ISSN:0975-9646

A Survey on Finding Selfish Nodes in Mobile Ad Hoc Networks

J.Vijithanand, K.Sreerama Murthy

Department of Information Technology
Sreenidhi Institute of science & Technology
Hyderabad-India

***Abstract*-- Securing the mobile ad hoc networks (MANETs) in an untrustworthy open environment is always a challenging problem. In recent years, mobile ad hoc networks have become a very popular research topic. MANETs are attractive technology for many applications such as rescue operations, tactical operations, environmental monitoring, conferences, and the like. However, performing network functions consumes energy and other resources. To save its energy a node may behave selfishly and uses the forwarding service of other nodes without correctly can severely degrade the performance at the routing layer. Specifically, nodes may participate in the route discovery and maintenance process but refuse to forward data packets. In this survey various methods for detecting selfish nodes are discussed with their key advantages. Moreover one of the most important aspects is to propose specific behavior pattern creation that would let to evaluate neighbor behavior; I surveyed the key algorithms for constructing behavior pattern for the neighboring nodes in MANETs. In the literature there are many methods which deal with the selfish behaviour of the nodes. This paper compares different methods available for reducing the effect of selfish nodes in mobile ad hoc networks.**

***Keywords:* Mobile Ad Hoc Networks, Routing misbehavior, Selfishness, Network security.**

I. INTRODUCTION

Mobile ad-hoc networks (MANETs) allow for wireless devices to form a network without the need for central infrastructure [1]. While the lack of need for infrastructure allows the network to be very flexible, it also makes routing a critical concern in the network. The *data collection* component is responsible for collection and pre-processing data tasks: transferring data to a common format, data storage and sending data to the detection module. In ad-hoc wireless networks each computer with a wireless interface can communicate directly with participating nodes. These nodes can self-organize without central management and special infrastructure[2][3]. The network is established using (limited range) radio communication where each node acts as both data terminal and data transfer equipment. Moreover, nodes can move freely resulting in changes to the network topology and updated routing in order to forward the packets. The topology change depends on different factors such as mobility model, node speed etc. Due to the infrastructure less nature of MANETs packets sent between distant nodes are expected to be relayed by intermediate ones [3], which act as routers and provide the forwarding service. The forwarding service is closely related to the routing. It consists in correctly relaying the received packets from node to node until reaching their final destination, following routes selected and maintained by the routing protocol [3]. These services (routing and data forwarding) together are at the core of the network layer. The nature of MANET makes cooperation among nodes essential for the system to be operational. In some MANET's applications where all nodes belong to a single authority (in the application layer point of view) and have a common goal, e.g.-soldiers in a military unit during a battlefield or rescuers in a rescue team during a rescue operation, nodes are cooperative by nature[2][3]. However, in many civilian applications, such as networks of cars and provision of communication facilities in remote areas, nodes typically do not belong to a single authority and do not pursue a common goal. In such networks, forwarding packets for other nodes is not in the direct interest of anyone, so there is no good reason to trust nodes and assume that they always cooperate. In MANETs critical functions like routing and forwarding performed by less trusted and less secured nodes. Indeed, nodes try to preserve their resources, and particularly their batteries[4]. An individual mobile node may attempt to benefit from other nodes, but refuse to share its own resources. Such nodes are called selfish or misbehaving nodes and their behavior is termed selfishness or misbehavior. Intentionally uncooperative behavior (misbehavior) may result in a total communication breakdown. A node may behave selfishly by agreeing to forward the packets and then failing to do so due to Overloaded, Selfish, Malicious or Broken. Behavior node models Collaborative model: A node that behaves properly executing both packet forwarding and routing functions. Selfish model: A node that misbehaves to save its battery life. This node could disable packet forwarding and/or routing functions.

II. RELATED WORK

A. Credit Based Methods

Credit based methods are also called as incentive based methods. In these methods selfish nodes are not punished instead unselfish nodes are rewarded for helping other nodes. This stimulates the cooperation of nodes in the network. This section discusses some of the credit based systems in the literature.

B. Secure Incentive Protocol

This approach assumes that each mobile node (MN) has a tamper-proof security module such as SIM cards in GSM networks, which deals with security related functions and each intermediate node (IN) puts non-forged stamps on the forwarded packets as a proof of forwarding[2]. Secure Incentive Protocol, (SIP) uses "credits" as the incentives to stimulate packet forwarding. For this purpose, each smartcard has a credit counter (CC) which is pre-charged with a certain amount of credits before shipped out[2][3]. The charging and rewarding on a node is done by decreasing or increasing the CC in that node and the CC will retain its value even when the MN is power off. When the MN is power-on again, it could still reuse the credits in the CC even in another SIP-enabled ad hoc network. To guarantee the security of SIP, each smartcard contains a private number and a public number (keys). The nodes have no knowledge about the keys stored in the smartcard and could not change CC in an unauthorized way either. SIP is session-based and mainly consists of three phases. During the first *Session initialization* phase, a session initiator (SI) negotiates session keys and other information with a session responder (SR) and INs between them. And each IN puts a non-forged stamp on each data packet forwarded and SI/SR collect those stamps for later rewarding use in the next *Data forwarding* phase[2]. The final phase is R*ewarding* phase, in which each IN is awarded a certain number of credits based on the number of forwarded packets. Advantages of this method are 1. SIP is routing- independent in the sense that it could coexist with any on- demand unicast routing protocol such as DSR and AODV. 2. SIP is session based rather than packet based. 3. Security module is tamper proof and hence unauthorized access is not allowed. But the problem with this approach is, it needs every node to possess the hardware module and SIP is implemented in the hardware module. Hardware module will not be available in the already existing mobile nodes.

C. Sprite

The basic idea of their scheme is as follows: a Credit Clearance Service (CCS) is introduced to determine the charge and credit to each node involved in the transmission of a message [5]. When a node receives a message, the node keeps a receipt of the message and later reports it to the CCS when the node has a fast connection with the CCS. Payments and charges are determined from a game theory perspective. The sender instead of the destination is charged in order to prevent denial-of-service attack in the destination by sending it a large amount of traffic [5][6]. Any node who has ever tried to forwarding a message is compensated, but the credit a node receives depends on whether or not its forwarding action is successful – forwarding is considered successful if and only if the next node on the path reports a valid receipt to the CCS.

Three selfish actions and the corresponding countermeasures are discussed in the paper:

1. After receiving a message, a selfish node may save a receipt but does not forward the message. To prevent this, the CCS should give more credit to a node who forwards a message than to a node that does not forward a message to motivate a selfish node to forward others' message. To achieve this objective, if the destination does not submit a receipt, the CCS first determines the last node on the path that has ever received the message. Then the CCS pays this last node less than it pays each of the predecessors of the last node [5].

2. A node received a message may not report the receipt. This is possible if the sender colludes with the

intermediate nodes, so that the sender can pay the node a behind-the-scene compensation, which is little bit more than the CCS will pay, and the sender still get a net gain.

In order to prevent this cheating action, the CCS charges the sender an extra amount of credit if the destination does not report the receipt so that colluding group get no benefit. .

3. Since reporting a receipt to the CCS is sufficient for getting credit, a group of colluding nodes may forward only the receipt of a message, instead of forwarding the whole message, to its successor.

Two cases are considered: 1) the destination colludes with the intermediate nodes; 2) the destination does not collude with the intermediate nodes. In the first case, since the message is for the destination and if the destination really submits the receipt, then the intermediate nodes and the destination should be paid as if no cheating had happened. In the second case, if the destination does not report a receipt of a message, the credit paid to each node should be multiply by a fraction, r, where r<1.

Modeling the submissions of receipts regarding a given message as a one-round game, the authors proved the correctness of the receipt-submission system using game theory. Although the main purpose of the system is for message-forwarding in unicast, it can be extended to route discovery and multicast as well. This scheme, however, may have several issues:

1. Receipts of each node along a path maybe submitted to the CCS at different times, making it difficult for the CCS to determine the actual payment to each node [5].

2. The scheme[6] is based on DSR, which includes the path in the forwarding message. A malicious node not on the path can collude with nodes on the path to forge a receipt and spoof the CCS.

III. Identifying and Isolating Selfish Nodes

This section explains methods that are used for punishing the selfish nodes. Selfish nodes are identified and isolated from the network. They are stopped from using the network services. Most of the approaches in the literature are following punishing system rather than rewarding system.

A. *Watch Dog and Path Ratter*

When a node forwards a packet, the node's *watchdog* verifies that the next node in the path also forwards the packet [6]. The *watchdog* does this by listening promiscuously to the next node's transmissions. If the next node does not forward the packet, then it is considered as misbehaving. The *path rater* uses this knowledge of misbehaving nodes to choose the network path that is most likely to deliver packets. The nodes rely on their own *watchdog* exclusively and do not exchange reputation information with others. F Fig 1 illustrates how the *watchdog* works. Suppose there exists a path from node S to D through intermediate nodes A, B, and C. Node A cannot transmit all the way to node C, but it can listen on node B's traffic [6]. Thus, when A transmits a packet for B to forward to C, A can often tell if B transmits the packet. If encryption is not performed separately for each link, which can be expensive, then A can also tell if B has tampered with the payload or the header.

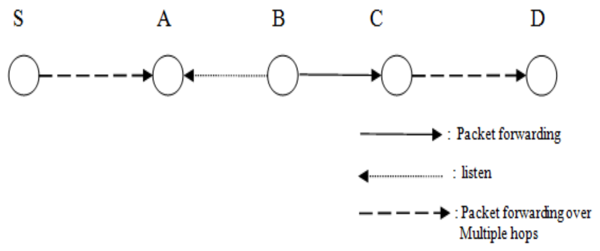

Figure 1: *Watchdog* technology

When B forwards a packet from S toward D through C, A can overhear B's transmission and can verify that B has attempted to pass the packet to C. The solid line represents the intended direction of the packet sent by B to C, while the dashed line indicates that A is within transmission range of B and can overhear the packet transfer. The *watchdog* is implemented by maintaining a buffer of recently sent packets and comparing each overheard packet with the packet in the buffer to see if there is a match. If so, the packet in the buffer is removed and forgotten by the *watchdog*, since it has been forwarded on. If a packet has remained in the buffer for longer than a certain timeout, the *watchdog* increments a failure tally for the node responsible for forwarding on the packet. If the tally exceeds a certain threshold bandwidth, it determines that the node is misbehaving and sends a message to the source notifying it of the misbehaving node. The *path rater*, run by each node in the network, combines knowledge of misbehaving nodes with link reliability data to pick the route most likely to be reliable. Each node maintains a rating for every other node it knows about in the network. It calculates a path metric by averaging the node ratings in the path. If there are multiple paths to the same destination, the path with the highest metric will be chosen. Nodes suspected of misbehaving by the *watchdog* mechanism are assigned a special highly negative value. When the *path rater* calculates the path metric, negative path values indicate the existence of one or more suspected misbehaving nodes in the path. If a node were marked as misbehaving due to a temporary malfunction or incorrect accusation it would be preferable if it were not permanently excluded from routing. Therefore nodes that have negative ratings should have their ratings slowly increased or set back to a non-negative value after a long timeout. In *watchdog* and *path rater* mechanism, wireless interfaces that support promiscuous mode operation are assumed, which is not appropriate for all mobile ad hoc network scenarios. Also, the *watchdog* technique has the weaknesses that it might not detect a misbehaving node in the presence of:

1. Ambiguous collision. As in the above example, an ambiguous collusion is the scenario that packet collusion occurs at A while it is listening for B to forward on a packet.

2. Receiver collisions. In the example, A can only tell whether B sends the packet to C, but it cannot tell if C receives it.

3. Limited transmission power, in which signal is strong enough to be overheard by the previous node but too weak to be received by the true recipient.

4. False misbehavior, in which nodes falsely report other nodes as misbehavior.

5. Collusion, where multiple nodes in collusion can mount a more sophisticated attack. For example, B forwards a packet to C but do not report to A when C drops the packet.

6. Partial dropping, in which a node can circumvent the *watchdog* by dropping packets at a lower rate than the *watchdog*'s configured minimum misbehavior threshold.

B. CONFIDANT

CONFIDANT stands for Cooperation of Nodes Fairness in Dynamic Ad-hoc Network, it works as an extension to on demand routing protocols [8]. CONFIDANT is based on selective altruism and utilitarianism. It aims at detecting and isolating misbehaving nodes, thus making it unattractive to deny cooperation. Nodes monitor their neighbors and change the reputation accordingly. *Reputation* is used to evaluate routing and forwarding behavior according to the network protocol. *Trust* is used to evaluate participation in the CONFIDANT meta-protocol. Trust relationships and routing decisions are based on experienced, observed, or reported routing and forwarding behavior of other nodes. CONFIDANT consists of the following components: *The Monitor, the Trust Manager, the Reputation System and the Path Manager. The monitor* is the equivalent of a "neighbor watch", where nodes locally look for deviating nodes. The node can detect deviation by the next node on the source route by either listen to the transmission of the next node or by observation of route protocol behavior [8]. *The trust manager* deals with incoming and outgoing ALARM messages. ALARM messages are sent by the trust manager of a node to warn others of malicious nodes. Outgoing ALARM messages are generated by the node itself after having experienced, observed, or received a report of malicious behavior [8]. The recipients of these ALARM messages are so-called friends, which are administered in a friends list. Incoming ALARM messages originate from either outside friends or other nodes, so the source of an ALARM has to be checked for trustworthiness before triggering a reaction. *The reputation system* in this protocol manages a table consisting of entries for nodes and their rating. The rating is changed only when there is sufficient evidence of malicious behavior that is significant for a node and that has occurred a number of times exceeding a threshold to rule out coincidences. To avoid a centralized rating, local rating lists and/or black lists are maintained at each node and potentially exchanged with friends. *The path manager* performs the following functions: path re-ranking according to reputation of the nodes in the path; deletion of paths containing malicious nodes, action on receiving a request for a route from a malicious node (e.g. ignore, do not send any reply) and action on receiving request for a route containing a malicious node in the source route (e.g. ignore, alter the source).

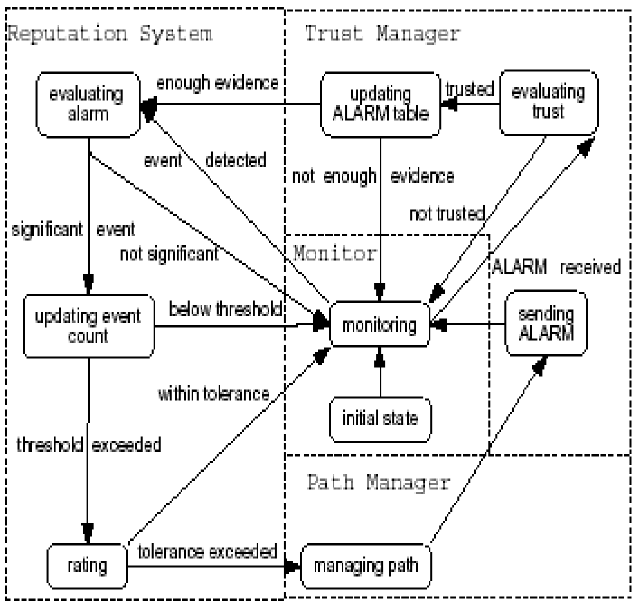

Fig 2: Trust architecture and finite state machine within each node.

As shown in Fig 2, each node monitors the behavior of its neighbors. If a suspicious event is detected, the information is given to *the reputation system*. If the event is significant for the node, it is checked whether the event has occurred more often than a predefined threshold that is high enough to distinguish deliberate malicious behavior from simple coincidences such as collisions. What constitutes the significance rating can be defined for different types of nodes according to their security requirements. If that occurrence threshold is exceeded, the reputation system updates the rating of the node that caused that event. If the rating turns out to be intolerable, the information is relayed to the path manager, which proceeds to delete all routes containing the misbehaving node from the path cache.

Although CONFIDANT can detect and isolate misbehaving nodes, it has some limitations:

1. It is a detection-based reputation system.
2. Events have to be observable and classified for detection.
3. Reputation can only be meaningful if the identity of each node is persistent; otherwise it is vulnerable to spoofing attack.

C. CORE

CORE (COllaborative REputation mechanism) is a generic mechanism that can be integrated with any network function like packet forwarding, route discovery, network management and location management [7]. CORE stimulates node cooperation by a collaborative monitoring technique and a reputation mechanism. In this mechanism, reputation is a measure of someone's contribution to network operations. Members that have a good reputation can use the resources while members with a bad reputation, because they refused to cooperate, are gradually excluded from the community [7]. Each node computes a reputation value for every neighbor using a sophisticated reputation mechanism that differentiates between *subjective reputation* (observation), *indirect reputation* (positive reports by others) and *functional reputation* (take-specific behavior). There are two basic components for the CORE mechanism: *reputation table* (RT) and *watchdog mechanism* (WD). The *watchdog* mechanism is used to detect misbehavior nodes [7]. The *reputation table* is a data structure stored in each node. Each row of the table consists of four entries: the unique identifier of the entity, a collection of recent *subjective observations* made on that entity's behavior, a list of the recent *indirect reputation* values provided by other entities and the value of the reputation evaluated for a predefined function. The CORE scheme involves two types of protocol entities, a *requestor* and one or more *providers* that are within the wireless transmission range of the requestor. If a *provider* refuses to cooperate (the request is not satisfied), then the CORE scheme will react by decreasing the reputation of the *provider*, leading to its exclusion if the non-cooperative behavior persists [7]. *Route tables* are updated in two different situations: during the request phase of the protocol and during the reply phase corresponding to the result of the execution. In the first case only the *subjective reputation* value is updated while in the second case, only the *indirect reputation* value is updated. To prevent a misbehaving entity to distribute false information about other entities in order to initiate a denial of service attack, the protocol allows only the distribution of positive rating factors. No negative ratings are spread between the nodes, so it is impossible for a node to maliciously decrease another node's reputation [7]. CORE suffers from spoofing attack because misbehaving nodes can change their network identity. The *watchdog* technique, a basic component of CORE, relies on the promiscuous mode operation, which is not always true (e.g. in military applications) and has some weakness. Though CORE successfully prevents false accusation that may decrease nodes' reputation maliciously, it cannot prevent colluding nodes from distribute false praise that may increase malicious nodes' reputation.

D. *Token-based Approach*

Token-based mechanism enforces cooperation in mobile ad hoc networks. In their proposal, each node has to have a *token* in order to participate in the network operations; its local neighbors collaboratively monitor it to detect any misbehavior in routing or packet forwarding services. The *token* is renewed via multiple neighbors after it is expired [9][10]. The period of validity of a node's *token* is dependent on how long it has stayed and behaved well in the network. A well-behaving node accumulates its credit and renews its *token* less and less frequently as time evolves. The solution takes a self-organized approach, where neither existence of any centralized trust entity nor any a priori secret association between nodes is assumed [9][10]. There is only a global secret/public key pair *SK/PK*, where *PK* is well known by every node of the network, and *SK* is shared by all nodes in the network, but each node only knows a limited portion of it. The solution is composed of four components:

- Neighbor verification: verify whether each node is legitimate or malicious.
- Neighbor monitoring: monitor behaviors of each node and detect attacks from malicious ones.
- Intrusion reaction: alert the network and isolate the attackers.
- Security enhanced routing protocol: incorporates the security information into the mobile ad hoc network routing protocol.

The *token* issuing process is decentralized, and the *token* of each node is issued and signed by its *k* neighbors collaboratively. Before the expiration of a node's current *token*, the node broadcasts a TREQ (Token Request) to its neighbors [11]. When a node receives a TREQ from its neighbor, it extracts the *token* from the TREQ packet. If the TREQ is valid and the owner of the TREQ matches the owner of the *token*, it constructs a new *token*, signs the newly constructed *token* using its own share of *SK*, encapsulates the signed *token* in a TREP (Token reply), and unicasts the TREP to the node requesting the *token[12]*. When the node which needs to renew its *token* receives *k* TREP from different neighbors, it can combine these partially signed *token* into a *token* signed by *SK*. The adopted credit based strategy in determining the expiration time of each node's *token*. Each time a legitimate node renews its *token*, the period of validity of its *token* increases by a fixed time interval. The authors also extend the AODV protocol into AODV-S, which is a *security enhanced routing protocol*. Routing security relies on the redundancy of routing information rather than cryptographic techniques [12]. Each AODV-S node maintains the list of all its verified neighbors which possess valid *token*s and only interacts with its verified neighbors. When a node broadcasts a new routing update, it explicitly claims the next hop. Each node also keeps track of the route entries previously announced by its neighbors. This redundancy of the routing information makes it possible for a node to prevent routing updates misbehavior. Packet forwarding misbehaviors, such as packet dropping, packet duplicating and network layer packet jamming, are also detected using an algorithm similar to the *watchdog* technique. Each node overhears the channel at all time and records the headers of the recent packets it has overheard. If a node detects a neighbor's misbehavior, it considers the neighbor as an attacker and broadcast a SID (Single Intrusion Detection) packet. A node is considered as an attacker if and only if *m* nodes out of all *n* neighbors have independently sent out SID packets against it. The selection of *m* represents the tradeoff between the prompt reaction to the attackers and the protection of legitimate modes from false accusation. When a node has received m independent SID packets against the same node, it constructs a notification of *token* revocation, signs the notification using its own share of *SK*, and broadcasts it in a GID (Group Intrusion Detection) packet. Then the first node that receives *k* GID packets against the same node combines them and constructs a TREV (*Token* Revocation), which is signed by the *SK*, based on polynomial secret sharing. The *intrusion reaction* process is triggered only when an attacker is detected. When a node receives a TREV packet and if the *token* is not on the TRL (*Token* Revocation List), it adds the *token* into the TRL. At the same time, each neighbor of an attacker deems the link between it and the attacker as broken and uses the path maintenance mechanism to cancel out these links. *Token*-based mechanism is more suitable in large and dense mobile ad hoc network and where node mobility is low than otherwise because it presents the following drawbacks:

1. Frequent changes in the local subset of the network that shares a key for issuing valid *token*s can cause high

computational overhead, not to mention the high traffic generated by issuing/renewing a *token*.
2. The localized monitoring mechanism executed by each node is intrinsically inaccurate due to the inaccuracy in the information obtained by overhearing the channel.
3. The bootstrap phase to generate a valid *token* for each node has limitation. For example, the node needs to have at least k neighbors, suggesting the use of such mechanism in a rather dense mobile ad hoc network.

IV. A FRAMEWORK FOR DETECTION OF SELFISHNESS

This Paper describes a new framework based on Dempster-Shafer theory-based selfishness detection framework (DST-SDF) with some mathematical back-ground and simulation analysis. The DST-SDF is dedicated for MANETs based on standard routing like dynamic source routing (DSR) [12]. The main concept relies on end-to-end packet acknowledgments in the following way: every time a source node sends a packet to a destination node, it waits for a certain predefined time for an acknowledgement of the packet. If one arrives within the predefined time, the source node has reason to claim that all nodes on the path are cooperative (none is selfish). Otherwise if there are no other indications of faultiness on the path (e.g., RERR messages), the source node knows that there are selfish nodes on the path. Whenever an acknowledgment does or does not arrive in time, a special *recommendation message* is sent out to inform the other nodes about the detected situation (selfish or cooperative behavior on the path, respectively). Every node in the network is equipped with a dedicated component executing a DST-based algorithm that uses received recommendation messages to evaluate the selfishness of each node. The resulting values can be used as routing metrics while selecting packets' routes in the near future

V. CONCLUSION

This paper discussed several approaches for dealing with selfish nodes. Selfish nodes are a real problem for ad hoc networks since they affect the network throughput. Many approaches are available in the literature. But no approach provides a solid solution to the selfish nodes problem. The Credit based approach provides incentives to the well behaving nodes and just by passes the selfish nodes in selecting a route to the destination. But selfish node still enjoys services without cooperating with others. The detection and isolation mechanism isolates the selfish nodes so that they don't receive any services from the network. Thus penalizing the selfish nodes. But what happens if many nodes become selfish? Network communication itself will become impossible. Thus we cannot eliminate all the selfish nodes from the network. A new method to reduce the effect of selfishness and stimulating the nodes to cooperate in the network services should be developed. But the overhead in achieving this should also be less. Because we should remember that after all we are dealing with battery operated devices

REFERENCES

[1] S.Murthy and J.J.Garcia-Lana_Aceves, An Efficient Routing Protocol for Wireless Networks, *ACM Mobile Networks and Applications Journal, Special Issue on Routing in Mobile Communication Networks*, pp. 183-197, October 1996.

[2] Yanchao Zhang , Wenjing Lou , Wei Liu, Yuguang Fang, " A secure incentive protocol for mobile ad hoc networks" in Journal of Wireless Networks , Volume 13 Issue 5, pp. 663-678 , October 2007

[3] L. Buttyan and J.P. Hubaux, "Enforcing service availability in mobile ad-hoc WANs", in Proc. of IEEE/ACM MobiHoc, Boston, Aug. 2000

[4] L. Buttyan and J.P.Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," ACM Journal for Mobile Networks (MONET), Vol. 8, No. 5, Oct. 2003.

[5] S.Zhong, J.Chen, and Y.R.Yang, " Sprite: A Simple, Cheat Proof, Credit based System for Mobile Ad Hoc Networks", in Proceedings of INFOCOM, Apr. 2003

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

[6] Pietro Michiardi and Refik Molva, "CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," Sixth IFIP conference on security communications, and multimedia (CMS 2002), Portoroz, Slovenia, 2002.

[7] Buchegger, Sonja ; Le Boudec, Jean-Yves, " Performance Analysis of CONFIDANT Protocol: Cooperation of Nodes - Fairness in Dynamic Ad- Hoc Networks," in Proceedings of IEEE/ACM Workshop on Mobile Ad

Hoc Networking and Computing (MobiHOC). IEEE, June 2002.

[8] Hongxun Liu, José G. Delgado-Frias, and Sirisha Medidi, "Using a cache scheme to detect selfish nodes in mobile adhoc networks " in proceedings of IEEE international Conference on Networks, pp- 7– 12, Nov. 2007

[9] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in Mobile Computing, T. Imielinski and H. Korth, Eds. Norwell, MA: Kluwer, vol. 353, pp. 153–181, 1996.

[10] Jerzy Konorski and Rafał Orlikowski "A Framework for Detection of Selfishness in Multihop Mobile Ad Hoc Networks in "Journal of telecomm-unications and information technology, pp 34- 40, 2009.

[11] G.Appenzeller, M.Roussopoulous, and M.Baker, "User-friendly access control for public network ports," in Proc. IEEE INFOCOM, pp. 699-707,1999.

[12] H.Miranda and L.Rodrigues, "Preventing Selfishness in Open Mobile Ad Hoc Networks", October 2002. conference on security communications, and multimedia (CMS 2002), Portoroz, Slovenia, 2002