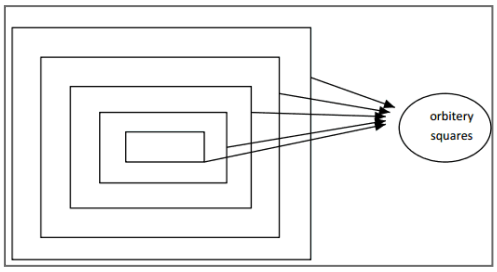

Figure : 2

For example if we identify that the N4 is the selfish node using the handling selfishness algorithms after that mobility node change their location simultaneously. So we can't able to identify the location. But by using this orbitery concept we can identify the location of the node. And we can able to identify the nearest neighbour location also.

4.Performance Evaluation

In the simulation, the number of mobile nodes is set to 40.Each node has its local memory space and moves with a velocity from 0 ~ 1 (m/s) over 50 (m) × 50(m) flatland .The movement pattern of nodes follows the random waypoint model, where each node remains stationary for a pause time and then it selects a random destination and moves to the destination. After reaching the destination, it again stops for a pause time and repeats this behaviour .The radio communication range of each node is a circle with a radius of 1~ 19 (m). We suppose that there are 40 individual pieces of data, each of the same size. In the network, node Ni (1 ~ i ~ 40) holds data Di as the original. The data access frequency is assumed to follow Zipf 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.

The default number of selfish nodes is set to be 70 percent of the entire nodes in our simulation, based on the observation of a real application [1]. We set 75 percent of selfish nodes to be partially selfish and the remaining to be fully selfish. Partially selfish nodes consist of three groups of equal size. Each group uses 25, 50,and 75 percent of its memory space for the selfish area.
We evaluate our strategy using the following four performance metrics:

- Overall selfishness alarm: This is the ratio of the overall selfishness alarm of all nodes to all queries that should be served by the expected node in the entire system.
- 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.

4.1.Evaluation Results

4.1.1.Communication cost

Communication cost decreases in every method. Compare to SCF tree based replica allocation method our method communication cost is decreased very much.
Since the orbitery concept determines position of mobile nodes and neighbouring nodes communication between different nodes for identifying the location can be avoided.

4.1.2.Nodes Versus Energy consumption

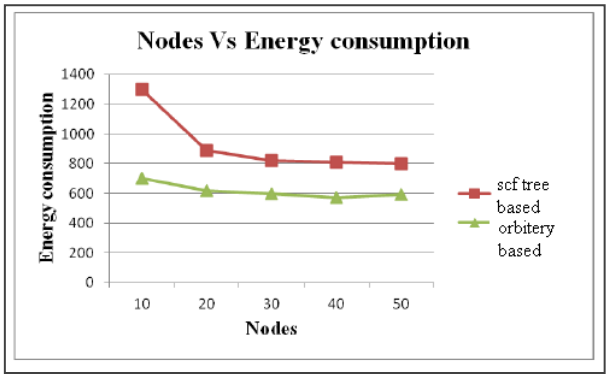

Figure : 3

4.1.3.Mobility Versus Delay

The following Figure shows the results of Mobility Vs Delay. From the results, we can see that orbitery based scheme has slightly lower delay than the SCF tree based concept.

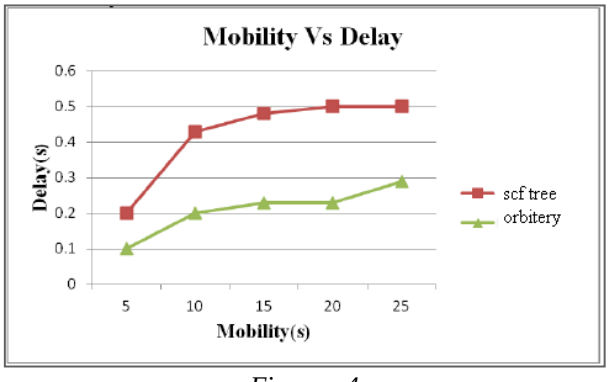

Figure : 4

5.Conclusion

In this paper, we had deal with the problem of selfish replica allocation. 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. 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. In this paper we consider trade off between nodes individual welfare and global welfare. The proposed replica allocation technique used for data accessibility, communication cost and query delay.

6.Future Work

In future we will plan to identify and handle false alarms in selfish replica allocation. In future we plan to find the selfish node and it can be remove from the network for save some space of memory to allocating replica.

7.References

1. Handling Selfishness in Replica Allocation over a Mobile Ad Hoc Network, IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 11, NO. 2, FEBRUARY 2012, Jae-Ho Choi, Kyu-Sun Shim, SangKeun Lee, and Kun-Lung Wu, Fellow, IEEE
2. E. Adar and B.A. Huberman, "Free Riding on Gnutella," First Monday, vol. 5, no. 10, pp. 1-22, 2000.
3. L.Anderegg and S. Eidenbenz, "Ad Hoc-VCG: A Truthful and Cost-Efficient Routing Protocol for Mobile Ad Hoc Networks with Selfish Agents," Proc. ACM MobiCom, pp. 245-259, 2003.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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. 840-854,July/Aug. 2003.
10. 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.
11. T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," Proc. IEEE INFOCOM, pp. 1568-1576, 2001
12. 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.
13. Efficient Data Replication Algorithm for Mobile Ad-Hoc 2011 International Conference on Information and Network Technology IACSIT Press, Singapore
14. EENMDRA: Efficient Energy and Node Mobility based Data Replication Algorithm for MANET: IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, May 2012 ISSN (Online): 1694-0814.
15. A Combined Credit Risk and Collaborative Watchdog Method for Detecting Selfish Node over Mobile Ad-Hoc Network ,International Journal of Advanced Research in Computer Science and Software Engineering .