A Hierarchical Account-aided Reputation Management System for Large-Scale MANETs

Ze Li and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University
Clemson, SC 29631
Email: {zel, shenh}@clemson.edu

***Abstract*—Encouraging cooperative and deterring selfish behaviors are important for proper operations of MANETs. For this purpose, most previous efforts either rely on reputation systems or price systems. However, both systems are neither sufficiently effective in providing cooperation incentives nor efficient in resource consumption. Nodes in both systems can be uncooperative while still being considered trustworthy. Also, information exchange between mobile nodes in reputation systems and credit circulation in price systems consume significant resources. This paper presents a hierarchical Account-aided Reputation Management system (ARM) to efficiently and effectively provide cooperation incentives. ARM builds a hierarchical locality-aware DHT infrastructure for efficient and integrated operations of both reputation and price systems. The infrastructure helps to globally collect all reputation information in the system, which helps to calculate more accurate reputation and detect abnormal reputation information. Also, ARM coordinately integrates resource and price systems by enabling higher-reputed nodes to pay less for their received services. Theoretical analysis demonstrates the properties of ARM. Simulation results show that ARM outperforms both a reputation system and price system in terms of effectiveness and efficiency.**

I. Introduction

A mobile ad-hoc network (MANET) is a self-configuring network automatically formed by a collection of mobile nodes without a fixed infrastructure or centralized management. Nowadays, wireless devices such as smart-phones (e.g., PDAs, IPhones and gPhones) increasingly become prevalent in our daily life. The number of wireless Internet users has tripled world-wide in the recent three years, and the number of smart-phone users has reached around 190 million in 2010 [1] and will reach 300 million by 2013 [2]. A MANET is expected to connect thousands or even millions of the mobile nodes for pervasive communication in the future. In a MANET, nodes communicate with each other by routing data through relay nodes in a multi-hop manner. Thus, reliable communication is critical to the proper operations of MANETs.

While many technologies are important to achieving high communication reliability, perhaps one of the most essential challenges to overcome is deterring selfish and encouraging cooperative behaviors. MANETs are particularly vulnerable to selfish behaviors due to the individualized nature of nodes. Each node labors under an energy constraint and selfish nodes tend not to forward data in order to save resources. The presence of only such a few misbehaving nodes can dramatically impede the performance of the entire system [3]. Current main methods to deal with the challenge can be divided into two categories: reputation systems and price systems. However, existing reputation systems and price systems are neither sufficiently efficient nor effective. By insufficient efficiency, we mean that the methods exacerbate the resource-efficiency problem in large-scale MANETs by consuming already stringent resources. By insufficient effectiveness, we mean that the methods lack capability to accurately reflect nodes' behavior and prevent nodes from gaining fraudulent benefits while being considered reputed. The accuracy of node reputation can be adversely affected by false information including falsified, conspiratorial and misreported information.

In most current reputation systems [4]–[14], a node collects locally-generated node feedbacks and aggregates them to yield the global reputation values (R_g) for others based on periodical information exchanges between neighbors. The node whose reputation is below a predefined threshold ($\mathcal{T}$) is considered as selfish and put into a blacklist, otherwise as trustworthy. However, the systems suffer from a number of problems. First, they lack efficient mechanisms to collect and propagate reputation information. Periodical information exchanges, keeping redundant reputations in each node, and broadcasting to query reputations [14] consume significant resources, failing to achieve high scalability. Second, reputation calculation based on partial local information, which may include false information, may result in insufficiently accurate reputation evaluation to truly reflect node behaviors. Third, solely relying on reputation system is not effective enough to thwart uncooperative behaviors. The reputation systems provide equal treatment to trustworthy nodes with $R_g \geq \mathcal{T}$. Thus, a node can be uncooperative for some time while still keeping $R_g \geq \mathcal{T}$.

Price systems [15]–[19] treat message forwarding as a service transaction, and introduce virtual credits for the transactions. In the systems, nodes forward others' messages in order to earn credits for their own message transmission. However, the systems also inherently have a number of problems. First, the circulation of credits in the network requires a fair amount of computation and storage resources and increases traffic overhead. Second, the systems fail to provide a way to know the service quality offered by a node and lack effective methods to punish a selfish and wealthy node (e.g., nodes that need few services) that sometimes drops others' packets.

Third, cooperative nodes located in a low-traffic region receive few forwarding requests, and thus may not earn enough credits for their own requests, while nodes located in a high-traffic region have more chances to earn more credits than they actually need and thus may drop some messages. Finally, the implementation of credits and virtual banks brings more complexity with high requirements on transmission security. For example, since credits are stored at the head of a packet which is transmitted through several nodes, how to prevent the credits from being stolen becomes a problem.

Directly combining a reputation system and a price system could foster cooperation incentives to a certain extent, but still cannot essentially resolve the individual problems such as reputation misreports and collusion, equal treatment to trustworthy nodes, cooperative and poor nodes, and complex implementation. What's even worse, the direct combination makes the problem of resource consumption and scalability even more severe. A formidable challenge is how to efficiently and coordinately combine the two systems to avoid problems in individual systems, ensuring they can be exploited to their fullest capacities.

To efficiently and effectively encourage cooperative and deter selfish behaviors, we propose a hierarchical Account-aided Reputation Management system (ARM). ARM selects low-mobility and trustworthy nodes as reputation managers (managers in short), builds them into a locality-aware distributed hash table (DHT) [20] infrastructure, and coordinately integrates reputation system and price system through the infrastructure. DHT is well-known for high scalability, efficiency and reliability, thus supports scalable and efficient operations in ARM. It helps to marshal all reputation and transaction information of a node into one manager, which calculates the reputation and increases/decreases credits in the account of the node accordingly. A node with $R_g < \mathcal{T}$ or deficit account is put into blacklists. Specifically, ARM consists of three components:

- *A locality-aware DHT infrastructure.* The infrastructure efficiently collects all reputation and transaction information of a node for effective reputation and account management. Experiment results show that including the maintenance overhead in node mobility, ARM still generates much lower overhead than current reputation and prices systems.
- *Reputation management.* Relying on the collected global reputation information by DHT, ARM effectively detects the false information, and accurately calculates node reputation. Also, with the aid of DHT, ARM reduces each node's burden for periodical information exchange and for storage and computing.
- *Reputation-adaptive account management.* ARM treats the nodes with different reputations differently, and also prevents nodes from gaining fraudulent benefits. Specifically, a higher-reputed node pays lower price while a lower-reputed node pays higher price for service. Also, a high-reputed node earns more credits than a low-reputed node for the same forwarding service. Using the DHT, ARM has no virtual credits circulating in the network and eliminates the implementation complexity.

In ARM, *uncooperative and reputed* nodes in reputation systems will have deficit accounts and *uncooperative and wealthy* nodes in price systems will have $R_g < \mathcal{T}$ quickly. Also, because a cooperative node pays lower price for the service, the credits earned by a node in a low-traffic area can sustain its requests. As far as we know, this is the first work that coordinately integrates the reputation system and price system through a DHT to efficiently and effectively provide cooperation incentives.

The remainder of this paper is organized as follows. Section II provides related works for cooperation incentive provision in MANETs. Section III describes the ARM system. Section IV presents simulation results to demonstrate the effectiveness and efficiency of ARM compared with a reputation system and a price system. Section V concludes this paper.

II. Related Work

Reputation systems can be classified into two categories: direct observation [9] and indirect observation [4]–[8], [10]–[14] methods. In the former, nodes independently assess their neighbors' reputations based on their direct interactions. It reduces the complexity of reputation management and achieves performance comparable to the approaches requiring reputation exchanges in a certain scenario. In the latter, nodes periodically share the reputation information they observed with others. The works in [4], [7] use techniques of *watchdog* and *pathrater*. *Watchdog* in a node promiscuously listens to the transmission of the next node in the path in order to detect misbehaviors. *Pathrater* in a node keeps the rating of other nodes to avoid interaction with uncooperative nodes in the transmission. CONFIDANT [5] detects uncooperative nodes and informs other nodes of observed misbehavior. Wu and Khosla [6] proposed an authentication mechanism to authenticate reputation messages in order to prevent a selfish node from playing tricks to benefit itself. Anantvalee and Wu [10] introduced a new type of nodes called suspicious nodes, which will be further investigated to see if they tend to behave selfishly by two reputation threshold(s). Buchegger *et al.* [11] proposed a Bayesian prediction mechanism to increase system robustness to falsely disseminated information. They also investigated the effect of using rumors on the detection time of misbehaving nodes and the robustness of reputation systems against wrong accusations. Mundinger *et al.* [12] built a stochastic process to formulate the behavior of the nodes in the system and derived a mean ordinary differential equation for misreport detection. Luo *et al.* [13] built a fuzzy logic model to deal with the uncertainty and tolerance of imprecise data inputs.

Price systems [15], [17]–[19], [21] provide incentives for cooperation by using micro payment. Buttyan *et al.* [15], [17] proposed two payment models: packet purse model, in which a source node pays relay nodes by storing virtual credits in the packet head, and packet trade model, in which a relay node buys packets from the previous node and sells

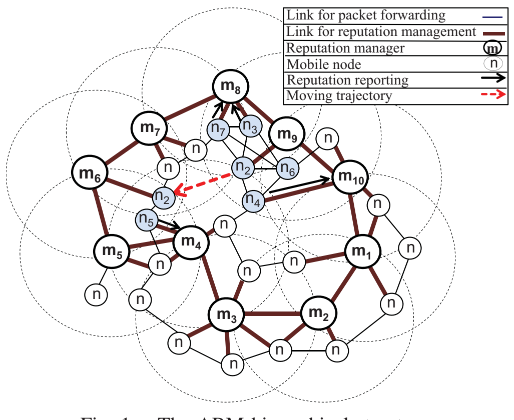


Fig. 1. The ARM hierarchical structure.

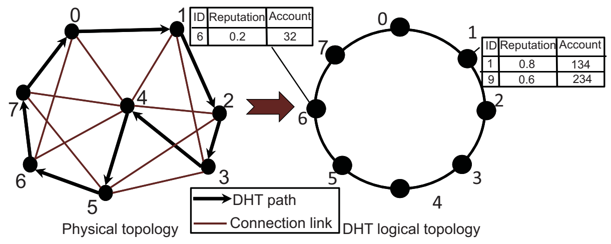
them to the next node in the path. In the credit-based system in [18], when a node forwards a message, it keeps a receipt and uploads it to the credit clearance service for credits. Crowcrof *et al.* [21] proposed a traffic price approach, in which the compensation of message forwarding depends not only on the energy consumption of the transmission but also on the congestion level of the relaying node. A node chooses a route to a destination with the minimum route compensation. Janzadeh *et al.* [19] proposed a price-based cooperation mechanism that utilizes hash chains to defend against cheating behavior. As described, individual reputation and price systems are not insufficiently efficient and effective. These deficiencies are confirmed in our previous work [22] which used game theory to investigate the underlying cooperation incentives of both systems. ARM resolves the problems in the individual system and greatly enhances the system efficiency and effectiveness by coordinately integrating the two systems through a DHT-based infrastructure.

III. The Design of the ARM System

A. Locality-aware DHT Infrastructure

Figure 1 illustrates the hierarchical structure of ARM. The higher level is a DHT network composed of managers (low-mobility and high-trustworthy nodes) and the lower level is composed of normal mobile nodes. A DHT network can partition ownership of a set of objects (e.g., files) among participating nodes, and efficiently route messages to the unique owner of any given object. Each object or node is assigned an ID that is the hashed value of the object (e.g., file name) or node IP address using consistent hash function [23]. An object is stored in a node whose ID equals to or immediately succeeds to the object's ID. The DHT provides two main functions: `Insert(ID,object)` and `Lookup(ID)` to store an object to a node responsible for the ID, and to retrieve the object. The message for the two functions is forwarded based on the DHT routing algorithm. The DHT achieves $O(\log n)$ path length per lookup request by using $O(\log n)$ neighbors per node, where n is the number of nodes in the system.

We leverage the Chord DHT network [20] for the infrastructure of ARM for scalable and efficient reputation and account management. ARM constructs a locality-aware DHT-based infrastructure where logical proximity abstraction



ID	Reputation	Account
6	0.2	32

ID	Reputation	Account
1	0.8	134
9	0.6	234

Fig. 2. Construction of the DHT infrastructure.

derived from the ARM matches the physical proximity information in reality. In this way, the packet routing path in the overlay is consistent to the packet routing path in the physical topology, which greatly reduces the physical routing distance and overhead. However, managers in MANETs are mobile while nodes in DHT networks are stable. Also, in a MANET, a node can only communicate with the nodes within its transmission range. This poses a challenge to build and maintain a DHT in a mobile environment.

The daily-increasing smartphones usually have dual-mode interfaces: low-power ad-hoc network interface (e.g., IEEE 802.11 interface) and high-power infrastructure network interface (e.g., WLAN radio interface). Thus, we assume some mobile nodes in the MANET have dual-mode interfaces. Some nodes in the MANET are highly trustworthy such as those equipped with tamper-proof equipment or owned by authorities (i.e., police stations, telecommunication companies or mobile servers). We assume the existence of an authority that can select high-trustworthy, low-mobility, and dual-mode interface nodes as managers for the DHT and notify all managers about the manager list. How to select managers is not the focus of this paper and remains as our future work.

a) Locality-aware DHT infrastructure construction: Figure 2 shows an example of a physical topology and its corresponding logical topology in ARM. In a logical topology, the distance between nodes' IDs represents their logical distance. To build managers into a locality-aware DHT infrastructure, we assign a sequence of consecutive DHT IDs to the managers along the path connecting all nodes once in a cycle.

In a MANET, each node identifies its neighbors by sending "hello" messages. Thus, a node can infer the relative physical closeness of its neighbors by the actual communication latency. To assign IDs to managers, as shown in Figure 2, we firstly choose a trustworthy bootstrap manager (m_0) and assign it ID 0. Then, it chooses its physically closest node as its successor, and assigns ID 1 to it. The successor finds its successor and assigns it ID 2. The process is repeated until the bootstrap node is reached. At this time, a complete cycle is formed and all managers have been assigned numerically continuous IDs. The last node in the created path with ID 7 must be in the transmission range of m_0, i.e., the successor of m_7 is m_0. Since only the physically close nodes can have sequential IDs, the constructed logical overlay topology is consistent with the physical topology of managers. Then, each manager builds a DHT routing table containing $\log N$ neighbors based on DHT neighbor determination protocol using broadcasting, where N is the number of managers in the system.

b) Locality-aware DHT infrastructure maintenance:

Proposition 3.1: In ARM, the average time period for a pair of neighbor managers to stay in the transmission range of each other (i.e., connection duration) is $\frac{r}{\overline{v}}$, where $\overline{v}$ is the average relative speed of their movement and r is the transmission range of a node.

Proof: Since the movement of each manager is independent and identical distributed (i.i.d), if manager m_i is d distance away from manager m_j and θ is the angle between r and d, the expected time period needed by m_i to move out of the transmission range of m_j is

$$E(T) = \int_{2\pi}^{0} \frac{1}{2\pi} \frac{\sqrt{r^2 + d^2 - 2rd\cos\theta}\cdot}{\overline{v}} \mathbf{d}(\theta) = \frac{r}{\overline{v}}.$$

■

Proposition 3.1 shows that the stability of the DHT infrastructure is primarily determined by the moving speed and transmission range of managers. To maintain the DHT structure in node mobility, managers need to maintain connectivity with their neighbors to guarantee that they are sequentially connected from ID 0 to $N-1$. Regarding node movement as node departure followed by node join, the original DHT maintenance mechanism could be used to maintain the ARM DHT infrastructure. However, it leads to high maintenance overhead due to node mobility. We propose a lightweight DHT maintenance algorithm to deal with node mobility.

Each manager relies on the "hello" messages to check its connectivity with its successor and update the managers in its routing table. When manager m_i senses its link to its predecessor m_{i-1} is about to break, it notifies m_{i-1}. When manager m_{i-1} receives the notification or senses that its link to its successor m_i is about to break, it finds an alternative path that ends in m_k $(k > i-1)$ and covers all managers with IDs $\in [i-1, k]$. The purpose of this operation is to maintain a complete DHT circle covering all managers with numerically continuous IDs. Since m_i moves in a local area, in order to find the path with low overhead, m_{i-1} pings manager m_{i+j} $(j \geq 2)$ in sequence by locally broadcasting a query message with $TTL = j$. That is, manager m_{i+2} is pinged first, then m_{i+3} is pinged, and so on. Each pinged manager replies m_{i-1} a message containing the routing path between them. Once the path covers ID$\in$ $[i-1, k]$, m_{i-1} reassigns IDs to the managers in the detected path in sequence to maintain numerically continuous IDs in the cycle. If no path is found after half of the managers in the system are pinged, then m_{i-1} functions as a bootstrap manager for DHT reestablishment. For routing table maintenance, when a manager notices that its routing table neighbor is not within its transmission range, it broadcasts a query message to find a new neighbor in that routing table entry.

As shown in Figure 3, when m_3 senses that its link to m_4 is about to break, it initializes a path querying process to find an alternate path covering all managers with ID$\in$ $[3, k]$ starting from itself and ending in m_k. m_3 first pings m_5. If such a path cannot be found, m_3 pings m_6, and then m_7, and so on. When an alternative path is discovered, the managers along the path will be assigned with new consecutive IDs for a complete circle. After finding a new path that travels

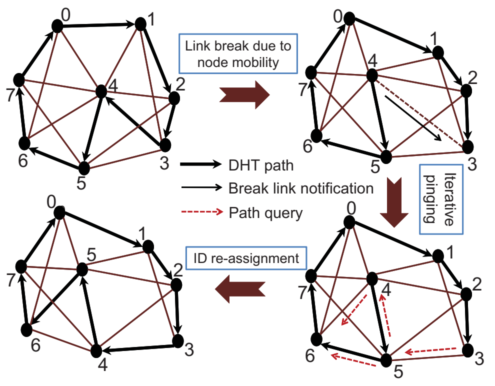


Fig. 3. Maintenance of the DHT infrastructure.

through manager m_i with ID 5 and m_j with ID 4, m_3 assigns m_i and m_j with ID 4 and 5, respectively.

c) DHT-based information collection and querying: The DHT supports efficient and scalable information collection and querying in ARM. Each normal mobile node n_i has a virtual id=i, which is the consistent hash of its IP address. In a DHT, an object is stored in a node whose id equals to or immediately succeeds to the object's id. We call this node as the object's owner manager. Nodes report the business information B and reputation information R of their observed data forwarding behaviors of a specific node n_i to their nearest managers. Relying on the `Insert(i,B+R)`, the managers marshal all the information of n_i in the system into its owner manager. The owner manager calculates n_i's reputation and increases/decreases the credits in its account. For example, in Figure 2, the information of the observed behavior of n_1 and n_9 are stored in m_1, which is responsible for their resource and account management. A node queries for the reputation of node n_i by sending `Lookup(i)` to its physically closest manager. The query will be forwarded to the owner manager of node n_i relying on DHT routing algorithm.

B. Reputation Management

In ARM, the reputation managers collect reputation information, calculate global reputation, identify misbehaving nodes, and manage nodes' accounts. ARM provides more accurate node reputation due to two reasons. First, it uses the global information rather than local partial information in reputation calculation. Second, the large amount of global information also makes it effective to detect falsified, conspiratorial and misreported information through deviation.

d) Neighbor monitoring: ARM uses neighbor monitoring to observe the packet-forwarding behaviors of nodes. Specifically, each observer uses a *watchdog* [4], [7] to keep track of the message forwarding behaviors of its neighbors. The observer records the total number of packets that n_i has received from other nodes for forwarding, denoted by D_i^r, and that n_i has forwarded, denoted by D_i^f during each time period T. Assume t_0 is the time instance that an observing node n_o joined in the system. At each time instance $t_0 + kT$ $(k \in [1, 2, 3...])$, n_o calculates the observed reputation value of node n_i by $R_i^{n_o} = \frac{D_i^r}{D_i^f}$, reports it to its closest manager, and resets D_i^r and D_i^f to zero. The manager then merges the collected reputations reported by the nodes in its transmission range to local reputation R_{l_i}.

Misreport avoidance. When a node in a region experiences an adverse network condition such as background interference due to traffic noise and thermal noise, the node's neighbors may also experience the adverse network condition. Thus, even though the nodes are cooperative, they are unable to transmit requested data. In this case, low R_ls are reported from the nodes that are clustered together. Also, the interfering regions and the nodes in the regions are changing constantly. ARM collects all reputations of a node in a region to one manager, which makes it easy to detect misreports. Therefore, when a manager notices that all nodes in an area report low R_ls, it temporally ignores the reports to reduce the uncertainty of the reported R_l, in order to avoid punishing nodes that fail to forward packets due to adverse network conditions.

False accusation avoidance. Some misbehaving nodes may report a high reputation for an uncooperative node, and a low reputation for a cooperative node. Since all observed reputations of a node in a region are collected into a manager and most nodes are benign, falsified reputations are always deviated largely from most reputations. Thus, in order to reduce the effect of falsified reports, a manager filters the R_is that dramatically deviate from the average R_i. The deviation of $R_i^{n_o}$ reported by node n_o about node n_i is calculated by:

$$\Delta R_i^{n_o} = |R_i^{n_o} - \sum_{n_j \in \mathbf{n}} R_i^{n_j}/|\mathbf{n}||, \tag{1}$$

where $\mathbf{n}$ denotes the group of observers that report R_i to the manger during T, and $|\mathbf{n}|$ denotes the number of nodes in the group. ARM sets a threshold δ_l for the deviation and ignores $R_i^{n_o}$ satisfying $\Delta R_i^{n_o} > \delta_l$. The manager m_o then calculates the local reputation value of n_i in T denoted by $R_{l_i}^{m_o}$.

$$R_{l_i}^{m_o} = \sum_{n_o \in \tilde{\mathbf{n}}} R_i^{n_o}/|\tilde{\mathbf{n}}|, \tag{2}$$

where $\tilde{\mathbf{n}}$ denotes $\mathbf{n}$ after removing the deviated observed reputations. Then, the manager reports the $R_{l_i}^{m_o}$ to n_i's owner manager using `Insert(i,`$R_{l_i}^{m_o}$`)`. According to Formula (1), the expected value of δ is

$$E(\delta) = \left| \frac{a \cdot \overline{R}_{l_h} + b \cdot \overline{R}_{l_f}}{a+b} - \overline{R}_{l_f} \right| = \frac{a(\overline{R}_{l_h} - \overline{R}_{l_f})}{a+b},$$

where $\overline{R}_{l_h}$ and $\overline{R}_{l_f}$ respectively denote the expected value of honest reports and false reports, and a and b respectively denote the number of honest reports and false reports in interval T.

Collusion avoidance. The nodes in a region may collude to conspiratorially report node reputations in order to fraudulently increase their reputations or decrease others' reputations. For example, the nodes in group A and group B are the nodes in the transmission range of m_k. The number of nodes in group B overwhelms group A. If the nodes in group B collude with each other to report low R_i for n_i, then the justified reports from group A are ignored by m_k according to Formula (1). This problem can be resolved by another filtering process at the owner manager m_i that collects all $R_{l_i}^{m_o}$ from different managers m_o. Again, m_i computes the variance of $R_{l_i}^{m_o}$ based on Equation (3), and ignores $R_{l_i}^{m_o}$ with $\Delta R_{l_i}^{m_o} > \delta_g$. δ_g can be determined in the same way as δ_l.

$$\Delta R_{l_i}^{m_o} = |R_{l_i}^{m_o} - \sum_{m_j \in \mathbf{m}} R_{l_i}^{m_j}/|\mathbf{m}||, \tag{3}$$

where $\mathbf{m}$ is the number of managers that report R_{l_i}. After that, the global reputation of node n_i is calculated as

$$R_{g_i} = \sum_{m_o \in \tilde{\mathbf{m}}} R_{l_i}^{m_o}/|\tilde{\mathbf{m}}|. \tag{4}$$

where $\tilde{\mathbf{m}}$ is the group of $\mathbf{m}$ after filtering.

For example, in Figure 1, node n_3, n_4, and n_7 monitor the transmission of n_2. Nodes n_3 and n_7 report observed reputation of n_2 to manager m_8, and n_4 reports its observed reputation of n_2 to m_{10}. Then, m_8 and m_{10} merge the reported reputations to a local reputation value of n_2 in its region, denoted by $R_{l_2}^{m_8}$ and $R_{l_2}^{m_{10}}$, and report the results to n_2's owner manager, m_2. Later, when n_2 moves close to n_5, n_5 will start to monitor the transmission of n_2 and report the observed reputation of n_2 to its nearby manager m_4, which subsequently reports $R_{l_2}^{m_4}$ to manager m_2. Therefore, all local reputations of n_2 are marshaled in m_2, which then calculates the global reputation for n_2. Unlike most existing reputation systems in which a node calculates its neighbors' reputation values based on its local observations and cannot easily retrieve its new neighbor's previous reputation, ARM globally collects all R_{l_i} of node n_i at all times in all regions for global reputation calculation, leading to more accurate reflection of n_i's trustworthiness. Also, global information (i.e., large data sample) makes it easy to precisely detect false information.

When a node is suddenly out of power or suffers from channel congestion, it cannot offer service to others and thus has low reputation though it is not actually selfish. It is unfair to punish such a cooperative node with low reputation. On the other hand, it is difficult to identify the real reason for a low reputation. Therefore, ARM takes into account the old reputation when calculating the new reputation [8]. That is,

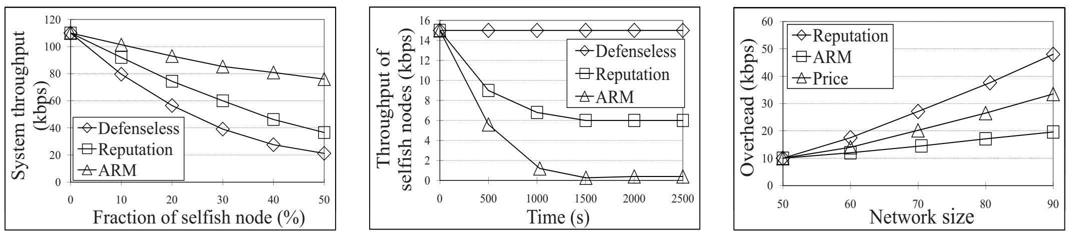
$$R_g^{new} = \alpha R_g^{old} + (1-\alpha) R_g, \tag{5}$$

where R_g is the currently calculated reputation value for period T and α is a weight factor that is adaptive to the traffic load in the system. In a system with high traffic, a node is more likely to be out of power or congested. Then, α should be set to a larger value. Therefore, we specify $\alpha = \bar{D}/\bar{C}$, where $\bar{D}$ is the average number of packets generated per second in the system and $\bar{C}$ is the total channel capacity of the system.

ARM periodically decreases the reputations of the nodes whose $R_g > \beta\mathcal{T} + (1-\beta)R_g^{max}$ $(\beta < 1)$ by:

$$R_g^{new} := \varphi R_g^{new} (\varphi < 1), \tag{6}$$

where R_g^{max} denotes the maximum global reputation, β and φ are weight factors. The rationale behind this policy is that the reputation of a high-reputed node will decrease over time if it does not receive new rating from others. Low reputation subsequently increases the service price for message forwarding of the node (Section III-C). Therefore, the only way a node can enjoy the low price is to cooperate with other nodes all the



(a) Average system throughput (b) Throughput of selfish nodes (c) System overhead

Fig. 4. Performance comparison between different systems.

time. As other reputation systems, ARM also sets a reputation threshold $\mathcal{T}$ to determine whether a node is selfish or not. Smart selfish nodes may keep their reputation just above $\mathcal{T}$. Thus, they can sometimes drop packets while being regarded as reputed nodes. These nodes will be detected by the account management function in ARM. That is, if a node always generates packets rather than forwarding packets for others, it will eventually run out of credits and be detected as selfish node.

C. Reputation-adaptive Account Management

ARM has an account management function to avoid equal treatment of high-reputed nodes in different reputation levels in order to effectively provide cooperation incentives and deter selfish behaviors. ARM assigns each newly joined node with an initial number of credits denoted by $A(0)$. The owner managers of nodes maintain their accounts, and transparently increase and decrease the credits in the accounts of forwarding service providers and receivers, respectively. Thus, different from previous price systems, ARM's account management does not need credit circulation in the network, reducing transmission overhead and system complexity.

Notice that R_g of a node equals to the percent of the forwarded packets among the node's received packets, we directly use R_g for the calculation of earned credits. Specifically, node n_i's owner manager increases its account in every T by

$$P_e = p_r R_{g_i}, \tag{7}$$

where p_r is a constant credit rewarding factor.

In order to foster the cooperation incentives, ARM connects forwarding service cost per packet p_c of a node to its reputation, so that higher-reputed nodes pay fewer credits while lower-reputed nodes pay more credits for the same forwarding service. The p_c of n_i, denoted by p_{c_i}, is calculated by:

$$p_{c_i} = \frac{\gamma}{R_{g_i}^{new}}, \tag{8}$$

where γ is a constant value.

When an observing node n_o notices that N_{p_i} packets of node n_i have been transmitted by others during T, it reports this business information B_i to its nearest manager along with R_i. By DHT function `Insert(i,`$B_i + R_i$`)`, the manager forwards the information to n_i's owner manager m_i, which then deducts $p_{c_i} N_{p_i}$ credits from n_i's account. Therefore, the account of node n_i at time $t_0 + kT$ ($k \in [1, 2, 3...]$) is:

$$A(t) = A(0) - \sum_{t=t_0}^{t_0+kT} (p_{c_i}(t) \cdot N_{p_i}(t) - p_r \cdot R_{g_i}(t)). \tag{9}$$

When the account of node n_i is negative, managers notify all nodes to put node n_i in their blacklists.

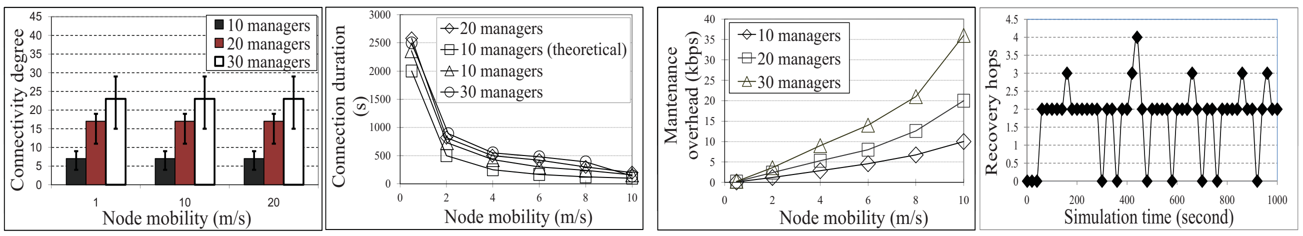
IV. Performance Evaluation

We conducted simulations on NS-2 [24] to demonstrate the performance of ARM. We describe our default settings below unless otherwise specified. The simulated network has 60 wireless nodes randomly deployed in a field of 1200×1200 square meters. We randomly selected 10 nodes as managers. The radio transmission ranges of low-power and high-power interfaces were set to 250m and 1000m, respectively. The raw physical link bandwidth was set to 2Mbits/s. The height of antennas for data transmitting and receiving was set to 1.5 meters. We used the random way-point mobility model [25] to generate node movement. The nodes are i.i.d deployed in the field. The nodes move at a speed randomly chosen from [1,10]m/s, wait for a pause time randomly chosen from [0,10]s, and then move to another random position. We randomly chose 10 pairs of source and destination nodes in every 40s. The range of reputation was set to [0,1], and the reputation threshold $\mathcal{T} = 0.4$. Each simulation lasted $5000s$. We run 10 simulations and reported the average as the experiment results.

We set $\alpha = 0.5$ in Formula (5), $\varphi = 0.5$ in Formula (6) and $p_r = 2$ in Formula (7). The time period T for periodically information reporting for mobile nodes and managers were set to 10s and 50s, respectively. Each node initially was assigned 5000 credits and 1 reputation value. We compared the performance of the DSR [26] routing algorithm in a defenseless MANET with neither reputation system nor price system (*Defenseless*), in a MANET with ARM (ARM), with the *Reputation* [5] reputation system (*Reputation*), and with a price system [15] (*Price*). To make results comparable, rather than using the absolute number of forwarded packets, we use $R_l = \frac{D^r}{D^f}$ to evaluate a node's reputation in *Reputation*. Selfish nodes keep their reputation just above $\mathcal{T}$. In the routing, a node chooses a node not in its blacklist for data forwarding.

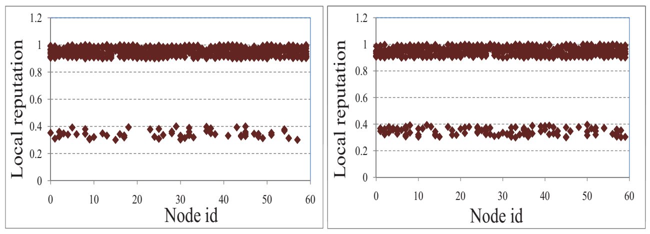
A. Performance Comparison of Different Systems

Figure 4(a) plots the average system throughput of different systems versus the percent of selfish nodes. The figure shows that ARM generates higher throughput than *Reputation*, which produces higher throughput than *Defenseless*. In *Defenseless*, a selfish node drops all of its received packets. *Reputation* can force the selfish nodes to be cooperative to a certain extent. However, a selfish node still can keep R_g just above $\mathcal{T}$ by dropping received packets with probability of $\mathcal{T}$. In ARM, the selfish nodes finally do not have enough credits to pay for their transmission services and are put into the blacklists. Therefore, ARM produces higher throughput than *Reputation*. Also, the figure shows the throughput of the system decreases as selfish



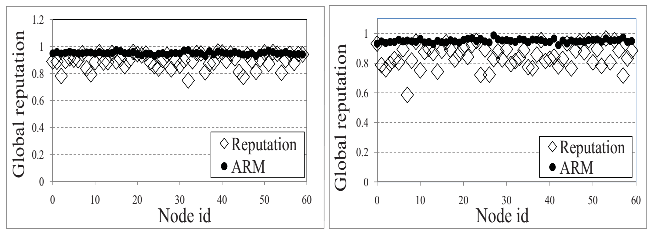
(a) Connectivity degree per manager (b) Average connection duration (c) Topology maintenance overhead (d) Reassigned IDs in DHT maintenance

Fig. 5. Performance of the locality-aware DHT infrastructure in ARM.



(a) Five false-reporting nodes (b) Ten false-reporting nodes

Fig. 6. Local reputations in a defenseless system.



(a) Five false-reporting nodes (b) Ten false-reporting nodes

Fig. 7. Node reputations in a defensive system.

nodes grow. Since *Defenseless* and *Reputation* cannot detect all selfish nodes, their throughput decreases as the fraction of selfish nodes grows. It is intriguing to see that ARM also exhibits performance degradation though it can detect most selfish nodes. This is because selfish nodes may be chosen as forwarding nodes before their credits are used up. Also, avoiding selfish nodes in routing leads to longer path lengths, which suffers from higher transmission interference.

In order to verify the effectiveness of punishing selfish nodes by refusing their transmission requests, we tested the throughput of packets generated by selfish nodes over a time interval. We setup 10 selfish nodes and used them as source nodes. Figure 4(b) plots the throughput of the selfish nodes. In *Defenseless*, selfish nodes keep constant throughput of 15kbps. In *Reputation*, the throughput decreases as time elapses and then keeps constantly at 6kbps. This is because the selfish nodes keep R_g just above $\mathcal{T}$, thus their transmission requests are accepted by other nodes. The throughput in ARM declines sharply as time goes on and finally reaches 0. This means that with the aid of account management, ARM can effectively detect and punish selfish nodes, excluding them from the network.

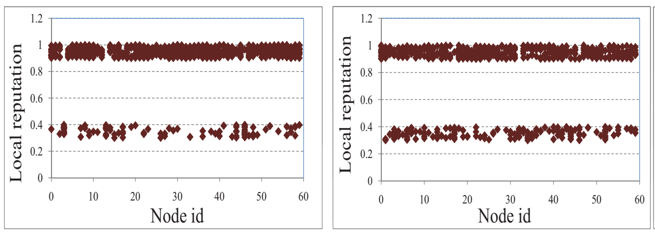
To evaluate the efficiency of the systems, we tested the overhead measured in kbps for all overhead messages in the systems. In addition to the "hello" messages, the overhead messages in ARM also include those for topology construction and maintenance and reputation querying; in *Reputation* also include the messages for reputation exchange; in *Price* also include the messages for credits payments. Figure 4(c) illustrates the overhead in each system versus network size. The figure demonstrates that ARM yields much less overhead than *Price* which produces less overhead than *Reputation*. In ARM, since nodes only communicate with managers, the overhead is in proportion to the network size. Though ARM needs to construct and maintain DHT infrastructure in node mobility, its total overhead is still lower than others. In *Reputation*, reputation information is exchanged among local nodes periodically, resulting in much higher overhead. In *Price*, credit circulation in the network generates transmission overhead. The results confirm that ARM consumes less resource than other systems.

B. Evaluation of the DHT Infrastructure in ARM

We measured the average, maximum and minimum of connectivity degree per manager when the managers move at the speed of 1m/s, 10m/s and 20m/s. In addition to the default experiment scenario, we also measured the performance with additional 10 and 20 managers, respectively. Figure 5(a) shows that the smallest connectivity degree of a manager is about $\frac{N}{2}$. The figure also shows that more managers incur higher connectivity degree because a manager has more neighbors in a DHT with more nodes. We find that the node mobility does not affect the connectivity degree per manager. The result illustrates that the proposed DHT maintenance mechanism can establish new links immediately upon link breakups.

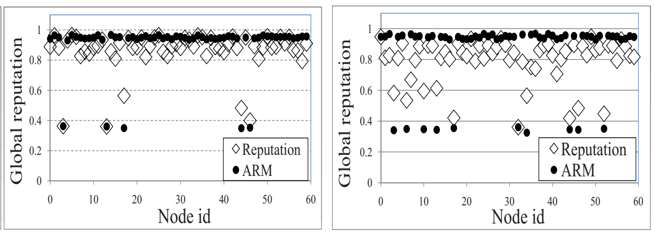
Figure 5(b) presents the average connection duration of managers versus node mobility. We also include the theoretical results based on Proposition 3.1 in the case of "10 managers". The figure demonstrates that when the mobility is 0.5 m/s, the DHT infrastructure is much more stable than other situations. As node mobility increases, the average connection duration drops sharply. We can also find that the connection duration stays almost the same for different number of managers. This is because with the high-power interface, a manager can contact another manager within a long range. Thus, the number of managers does not greatly affect the stability of DHT infrastructure. The simulation results closely match the theoretical result. The small gap between them is because the theoretical analysis does not consider node pause time during movement.

Figure 5(c) shows the maintenance overhead of the DHT infrastructure versus node mobility. The overhead is represented by the number of messages exchanged for DHT maintenance. The overhead grows with the increase of node mobility. Higher mobility leads to higher probability of link breakups, incurring



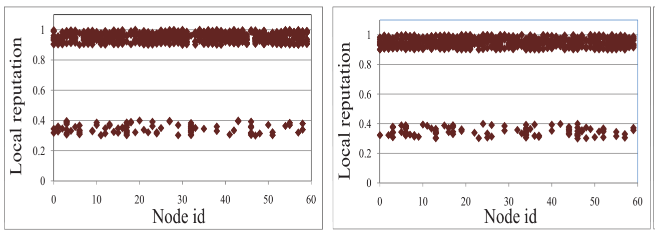
(a) Defenseless system (5 colluders) (b) Defenseless system (10 colluders)

Fig. 8. Reputations in a defenseless system with non-group collusion.



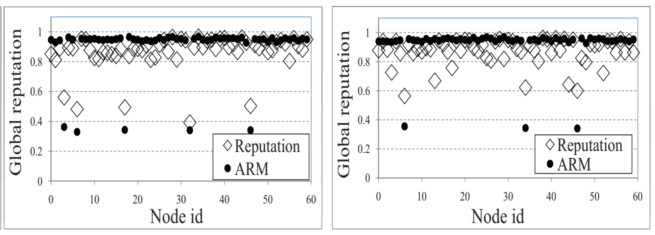
(a) Defensive system (5 colluders) (b) Defensive system (10 colluders)

Fig. 9. Reputations in a defensive system with non-group collusion.



(a) Defenseless system (5 colluders) (b) Defenseless system (10 colluders)

Fig. 10. Reputations in a defenseless system with group collusion.



(a) Defensive system (5 colluders) (b) Defensive system (10 colluders)

Fig. 11. Reputations in a defensive system with group collusion.

higher maintenance overhead. The overhead also grows as the number of managers increases, because more managers generate more messages for DHT maintenance. Therefore, fewer nodes with low mobility should be chosen as managers in order to reduce DHT maintenance overhead.

Figure 5(d) shows the number of nodes that have been reassigned IDs in DHT maintenance over time. It shows that 2 nodes need ID reassignment for DHT maintenance most of the time. Although the physical link between nodes n_i and n_j is broken, they still share the same manager neighbors. Therefore, most of time, by re-ordering the IDs of n_i and n_j and the shared neighbors, the DHT structure with numerically continuous IDs can be recovered.

C. Performance in False Accusation Resilience

In this experiment, all the nodes are cooperative. We chose some nodes that deliberately evaluate their neighbors with low reputations randomly chosen in $[0.3, 0.4]$. In the defenseless system, every node rates its neighbors based on their forwarding behavior. Figure 6(a) and (b) plot all evaluated local reputations of each node in a defenseless system with 5 and 10 false-reporting nodes, respectively. Because of the false-reports, some of the cooperative nodes are rated with low reputations. Comparing Figure 6(a) and (b), we can see that as the number of the false-reporting nodes increases, the number of low reputations each node received increases.

To make the global values of a given node in different nodes the same, we used broadcasting to ensure each node has receives others' local reputations. Figure 7 shows the global reputation of each node in *Reputation* and ARM. *Reputation* exhibits large variance in reputations and cannot accurately reflect cooperative nodes' reputations. This is because in *Reputation*, each node considers the false reports when calculating the global reputations. In the figure, all reputations in ARM are close to 1. This means ARM can more accurately reflect node reputations. Some reputations are not 1 because some cooperative nodes may drop packages due to interference in transmission.

Comparing Figure 7(a) with Figure 7(b), we find that more false-reporting nodes in the system generate greater variance in node reputations in *Reputation*, while they do not have influence on node reputations in ARM. More false reports incur higher inaccuracy of node reputations in the final global reputation calculation in *Reputation*. In contrast, by taking advantage of the DHT infrastructure, ARM can efficiently gather all local reputations of each node in the system and filter out the false reports.

D. Performance in Collusion Resilience

According to the movement of the colluders, collusion can be classified to non-group collusion and group collusion. In the former, the colluders move individually, and they report high reputation for each other when meeting together. In the latter, all colluders in a group move together as a group, and always rate high reputations for each other. We conducted experiments for both non-group collusion and group collusion. We consider collusion where colluders drop received packets with probability 0.3, and falsely report low reputation randomly chosen in [0.3,0.4] for their neighboring cooperative nodes, and higher reputation randomly chosen in [0.9,1] for other colluders.

Figure 8(a) and (b) show node local reputations in a defenseless system with 5 and 10 colluders, respectively. The figures show that a certain portion of nodes receive low R_ls. These low R_ls are from the false reports of the colluders on benign nodes and correct reports from benign nodes on colluders. Comparing the two figures, we find that the number of low R_ls is proportional to the number of colluders in the system.

Figure 9 (a) and (b) show the global reputation of each node in *Reputation* and ARM in non-group collusion, respectively. *Reputation* exhibits a larger variance than ARM in the reputations of cooperative nodes. This is because *Reputation* includes the false reports for the cooperative nodes in calculating global reputation. By collecting all the reports in the system through the DHT infrastructure, ARM can easily identify and filter the reports from colluders that are largely different from others, since the majority of the nodes in the system are benign. Also, though both systems can identify the colluders, *Reputation* cannot accurately reflect the R_gs of colluders since some colluders have high R_gs. This is because the colluders report high R_l for each other when meeting each other. *Reputation* takes these false reports into account while ARM filters them out when calculating R_g.

Figure 10 shows the local reputations for group collision in a defenseless system. Compared to Figure 8, Figure 10 has much less low node R_ls due to two reasons. First, in the group node collusion, the colluders can always report high reputations for each other to increase their own reputations. Second, more colluders generate more low R_ls for cooperative nodes. Figure 11(a) and (b) show the global reputation with group collision in *Reputation* and ARM. When the number of colluders is 5, even they always collude with each other, *Reputation* and ARM can identify the colluders since the majority of the neighbors of a colluder are benign. By filtering out the false reports, ARM generates more accurate R_gs than *Reputation* for both co-operative nodes and colluders. However, when the number of the colluders increases to 10, it is very difficult for *Reputation* to detect colluders. Also, ARM cannot detect some colluders directly based on reputation. Since the majority of the neighbors of a colluder are colluders and the false reports from the colluders overwhelm the reports from benign nodes. The account management in ARM can help to detect the colluders as shown in Figure 12.

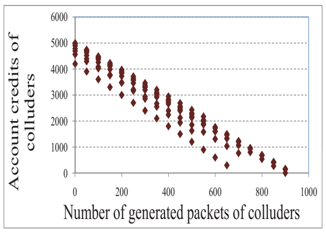


Fig. 12. Credits of colluders.

Figure 12 shows the account value of each colluder versus the number of its generated packets in ARM. We can observe that the colluders' account credits decrease linearly as they generate more packets. Although the colluders can keep high R_g by rating high for each other and receive fraudulent benefit of low service price, they will ultimately use up their credits as they generate more packets, and finally are detected as uncooperative nodes by deficit accounts.

V. Conclusions

Previous reputation systems and price systems in MANETs cannot effectively prevent selfish behaviors and also generate high overhead. In this paper, we propose a hierarchical Account-aided Reputation Management system (ARM) to efficiently and effectively deter node selfish behaviors and provide cooperation incentives. ARM builds an underlying locality-aware DHT infrastructure to efficiently collect global reputation information in the entire system for node reputation evaluation, which avoids periodical message exchange, reduces information redundancy, and more accurately reflects a node's trustworthiness. ARM has functions of reputation management and account management, the integration of which fosters the cooperation incentives and non-cooperation deterrence. ARM can detect the uncooperative nodes that gain fraudulent benefits while still being considered as trustworthy in previous reputation systems and price systems. Also, it can effectively identify falsified, conspiratorial and misreported information so as to provide accurate node reputations that truly reflect node behaviors. In our future work, we will study show to choose reputation manager in a distributed manner.

Acknowledgements

This research was supported in part by U.S. NSF grants OCI-1064230, CNS-1049947, CNS-1025652, CNS-1025649, and CNS-0917056, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282.

References

[1] The state of the smartphone market. http://www.allaboutsymbian.com/.
[2] Next Generation Smartphones Players, Opportunities & Forecasts 2008-2013. Technical report, Juniper Research, 2009.
[3] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of MobiCom*, 2000.
[4] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in manets. In *Proc. of CMS*, 2002.
[5] S. Buchegger and J.-Y. L. Boudec. Performance analysis of the confidant protocol. In *Proc. of MobiHoc*, 2003.
[6] Q. He, D. Wu, and P. khosla. Sori: A secure and objective reputation-based incentive scheme for ad-hoc networks. In *Proc. of WCNC*, 2004.
[7] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of MobiCom*, 2000.
[8] T. Anantvalee and J. Wu. Reputation-based system for encouraging the cooperation nodes in mobile ad hoc networks. In *Proc. of ICC*, 2007.
[9] S. Bansal and M. Baker. Observation-based cooperation enforcement in ad hoc networks. *Technical Report, CS Dept., Stanford University*, 2003.
[10] T. Anantvalee and J. Wu. Reputation-based system for encouraging the cooperation nodes in mobile ad hoc networks. In *Proc. of ICC*, 2007.
[11] S. Buchegger and J. Y. LeBoudec. A robust reputation system for mobile ad-hoc networks. In *Proc. of P2PEcon*, 2004.
[12] J. Mundinger and J. Le Boudec. Analysis of a reputation system for mobile ad-hoc networks with liars. *Performance Evaluation*, 2008.
[13] J. Luo, X. Liu, and M. Fan. A trust model based on fuzzy recommendation for mobile ad-hoc networks. *Computer Network*, 2009.
[14] T. Anantvalee and J. Wu. Reputation-based system for encouraging the cooperation of nodes in mobile ad hoc networks. In *Proc. of ICC*, 2007.
[15] M. Jakobsson, J. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks. In *Proc. of Financial*, 2003.
[16] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proc. of MobiHoc*, 2000.
[17] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organzing mobile ad hoc network. *ACM Journal for MONET*, 2002.
[18] S. Zhong, Y. R. Yang, and J. Chen. Sprite: A simple, cheat-proof, credit-based system for mobile ad hoc networks. In *Proc. of INFOCOM*, 2003.
[19] H. Janzadeh and K. Fayazbakhsh and M. Dehghan and M. S. Fallah. A secure credit-based cooperation stimulating mechanism for MANETs using hash chains. *Elservier, Future Generation Comput. Sys.*, 2009.
[20] I. Stoica, R. Morris, and et al. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *TON*, 2003.
[21] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modelling incentives for collaboration in mobile ad hoc networks. In *Proc. of WiOpt*, 2003.
[22] Z. Li and H. Shen. Analysis of a hybrid reputation management system for mobile ad hoc networks. In *Proc. of ICCCN*, 2009.
[23] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and Panigrahy R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proc. of STOC*, 1997.
[24] The network simulator ns-2. http://www.isi.edu/nsnam/ns/.
[25] Delay tolerant networking research group. http://www.denrg.org.
[26] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *IEEE Mobile Computing*, 1996.