A Hierarchical Account-Aided Reputation Management System for MANETs

Haiying Shen, *Senior Member, IEEE, Member, ACM*, and Ze Li

***Abstract*—Encouraging cooperation and deterring selfish behaviors are important for proper operations of mobile ad hoc networks (MANETs). For this purpose, most previous efforts rely on either reputation systems or price systems. However, these systems are neither sufficiently effective in providing cooperation incentives nor sufficiently 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 consumes 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 distributed hash table (DHT) infrastructure for efficient and integrated operation of both reputation and price systems. The infrastructure helps to globally collect all node reputation information in the system, which can be used to calculate more accurate reputation and detect abnormal reputation information. Also, ARM integrates reputation 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 the individual reputation system and price system in terms of effectiveness and efficiency of providing cooperation incentives and deterring selfish behaviors.**

***Index Terms*—Mobile ad hoc networks, price systems, reputation systems.**

I. Introduction

A MOBILE ad hoc network (MANET) is formed by a collection of mobile nodes without a fixed infrastructure or centralized management. Nowadays, wireless devices such as smartphones are increasingly prevalent in our daily life. The number of wireless Internet users has tripled worldwide over the past three years, and the number of smartphone 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 mobile nodes for pervasive communication in the future. In a MANET, nodes communicate with each other by routing data through relay nodes in a multihop manner. Thus, reliable communication is critical to the proper operation 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 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 sufficiently effective. By insufficient efficiency, we mean that the methods exacerbate the resource-efficiency problem in MANETs by consuming already scarce resources. By insufficient effectiveness, we mean that the methods lack the capability to accurately reflect nodes' behavior and prevent nodes from gaining fraudulent benefits. The accuracy of node reputation can be adversely affected by false information including falsified, conspiratorial, and misreported information. Falsified information is reported by a misbehaving node in order to deliberately increase/decrease others' reputations. Conspiratorial information is generated by colluders that report high reputations for each other to raise their own reputations and report low reputations for others to decrease their reputations. Misreported information in this paper means low reputations for cooperative nodes who cannot offer high-quality transmission service due to adverse network conditions such as background interference. In this paper, we only focus on these three kinds of false information though there are many other kinds.

In most current reputation systems [3]–[14], a node collects locally generated node feedback and aggregates it to yield the global reputation values (R_g) for others based on periodic information exchanges between neighbors. The node whose reputation is below a predefined threshold ($\mathcal{T}$) is considered selfish and put into a blacklist. However, the systems suffer from a number of problems. First, they lack efficient mechanisms to collect and propagate reputation information. Periodic information exchanges, keeping redundant reputations in each node, and broadcasting to query reputations [7] consume significant resources, thus failing to achieve high scalability. Second, reputa-

Manuscript received February 12, 2012; revised January 02, 2013 and May 24, 2013; accepted November 09, 2013; approved by IEEE/ACM Transactions on Networking Editor K. Kar. Date of publication December 05, 2013; date of current version February 12, 2015. This work was supported in part by the US NSF under Grants IIS-1354123, CNS-1254006, CNS-1249603, OCI-1064230, CNS-1049947, CNS-1025652, CNS-1025649 and CNS-0917056; the Microsoft Research Faculty Fellowship under Grant 8300751, and the United States Department of Defense under Grant 238866. An early version of this work was presented in the Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Shanghai, China, April 10–15, 2011.

H. Shen with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29631 USA (e-mail: shenh@clemson.edu).

Ze Li was with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29631 USA. He is now with MicroStrategy Incorporation, Tysons Corner, VA 22182 USA (e-mail: zel@clemson.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2013.2290731

tion calculation based on partial local information, which may include false information, may result in an insufficiently accurate reputation evaluation to truly reflect node behaviors. Third, solely relying on a reputation system is not effective enough to thwart uncooperative behaviors. The reputation systems only punish the nodes with $R_g < \mathcal{T}$. Thus, a node can be uncooperative for some time while still keeping $R_g \geq \mathcal{T}$. Even in a fine-grained reputation system that has many levels of reputation thresholds [7], nodes can still manage to stay above a threshold to avoid corresponding punishment.

Price systems [15]–[21] treat message forwarding as a service transaction, in which nodes forward others' messages in order to earn credits for their own message transmissions. However, these systems also have a number of problems. First, the circulation of credits in the network increases traffic overhead. Second, the systems fail to provide a way to know the service quality offered by a node. They also cannot detect and punish a selfish and wealthy node that earns many credits by being cooperative at first and then always drops others' packets while keeping its credit account higher than 0 [6], though such packet droppings greatly decrease system performance. Third, cooperative nodes located in a low-traffic region receive few forwarding requests, and thus may not earn enough credits for their own requests. However, 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 a high requirement on transmission security.

This paper aims to handle the aforementioned problems in reputation systems and prices systems for more effective cooperation incentives. Our previous game-theoretic analysis [22] proves that the combination of a reputation system and a price system can enhance the effectiveness of cooperation incentives of individual system. That is, the amount of credits a node needs to pay for a service is inversely proportional to its reputation. However, directly combining a reputation system and a price system by having them function independently and monitor nodes separately as before makes the problem of resource consumption and scalability even more severe. Also, such combination still cannot resolve some individual problems such as reputation misreports, collusion, and complex implementation. A formidable challenge is how to efficiently and coordinately combine the two systems to avoid the problems in the individual systems, ensuring they can be exploited to their fullest capacities. Therefore, in this paper, we focus on system design to handle this challenge, which is different from our theoretical work in [22].

We propose a hierarchical Account-aided Reputation Management system (ARM), which coordinately integrates a reputation system and a price system to effectively and efficiently deter and detect selfish behaviors. By leveraging the distributed hash table (DHT) [23], the reputation information of the nodes can be collected and accessed efficiently in a large-scale MANET. By identifying abnormal reported reputation values, ARM can effectively exclude the information from reputation misreporting and collusion in reputation calculation. As the credits of each node are directly managed by the reputation managers without the need of credit circulation in the network, the system complexity and the threat to security are reduced. Furthermore, ARM achieves fairness in reputation and price calculation by considering different region traffic loads.

ARM selects low-mobility and trustworthy nodes as reputation managers (managers in short), builds them into a locality-aware DHT infrastructure, and coordinately integrates a reputation system and a price system through the infrastructure. Thus, ARM avoids frequent information exchanges between nodes. A DHT provides two main functions: `Insert(ID,object)` to store an object to a node responsible for the ID, and `Lookup(ID)` to retrieve the object. These two functions help marshal each node's reputation and transaction information 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 a deficit account is put into blacklists. A DHT infrastructure achieves a time complexity of $O(\log N)$ for the two functions by using $O(\log N)$ neighbors per node, where N is the number of DHT nodes. Therefore, it supports scalable and efficient operations in ARM for large-scale MANETs by providing efficient information collection and querying services. We use the DHT infrastructure because, given a node's ID, it can efficiently forward an information report or an information query of the node to the node's manager. In a large-scale MANET network, it is not scalable to let each manager maintain a record for the mapping between each node and its manager or to let each manager maintain paths to all other managers.

Specifically, ARM consists of three components:

- *A locality-aware DHT infrastructure*. We study the requirements to create a locality-aware DHT infrastructure in a MANET and propose the construction and maintenance algorithms. The infrastructure efficiently collects node reputation and transaction information for effective reputation and account management. Experimental results show that even when including the maintenance overhead in node mobility, ARM still generates a lower overhead than current reputation and price systems.
- *Reputation management*. Relying on the collected global information by DHT, ARM effectively detects false information and accurately calculates node reputation. Also, ARM avoids periodical information exchange, the storage and computing burden of each node in the system.
- *Reputation-adaptive account management*. ARM treats nodes with different reputations differently and also prevents nodes from gaining fraudulent benefits. Specifically, a higher-reputed node pays a lower price while a lower-reputed node pays a 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 system can be detected based on their deficit accounts and *uncooperative and wealthy* nodes in price system can be detected as they quickly reach $R_g < \mathcal{T}$. Also, because a cooperative node pays less for the service, the credits earned by a node in a low-traffic area can sustain its requests. As a result, our reputation system for MANETs can more efficiently effectively encourage cooperation incentives and deter selfish behavior and misbehaviors. Note that ARM is for MANETs that have some low-mobility and trustworthy nodes.

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

II. Related Work

In this section, we present the related work about reputation systems and price systems in MANETs. Reputation systems can be classified into two categories: direct observation [24]–[28] and indirect observation [3]–[14] methods. In the former, nodes independently assess their neighbors' reputations based on their direct interactions. OCEAN [26] avoids indirect reputation information and uses only direct observations in order to see the performance of this method. To increase the routing reliability, Conti *et al.* [24] proposed a reliability indexing mechanism, in which each node controls its in/out traffic based on the reliability index value associated with the neighbor through which the packet is forwarded. Dewan *et al.* [25] calculated the reputation value of each node only based on its past history of relaying packets. Liu *et al.* [27] proposed to expand the scope of the behavior observation from one hop to two hops. Jaramillo *et al.* [28] showed that the punishment policy in reputation systems always leads to a retaliation situation where detected selfish nodes never cooperate again, decreasing the throughput of cooperative users. They proposed a contriteness strategy to avoid the retaliation situation. However, because the node behaviors that each node can observe are limited, exclusively relying on direct observations may increase the selfish and misbehaving node detection time.

In the indirect observation, nodes periodically share their observed reputation information with others. The works in [3] and [4] use the techniques of *watchdog* and *pathrater*. The *Watchdog* in a node promiscuously listens to the transmission of the next node in the path in order to detect misbehaviors. The *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 [7] introduced a new type of node called suspicious nodes, which will be further investigated to see if they tend to behave selfishly with two reputation thresholds. Zong *et al.* [8] proposed to calculate the reputation values based on an artificial neural network in order to tune the parameters automatically to adapt to various personal requirements. Refaei *et al.* [9] introduced a time-slotted approach to allow the evaluation function to quickly and accurately capture changes in node behavior. Also, they use a sequential probability ratio to distinguish between cooperative and misbehaving neighbors by tracking their misbehaving probabilities. The indirect observation based reputation systems decrease the selfish and misbehaving node detection time. However, the malicious nodes may report false information about other nodes, which may decrease selfish node detection accuracy.

In order to reduce the number of false ratings, Buchegger *et al.* [10] proposed a Bayesian prediction mechanism to increase system robustness with regard to falsely disseminated information. Mundinger *et al.* [11] built a stochastic process to formulate the behavior of the nodes in the system and derive a mean ordinary differential equation for misreport detection. Luo *et al.* [12] built a fuzzy logic model to deal with the uncertainty and tolerance of imprecise data inputs. These two models are complex and based on only partial reputation information that may not be quite accurate. Akbani *et al.* [13] proposed to build support vector machine models against different types of malicious behaviors offline, and then upload the models to the nodes in the network to classify malicious nodes and normal nodes. Wang *et al.* [14] proposed to use a number of statistical testing techniques such as expectation maximization algorithm, Kalman aggregation, and hypothesis test to defend against malicious and coordinated feedbacks. However, all these methods use complex machine learning and statistical algorithms for selfish and misbehaving node detection, which generates high computational costs of the mobile nodes. Also, the periodical exchanges of the observations lead to high communication overhead. Taking advantage of the DHT structure, ARM efficiently and more accurately calculated the global reputation of the nodes with low overhead. By measuring the deviation of the reputation ratings among average ratings, the selfish and misbehaving nodes can be easily detected with low computational costs. There are many other reputation system works that specifically deter misreporting, fake IDs, and free-riding as presented in our previous work [29]. ARM only focuses on excluding the aforementioned falsified, conspiratorial and misreported information when calculating node global reputation values. Its lightweight method can compensate those works to strengthen their capability in achieving the goals.

Price systems [15]–[21] provide incentives for cooperation by using micro payments. Buttyan *et al.* [15]–[17] proposed two payment models: the packet purse model, in which a source node pays relay nodes by storing virtual credits in the packet head, and the packet trade model, in which a relay node buys packets from the previous node and sells 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.* [19] proposed a traffic price approach, in which the compensation for message forwarding depends not only on the energy consumption of the transmission, but also on the congestion level of the relaying node. Janzadeh *et al.* [21] proposed a price-based cooperation mechanism that utilizes hash chains to defend against cheating behavior such as requiring credits for fake service requests and denying service after receiving credits. A number of works have been proposed to enhance the cooperation between nodes based on game theory [20], [22], [30]–[32]. Since these works focus on theoretical algorithm design, they did not provide details on the system design. ARM focuses on the reputation system design and aims to enhance the system efficiency and effectiveness by coordinately integrating the reputation system and price system

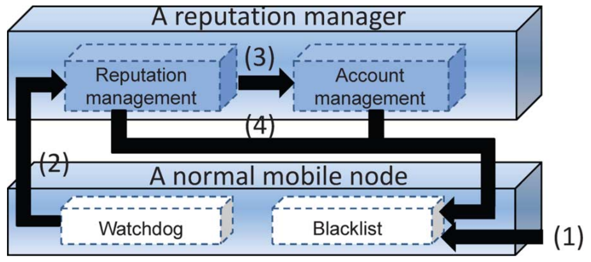


Fig. 1. Overview of the ARM system.

through a DHT-based infrastructure. Unlike current price systems that circulate credits in the network, ARM uses an account management mechanism that transparently processes the credit accounts of the nodes based on the node transactions.

III. Design of the ARM Management

A. Overview

ARM selects a number of trustworthy and low-mobility nodes as reputation managers. The reputation managers constitute a locality-aware DHT, functioning as a backbone at the center of the MANET for efficient operations of ARM. As shown in Fig. 1, each normal mobile node has a *watchdog* [3], [4] to monitor and report the behaviors of its neighbors to managers. Here, we assume that a rational user is willing to conduct neighborhood monitoring as it is willing to periodically exchange reputation information in previous reputation systems, aiming to create a trustworthy network environment. The DHT helps to marshal all reputation and transaction information in the system of a given node to a specific manager. The managers have two functions: reputation management and account management. Each manager calculates the reputations and increases/decreases the credits in the accounts of the mobile nodes for which it is responsible. Nodes with reputations below the threshold or deficit accounts are regarded as uncooperative nodes. Managers notify mobile nodes about uncooperative nodes, which are then put into blacklists. The blacklisted nodes' forwarding requests are ignored by others. Like price systems, ARM also requires that the source node pays the relay nodes for packet forwarding, but it eliminates the need for credit circulation in the network. Moreover, in ARM, a high-reputed node pays less credits while a low-reputed node pays more credits in a forwarding service transaction, thus effectively providing incentives for cooperation between nodes.

We use an example to briefly explain the operation process of ARM. When node n_1 looks for a path for packet transmissions, it broadcasts a path query message to the packet destination. When nodes n_2 and n_3 receive the query, they check whether n_1 is on their blacklists [step (1) in Fig. 1]. If so, they ignore n_1's query. Otherwise, they respond to n_1. n_1 then forwards the packet along a discovered path consisting of cooperative nodes including n_2 and n_3. In step (2), the neighbor nodes of communicating nodes n_2 and n_3 monitor the data transmission using their *watchdog* and report the observed transmission rate to their closest managers. Relying on the DHT, the managers merge all reputation reports about n_2 and n_3, respectively, and produce their global reputations. The DHT overlay supports efficient reputation information collection and querying. In step (3), ARM adds credits to the accounts of n_2 and n_3 and decreases the account of n_1. According to the reputation-adaptive account management in ARM, higher reputation leads to more earned credits for service suppliers (n_2 and n_3) and lower service charges for service receivers (n_1). In step (4), if the reputations of n_2 and n_3 are below a threshold or n_1 has a deficit account, managers inform all nodes in the network to put these uncooperative nodes on their blacklists.

B. Assumptions of the Network

In this paper, we consider the scenario of MANETs with normal or relatively large network size and area size that have no centralized dedicated servers in the network. Therefore, we form trustable and relatively stable nodes into a DHT for efficient reputation management. We use the DHT infrastructure due to two reasons: 1) given a node's ID, DHT's `Insert(ID, object)` function can forward the reputation or transaction reports on this node to its manager, and DHT's `Lookup(ID)` function can forward the reputation or account query to this node's manager; and 2) these two functions have $O(\log N)$ time complexity and each node maintains $O(\log N)$ neighbors, which achieves high scalability. Without the DHT infrastructure, each manager must maintain a record for each node's mapped manager, which is not scalable in a large-scale MANET.

Our proposed mechanisms can be directly used in a wireless hybrid network that integrates dedicated servers (base stations) with a MANET for reputation management. The widely used smartphones usually have dual-mode interfaces: a low-power ad hoc network interface (e.g., IEEE 802.11 interface, WLAN radio interface) that has short transmission range with high data transmission rate and a high-power infrastructure network interface (e.g., cellular interface, WiMAX interface) that has long transmission range with low data transmission rate. Thus, we assume some mobile nodes in the MANET have dual-mode interfaces.

The network designers can initially deploy a number of peers in the network, serving as bootstrap manager nodes for DHT construction. These nodes can be considered as trustworthy. As more nodes join in the network, the nodes with high reputations and low mobility can be gradually selected as reputation manager and added into the DHT. The selected reputation managers can be rewarded with virtual credits to compensate their contribution and provide incentives for nodes to become the reputation managers. The reputation messages exchanged between managers are of small size and delay-tolerant compared to data messages. Managers use the low-power interface for data transmission. For reputation data transmission, they can use the high-power interface through cellular/WiMAX infrastructure [33] or use multihop *ad hoc* transmission mode [34], [35].

C. Locality-Aware DHT Infrastructure

Fig. 2 illustrates the hierarchical structure of ARM. The higher level is a DHT network composed of managers, and the lower level is composed of normal mobile nodes. A DHT partitions ownership of a set of objects among participating nodes and efficiently route messages to the unique owner of any given object. Each object or node is assigned an ID that

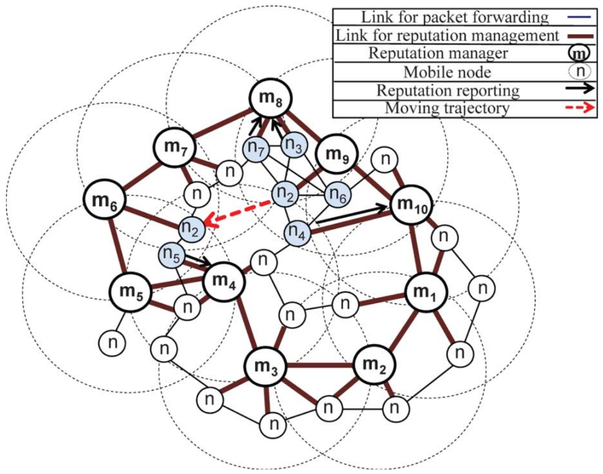


Fig. 2. ARM hierarchical structure.

is the hashed value of the object or node IP address using a consistent hash function [36]. An object is stored in a node whose ID equals or immediately succeeds 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.

We leverage the Chord DHT [23] as the infrastructure of ARM for scalable and efficient reputation and account management. ARM constructs a locality-aware DHT where logical proximity matches the physical proximity in reality. In this way, the packet routing path in the overlay is consistent with 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, while neighbor nodes in the DHT overlay network can always communicate with each other. Two questions naturally arise. First, is it possible to form managers to a locality-aware DHT infrastructure in a MANET? Second, how is a DHT built and maintained in a mobile environment?

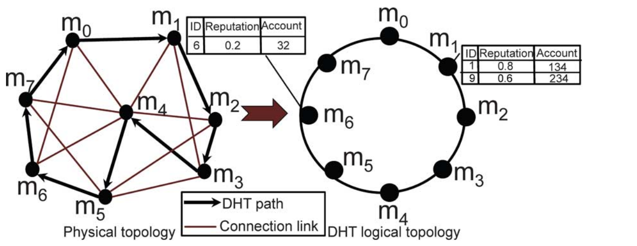
Ring-topology-based Chord is actually a Hamiltonian cycle because successor neighbor links connect all nodes to a circle. Therefore, to build a locality-aware DHT, the physical topology should also be a Hamiltonian cycle [37]. As the work in [38], we assume that the movement of each node is independent and identically distributed (i.i.d.) in a square area with space length l and derive the following proposition.

Proposition 3.1: If the transmission range of nodes satisfies $r \geq (2/\sqrt{2\pi})l$, a Hamiltonian cycle can be formed to build a Chord DHT.

Proof: In order to guarantee that the nodes in a graph can form a Hamiltonian cycle, the number of neighbors of each node (i.e., connectivity degree) should satisfy $\geq (N/2)$ [37], where N $(N \geq 3)$ is the total number of nodes. With the assumption of the i.i.d. movement, a node has the least connectivity degree when it moves to the corner of the square field. That is

$$\frac{\pi r^2}{4l^2}N \geq \frac{N}{2} \Longrightarrow l \leq \frac{\sqrt{2\pi}}{2}r \Longrightarrow r \geq \frac{2}{\sqrt{2\pi}}l. \qquad (1)$$

□



ID	Reputation	Account
6	0.2	32

ID	Reputation	Account
1	0.8	134
9	0.6	234

Fig. 3. Construction of the DHT infrastructure.

1) Locality-Aware DHT Infrastructure Construction: Fig. 3 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, we assign a sequence of consecutive 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 Fig. 3, we first choose a trustworthy bootstrap manager (m_0) and assign it ID 0. Then, it chooses its physically closest node as its successor and assigns it ID 1. 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 $N-1=7$ 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 a DHT neighbor determination protocol using broadcasting.

2) Locality-Aware DHT Infrastructure Maintenance:

Proposition 3.2: In ARM, with the i.i.d. node movement assumption, the average time period for a pair of neighbor managers to stay in the transmission range of each other (i.e., connection duration) is $r/\overline{v}$, where $\overline{v}$ is the average relative speed of their movement.

Proof: Since the movement of each manager is i.i.d., if manager m_i is distance d away from manager m_j, the expected time period needed by m_i to move out of the transmission range of m_j is

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

□

Proposition 3.2 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 a node departures followed by a node joins, 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.

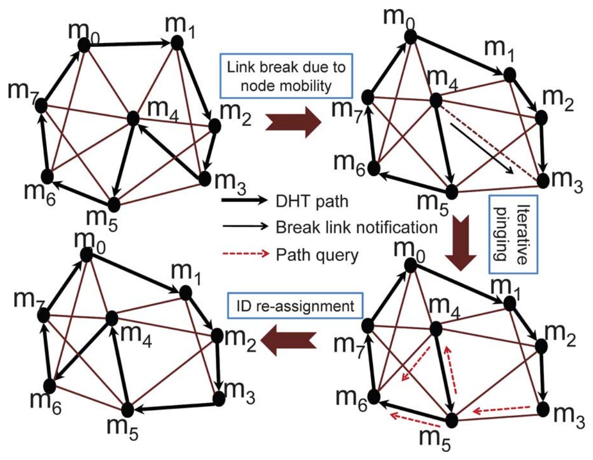


Fig. 4. Maintenance of the DHT infrastructure.

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 $(k > i)$ and covers all managers with IDs $\in [i-1, k]$ in order to maintain a complete DHT circle covering all managers. 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} $(j \geq 1)$ sequentially by locally broadcasting a query message with $\text{TTL} = j$. That is, manager m_{i+2} is pinged first, then m_{i+3} is pinged, and so on. Each pinged manager replies to m_{i-1} with 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 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 Fig. 4, 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 $\in [3, k]$ $(k > 4)$ 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 are assigned with new consecutive IDs for a complete circle. After finding a new path that travels through manager m_i with ID 5 and m_j with ID 4, m_3 assigns m_i and m_j ID 4 and 5, respectively.

3) 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. Relying on the `Insert(i,B+R)`, the managers marshal all the information of n_i in the system into n_i's *owner manager*. The owner manager calculates n_i's reputation and increases/decreases the credits in its account. For example, as shown in the table of node m_1 in Fig. 3, the reputation and account information of n_1 and n_9 is stored in m_1, which is responsible for their reputation 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 the DHT routing algorithm.

D. 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 for two reasons: It uses the global information rather than local partial information in reputation calculation, and the large amount of global information makes it effective in detecting falsified, conspiratorial, and misreported information through deviation.

ARM uses neighbor monitoring to observe the packet-forwarding behaviors of nodes. Specifically, each observer uses a *watchdog* [3], [4] 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. Observing node n_{o} calculates the observed reputation value of node n_i by $R_i^{n_{\mathrm{o}}} = D_i^{\mathrm{r}}/D_i^{\mathrm{f}}$ and reports it to its closest manager. The manager then merges the collected reputations reported by the nodes in its transmission range to local reputation R_{l_i}.

Misreports Avoidance: When a node in a region experiences an adverse network condition such as background interference due to traffic or thermal noise, the node's neighbors may also experience the adverse network conditions. Then, even though the nodes are cooperative, they are unable to transmit requested data. As a result, these nodes that mutually monitor each other report low R_{l} values for others. In this case, low R_{l} values are reported from the nodes that are clustered together. ARM can easily solve this problem as all reputations of each node in a region are reported to one manager. When a manager notices that all nodes in an area report low observed reputations, it temporarily ignores the reports to reduce the uncertainty of the reported information in order to avoid punishing nodes for failing 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 always deviate largely from most reputations. Thus, in order to reduce the effect of falsified reports, a manager filters the R_i's that dramatically deviate from the average R_i. The deviation of $R_i^{n_{\mathrm{o}}}$ reported by node n_{o} about node n_i is calculated as

$$\Delta R_i^{n_{\mathrm{o}}} = \left| R_i^{n_{\mathrm{o}}} - \sum_{n_j \in \mathbf{n}} R_i^{n_j} / |\mathbf{n}| \right| \tag{3}$$

where $\mathbf{n}$ denotes the group of observers that report R_i to the manager during T, and $|\mathbf{n}|$ denotes the number of nodes in the group. ARM sets a threshold δ_{l} for the deviation and ignores $R_i^{n_{\mathrm{o}}}$ satisfying $\Delta R_i^{n_{\mathrm{o}}} > \delta_{\mathrm{l}}$. δ_{l} is determined based on practical rating values. If the values differ greatly, δ_{l} should be set to a larger value. The manager m_o then calculates the local reputation value of n_i in T denoted by $R_{\mathrm{l}_i}^{m_o}$

$$R_{\mathrm{l}_i}^{m_o} = \sum_{n_o \in \tilde{\mathbf{n}}} R_i^{n_o} / |\tilde{\mathbf{n}}| \quad (4)$$

where $\tilde{\mathbf{n}}$ denotes $\mathbf{n}$ after removing the deviated observed reputations. Then, the manager reports $R_{\mathrm{l}_i}^{m_o}$ to n_i's owner manager using $\mathtt{Insert(i,} R_{\mathrm{l}_i}^{m_o}\mathtt{)}$. According to (3), the expected value of δ is

$$E(\delta) = \left| \frac{a \cdot \overline{R}_{\mathrm{l_h}} + b \cdot \overline{R}_{\mathrm{l_f}}}{a+b} - \overline{R}_{\mathrm{l_f}} \right| = \frac{a\left(\overline{R}_{\mathrm{l_h}} - \overline{R}_{\mathrm{l_f}}\right)}{a+b} \quad (5)$$

where $\overline{R}_{\mathrm{l_h}}$ and $\overline{R}_{\mathrm{l_f}}$ denote the expected values of honest reports and false reports, and a and b respectively denote the number of honest reports and the number of 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 (3). This problem can be resolved by another filtering process at the owner manager m_i that collects all $R_{\mathrm{l}_i}^{m_o}$ from different managers m_o. Again, m_i computes the variance of $R_{\mathrm{l}_i}^{m_o}$ based on (6) and ignores $R_{\mathrm{l}_i}^{m_o}$ with $\Delta R_{\mathrm{l}_i}^{m_o} > \delta_{\mathrm{g}}$. δ_{g} can be determined in the same way as δ_{l}

$$\Delta R_{\mathrm{l}_i}^{m_o} = \left| R_{\mathrm{l}_i}^{m_o} - \sum_{m_j \in \mathbf{m}} R_{\mathrm{l}_i}^{m_j} / |\mathbf{m}| \right| \quad (6)$$

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

$$R_{\mathrm{g}_i} = \sum_{m_o \in \tilde{\mathbf{m}}} R_{\mathrm{l}_i}^{m_o} / |\tilde{\mathbf{m}}| \quad (7)$$

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

The colluders may be in the transmission range of different managers. In this case, if the colluders in the transmission range of the manager do not constitute the majority of the reporting nodes, the reported information from colluders is filtered out by the manager according to (3). Otherwise, the falsified information from colluders is filtered out according to (6) by the owner manager.

For example, in Fig. 2, nodes n_3, n_4, and n_7 monitor the transmissions of n_2. Nodes n_3 and n_7 report the 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 $R_{\mathrm{l}_2}^{m_8}$ and $R_{\mathrm{l}_2}^{m_{10}}$, and report the results to n_2's owner manager, m_2. Later, when n_2 moves close to n_5, n_5 starts to monitor the transmissions of n_2 and reports the observed reputation of n_2 to its nearby manager m_4, which subsequently reports $R_{\mathrm{l}_2}^{m_4}$ 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 a more accurate reflection of n_i's trustworthiness. Also, global information (i.e., large data samples) makes it easy to precisely detect false information.

When a node is out of power or suffers from channel congestion, it cannot offer service to others and thus has a low reputation though it is cooperative. It is unfair to punish such a cooperative node with a low reputation. However, it is difficult to identify the real reason for a low reputation. Therefore, like the previous work [7], ARM takes into account the old reputation when calculating the new reputation. That is

$$R_{\mathrm{g}}^{\mathrm{new}} = \alpha R_{\mathrm{g}}^{\mathrm{old}} + (1-\alpha) R_{\mathrm{g}} (\alpha < 0) \quad (8)$$

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 monitoring region and $\bar{C}$ is the expected channel capacity of the monitoring region.

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

$$R_{\mathrm{g}}^{\mathrm{new}} := \varphi R_{\mathrm{g}}^{\mathrm{new}} (\varphi < 1) \quad (9)$$

where $R_{\mathrm{g}}^{\max}$ denotes the maximum global reputation, and β and φ are weight factors. β is determined by the consideration weight of $\mathcal{T}$ and $R_{\mathrm{g}}^{\max}$, and φ is determined by the decrease speed of the reputation of high-reputed nodes.

The rationale behind this policy is that the reputation of a high-reputed node decreases over time if it does not receive a new rating from others. The low reputation subsequently increases the service price for message forwarding of the node (Section III-F). Therefore, the only way a node can enjoy a low price is to cooperate with other nodes all the time. In ARM, those selfish nodes that drop packets while keeping $R_{\mathrm{g}} \geq \mathcal{T}$ can 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 a selfish node. However, how to set the value of threshold $\mathcal{T}$ depends on the system environment. In a system with high background noise, we can set $\mathcal{T}$ to a low value since a high $\mathcal{T}$ value may lead to a number of normal nodes mistakenly regarded as selfish nodes.

E. Distributed Reputation Manager Auditing

A compromised reputation manager may modify nodes' reputation values and (or) account values in two situations. First, a reputation manager misreports the reputation of a node to its owner manager in the local reputation calculation. Second, the owner manager of a node modifies the reputation value and account value of the node in the global reputation calculation.

In the first situation, since the nodes in the transmission range of a reputation manager always change, the local reputation values of a node can be collected by several reputation managers in an interval T. After these managers report the collected reputation values of a node to its owner manager, the owner manager can detect the misbehaviors of the malevolent reputation mangers using the collusion avoidance method introduced in Section III-D.

In the second situation, as the owner reputation manager of a node calculates its final reputation value and manages its account value, if the manager modifies the reputation value, no other nodes can detect it. To handle this problem, we use redundant reputation managers for each node. Specifically, we set c different consistent hash functions. When a manager reports the local reputation of a node to its owner managers, it uses the c consistent hash functions to generate c virtual IDs. Then, it uses `Insert(id,B+R)` to report the reputation to the owner managers of the node. When a node inquires the reputation value of node n_i from reputation managers, it also uses the c consistent hash functions on n_i's IP address to generate c virtual IDs. Then, it executes `Lookup(id)` to retrieve the values. The node first calculates the average of the c returned values. The reputation managers whose returned reputation values deviate the average value for a certain threshold $\delta_{\rm a}$ are considered as malevolent managers. $\delta_{\rm a}$ is set to an appropriate value based on the deviation of reported values in practice. A larger $\delta_{\rm a}$ may lead to false negatives while a smaller $\delta_{\rm a}$ may lead to false positives. Then, the node regards the average value of the reputation values from the nonmalevolent managers as n_i's global reputation value. The node also reports the suspicious malevolent manager to other $c-1$ managers. The managers periodically exchange their received misbehavior reports and dismiss the manager who has been reported as a malevolent manager after checking the reputation values managed by the manager by executing `Lookup(id)`.

In this case, a high-reputed node is selected to join in DHT to replace the dismissed manager. To select a new reputation manager, manager m_0 first finds the node i with the highest reputation among the normal nodes it managers and then transfers a token, $\mathrm{TOKEN}(\max R_{\rm g}, \mathrm{ID}_i)$, to its successor manager m_1. If m_1 has a normal node with $R_{\rm g}$ higher than $\max R_{\rm g}$ in the token, it updates $\max R_{\rm g}$ and ID_i in the token and passes the token to its successor manager m_2. This process continues until manager m_0 receives the token. Then, m_0 informs the node with reputation value $\max R_{\rm g}$ to be the reputation manager. The locality-aware DHT infrastructure maintenance algorithm in Section III-C.3 maintains the locality of the DHT infrastructure.

In the redundant reputation manager method, there is a tradeoff between the overhead and reliability of reputation management. More resource managers for a node lead to higher reliability but higher overhead. The number of resource managers for a node should be determined by the probability that resource managers are compromised or malicious. The number can be small if the probability is low.

F. 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, as opposed to previous price systems, ARM's account management does not need credit circulation in the network, reducing transmission overhead and system complexity.

In previous price systems [15], [17], the credits a node earns or pays equal the product of the unit price and the absolute number of packets forwarded (*absolute method* in short). Cooperative nodes in a region with low traffic may not earn enough credits for their transmission needs, and nodes in a region with high traffic or without many transmission service needs can be uncooperative without being punished. To deal with these problems, rather than relying on the absolute number, ARM determines the credits earned by a node based on the percent of forwarded packets among its received packets. Notice that $R_{\rm g}$ is exactly the percentage in ARM; We use it directly for the calculation of earned credits. Specifically, node n_i's owner manager increases its account every T by

$$P_{\rm e} = p_{\rm r} R_{{\rm g}_i} \tag{10}$$

where $p_{\rm r}$ is a constant credit rewarding factor. We call this method the *relative method*. The relative method brings about two advantages. First, managers can directly use the latest reported reputation for account calculation instead of taking extra efforts to record packet forwarding activities between nodes, reducing transmission overhead. Second, it awards nodes fairly according to the cooperative degree of node behaviors.

Proposition 3.3: For cooperative behavior rewarding, the relative method provides fairer treatment to nodes than the absolute method.

Proof: We use $q_{\rm l}$ and $q_{\rm h}$ ($q_{\rm h} > q_{\rm l}$) to denote the percent of the time period T used for packet transmissions in a relay node in low-traffic and high-traffic regions, respectively. In the absolute method, we use $p_{\rm a}$ to denote the amount of awarded credits per packet. Suppose λ is the average packet generation rate of the source; during time period t, the cooperative relay node gains $(q_{\rm h} - q_{\rm l}) t p_{\rm a} \cdot \lambda R_{\rm g}$ more credits in the high-traffic region than in the low-traffic region. Using the relative method, whether the relay node is in a low-traffic region or a high-traffic region, it always earns $(t/T) p_{\rm r} R_{\rm g}$. □

In order to foster the cooperation incentives, ARM connects the forwarding service cost per packet $p_{\rm c}$ of a node to its reputation, so that higher-reputed nodes receive more credits while lower-reputed nodes receive fewer credits for offering the same forwarding service. The $p_{\rm c}$ of n_i, denoted by $p_{{\rm c}_i}$, is calculated by

$$p_{{\rm c}_i} = \frac{\gamma}{R_{{\rm g}_i}^{\rm new}} \tag{11}$$

where γ is a weight. It can be the unit service price in the price systems. Thus, higher reputation of a node leads to lower price cost.

When an observing node $n_{\rm o}$ 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 the DHT function $\mathtt{Insert}(\mathtt{i}, B_i + R_i)$, 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 $t_0 + kT$ $(k = 1, 2, \ldots)$ is

$$A(t) = A(0) - \sum_{t=t_0}^{t_0+kT} \left(p_{c_i}(t) \cdot N_{p_i}(t) - p_r \cdot R_{g_i}\right). \tag{12}$$

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

Proposition 3.4: ARM exponentially increases the credits of a node while it is cooperative, and exponentially decreases the credits of a node while it is uncooperative.

Proof: A node's R_g stays approximately constant during the time period it is cooperative or uncooperative. We use $R_g(t)$ to denote the reputation of a node at an arbitrary time instance $t = kT$ $(k \in [0, 1, \ldots, m])$ during time period mT. $R_g(t+T)$ and $R_g(t)$ correspond to R_g^{new} and R_g^{old} in the $(k+1)$th time period. From (8), we can determine that

$$R_g(t+T) - R_g = \alpha \cdot (R_g(t) - R_g) \tag{13}$$

$$\Rightarrow R_g(t) = \alpha^{\frac{t}{T}} (R_g(0) - R_g) + R_g. \tag{14}$$

Based on (11) and (12), after time t, a node's account is

$$\begin{aligned} A &= A(0) - \sum_{t=T}^{kT} (p_c(t) \cdot N_p - p_r R_g) \\ &> A(0) - \int_{T}^{kT+T} (p_c(t) \cdot N_p - p_r R_g) \cdot \mathbf{d}(t) \\ &= \begin{cases} A(0) - \frac{\gamma N_p (1-\alpha^{-k}) \cdot T}{(R_g - R_g(0)) \cdot \ln \alpha \cdot \alpha} \\ \quad + p_r \cdot T \cdot R_g \cdot k, & \text{if } R_g \neq R_g(0) \\ A(0) - \frac{\gamma N_p \cdot k \cdot T}{R_g} + p_r \cdot T \cdot R_g \cdot k, & \text{if } R_g = R_g(0). \end{cases} \end{aligned}$$

Because $\alpha^{-k} > 1$ and $\ln \alpha < 0$, when $R_g < R_g(0)$, the account exponentially decreases with k; when $R_g > R_g(0)$, the account exponentially increases with k; and when $R_g = R_g(0)$, the account decreases linearly with k. □

From Proposition 3.4, we can arrive that in order to ensure a selfish node will finally run out of the credits if it manipulates its reputation just above the threshold $T_R = R_g$, we need ensure

$$\gamma > \begin{cases} \frac{(R_g - R_g(0)) \ln \alpha \cdot \alpha \cdot (A(0) + T k p_r R_g)}{(1-\alpha^{-k}) N_p T}, & \text{if } R_g \neq R_g(0) \\ \frac{(A(0) + T p_r R_g k) \cdot R_g}{T N_p k}, & \text{if } R_g = R_g(0). \end{cases} \tag{15}$$

IV. Performance Evaluation

We conducted simulations on NS-2 [39] to demonstrate the performance of ARM. We used the Distributed Coordination Function (DCF) of IEEE 802.11 as the MAC-layer protocol. We chose the two-ray propagation model as the physical-layer model, and the constant bit rate as the traffic mode. 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 250 and 1000 m, respectively. The raw physical link bandwidth was set to 2 Mb/s. The heights of antennas for data transmitting and receiving were set to 1.5 m. We used the random way-point mobility model [40] to generate node movement. The nodes are i.i.d. deployed in the field. They move at a speed 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 every 40 s. The range of the reputations was set to [0, 1], and the reputation threshold $\mathcal{T} = 0.4$. The deviation thresholds are set as $\delta_l = \delta_g = \delta_a = 0.2$. Each simulation lasted 5000 s. We ran 10 simulations and reported the average as the experiment results.

We set $\alpha = 0.7$ in (8), $\varphi = 0.5$ in (9), $p_r = 2$ in (10), and $\gamma = 1$ in (11). The time period T for periodical reputation exchange/report between mobile nodes and to managers was set to 10 and 50 s, respectively. Each node initially was assigned 5000 credits and a reputation value of 1. We compared the performance of the DSR routing algorithm [35] in a defenseless MANET with neither reputation system nor price system (*Defenseless*), in ARM, in a reputation system (Reputation) [5], [9], and in a price system (Price) [15], [21]. We used the basic reputation management mechanism in [5] and [9] and the basic price management mechanism in [15] and [21] in the experiments. We chose these works for comparison because they have the representative mechanisms for reputation systems and price systems, respectively. To make the results comparable, rather than using the absolute number of forwarded packets, we use $R_l = D^r/D^f$ 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 on its blacklist for data forwarding. By default, every node just has one reputation manager. Like previous reputation and prices systems, we assume that each node in the network is rational.

A. Performance Comparison of Different Systems

Higher effectiveness of cooperation incentives leads to more cooperative nodes, hence higher system throughput. This experiment measures the system throughput with a certain fraction of selfish nodes. We configured selfish nodes that manipulate their reputations just above the reputation threshold. Fig. 5(a) plots the average system throughput of different systems versus the fraction of selfish nodes. The figure shows that ARM generates a higher throughput than *Price* and *Reputation*, which produce 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 *Price*, the selfish nodes will finally run out of credits to pay the service and are isolated from the network, resulting in a higher throughput than *Reputation*. Similarly in ARM, the selfish nodes eventually do not have enough credits to pay for their transmission services and are put on the blacklist. Since the credits of the selfish nodes in ARM decrease much faster than in *Price*, ARM produces a higher throughput than *Price*. Also, the figure shows the throughput of the systems decreases as selfish nodes grow. Since *Defenseless* and *Reputation* cannot detect all selfish nodes, their throughput decreases as the fraction of selfish nodes grows. The reason why *Price* and ARM also exhibit performance degradation even though they can detect most selfish nodes is because

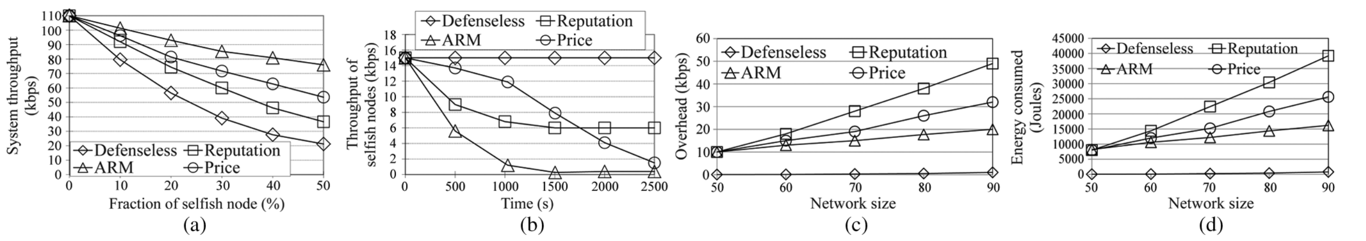


Fig. 5. Performance comparison between different systems. (a) Average system throughput. (b) Throughput of selfish nodes. (c) System overhead. (d) Energy consumption.

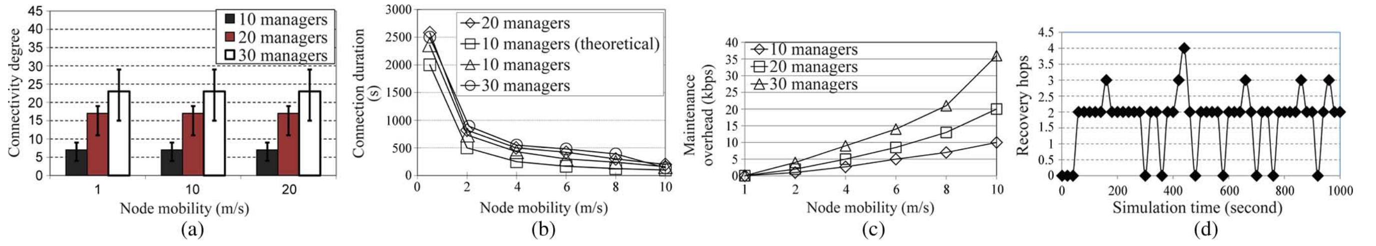


Fig. 6. Performance of the locality-aware DHT infrastructure in ARM. (a) Connectivity degree per manager. (b) Average connection duration. (c) Topology maintenance overhead. (d) Reassigned IDs in DHT maintenance.

selfish nodes may be chosen as forwarding nodes before their credits are used up. Also, avoiding selfish nodes in routing leads to a longer path length, which suffers from a higher transmission interference.

Higher effectiveness of cooperation incentives in deterring node selfish behaviors and misbehaviors leads to less throughput of selfish nodes. 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 set up 10 selfish nodes and used them as source nodes. Fig. 5(b) plots the throughput of the selfish nodes. In *Defenseless*, selfish nodes keep a constant throughput of 15 kb/s. In *Reputation*, the throughput decreases as time elapses and then stays constant at 6 kb/s. This is because the selfish nodes keep R_g just above $\mathcal{T}$; thus, their transmission requests are accepted by other nodes. The throughput of *Price* and ARM decline sharply as time goes on. This is because the selfish nodes running out of the credits are isolated from the network. We also see that the throughput of selfish nodes in *Reputation* decreases much slower than ARM. This means that with the aid of account management, ARM can effectively detect and punish all selfish nodes, excluding them from the network.

To evaluate the efficiency of the systems, we tested the overhead measured in kilobits per second for all overhead messages in the systems. In addition to the "hello" messages, the overhead messages in ARM include those for topology construction, maintenance, and reputation report and querying. In *Reputation*, it also includes the messages for reputation exchange. In *Price*, it also includes the messages for credit payments. Fig. 5(c) illustrates the overhead in each system versus network size. We see 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 proportional 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*, each node periodically exchanges reputation information between its neighbors, then the reputation information of each node is flooded throughout the network, resulting in higher overhead. In *Price*, credit circulation in the network generates transmission overhead. *Defenceless* has the smallest amount of overhead as it does not have any cooperation incentive mechanism. Fig. 5(d) shows the amount of energy consumed by communication overhead during the experiment. Based on [41], we assumed that the energy consumption overhead is 10.50 J/packet using the high-power interface and is 1.76 J/packet using the low-power interface. Though high-power interface consumes more energy, ARM still consumes much less energy than *Reputation* and *Price* because it reduces message exchanges among nodes. This result demonstrates that ARM is more energy-efficient than other methods.

B. Evaluation of the DHT Infrastructure in ARM

Proposition 3.1 indicates the condition of building a locality-aware DHT. In this experiment scenario, the condition of $r \geq (2/\sqrt{2\pi})l \approx 1000$ m is satisfied. We measured the average, maximum, and minimum connectivity degree per manager while the managers move at the speeds of 1, 10, and 20 m/s. Fig. 6(a) shows that the smallest connectivity degree of a manager is about $N/2$. This result verifies Proposition 3.1. The figure also shows that more managers incur a 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 DHT maintenance mechanism can establish new links immediately upon link breakups.

We use *connection duration* to denote the time period that a pair of neighbor managers stay in the transmission range of each other. Fig. 6(b) presents the average connection duration of managers versus node mobility. We also include the theoretical results based on Proposition 3.2 in the case of "10 managers." The figure demonstrates that when the mobility is 0.5 m/s, the DHT is much more stable than other situations. As node mobility increases, the average connection duration drops

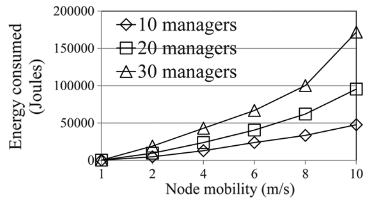


Fig. 7. Energy consumption for DHT maintenance in ARM.

sharply. We also find that the connection duration stays almost constant for different numbers 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 the DHT. This feature enables us to build an ARM with high scalability without significantly compromising the DHT stability. The simulation results closely match the theoretical result. The small gap is caused by the fact that the theoretical analysis does not consider node pause time during movement.

Fig. 6(c) shows the maintenance overhead of the DHT infrastructure versus node mobility. The overhead is measured by the number of messages exchanged for DHT maintenance. The overhead grows with increasing node mobility. Higher mobility leads to a higher probability of link breakups, incurring a 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.

In DHT maintenance, when n_i's link to n_j breaks, n_i needs to find a new path to n_j and reassign IDs to the nodes in the path. Fig. 6(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. The neighbors in the DHT of ARM are also the neighbors in physical topology. Although the physical link between nodes n_i and n_j is broken, they still share the same manager neighbors. Therefore, by reordering the IDs of n_i and n_j and the shared neighbors, the DHT structure with numerically continuous IDs can be recovered most of the time. Even if such neighbors do not exist, as a Hamiltonian cycle can always be found among managers, we can always rebuild a new DHT among managers. Fig. 7 shows the amount of energy consumed for DHT maintenance based on Fig. 6(c). We see that more managers consume more energy since the number of neighbors of each manager increases.

C. Performance of Reputation-Adaptive Account Management

In this experiment, all nodes are static and cooperative. We equally divide the simulation region into two parts. One part is a subregion with high traffic, and the other part is a subregion with low traffic. A source node only selects the nodes that are in the same region as the destination node. We randomly and periodically selected seven source nodes in the high-traffic region and two source nodes in the low-traffic region. The nodes kept sending messages to the destinations for 100 s.

Fig. 8(a) shows the awarded credits each node has using the absolute method and the relative method, respectively. We see that using the relative method, all nodes have almost the same amount of credits regardless of the region the nodes stay. In contrast, using the absolute method, the credits of the nodes have deviated values. Recall that in the absolute method, the credits awarded to a node are based on the number of packets it has forwarded. Therefore, nodes in the high-traffic region gain more credits than nodes in the low-traffic region. In the relative method, the credits of a node are awarded based on their packet forwarding rate. Since all nodes are cooperative, their forwarding rate should be almost the same. The slight difference is caused by the communication interference that leads to some packet drops of the cooperative nodes. Therefore, the relative method is much fairer than the absolute method as it awards all cooperative nodes based on their cooperative behavior rather than the traffic through them.

Fig. 8(b) and (c) shows the awarded credits of the nodes in the system with 5 and 10 selfish nodes in the high-traffic region, respectively. The forwarding ratings of the selfish nodes are randomly selected from [0.3, 0.4]. We plot a black line to show the number of nodes whose credits are below 100. The figures show that with the relative method, there are five and 10 nodes, respectively, whose awarded credits are less than 100. All of these nodes are selfish nodes, which means that the selfish nodes can be accurately detected using the relative method. The selfish nodes with the relative method cannot gain a high amount of awarded credits as their packet forwarding rate is low. However, with the absolute method, the number of nodes whose awarded credits are below 100 is seven and six, respectively. That is, as the number of selfish nodes in the system increases, the number of nodes whose credits are below 100 does not increase significantly. This is because the selfish nodes in the absolute method can still gain a large amount of credit as they receive a high amount of service requests. The nodes whose credits are less than 100 in absolute methods are the nodes in the low-traffic region.

Fig. 9 shows the account credits of a randomly selected node over the simulation time. During the simulation time, we let the node be cooperative during 0–2000 s and be uncooperative thereafter. We see that the account value of nodes increases exponentially and also decreases exponentially. The exponentially increase/decrease rate prevents the system from being too sensitive to the packet droppings, which may be caused by interference. Therefore, if an uncooperative node keeps dropping packets, its credits decrease exponentially. If a cooperative node keeps forwarding packets, its credits increase exponentially. We can also see that with a large α value, the credit increases/decreases more sharply. Therefore, we can adjust the account increase/decrease rate by adjusting the α value.

D. Performance in False Accusation Resilience

Misreporting nodes are cooperative nodes that deliberately report low observed reputation values for their neighbors to managers. In this experiment, all nodes are cooperative. We chose some nodes that deliberately evaluate their neighbors with low reputations randomly chosen in [0.3, 0.4]. Other nodes give their neighbors reputations randomly chosen from [0.9, 1] considering possible interference in transmission. In the defenseless system, every node rates its neighbors based on their forwarding behavior. Fig. 10(a) and (b) plots all evaluated local reputations of each node in a defenseless system with five and

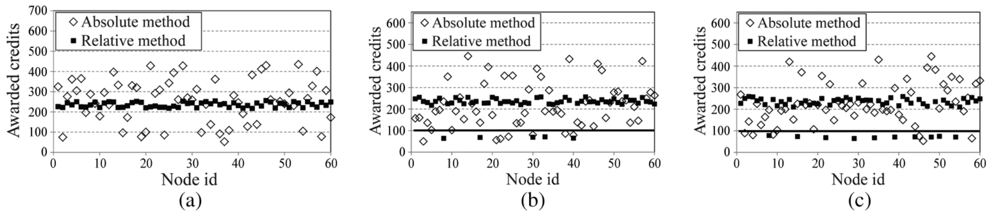


Fig. 8. Evaluation of node account value versus traffic load. (a) No selfish nodes. (b) Five selfish nodes. (c) Ten selfish nodes.

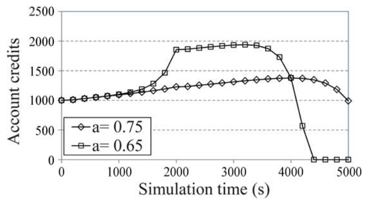


Fig. 9. Credits versus simulation time.

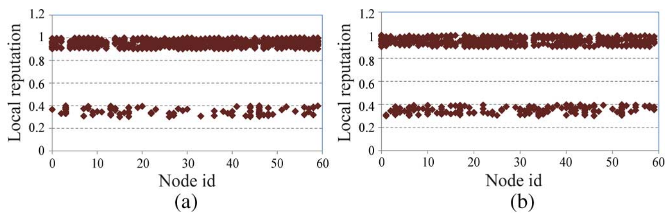


Fig. 12. Reputations in defenseless system with nongroup collusion. (a) Defenseless system (five colluders). (b) Defenseless system (10 colluders).

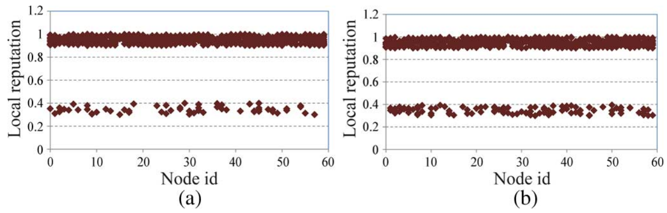


Fig. 10. Local reputations in defenseless system. (a) Five false-reporting nodes. (b) Ten false-reporting nodes.

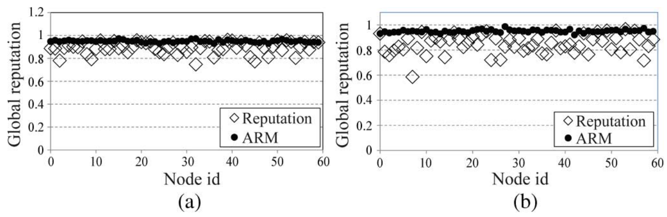


Fig. 11. Node reputations in defensive system. (a) Five false-reporting nodes. (b) Ten false-reporting nodes.

10 false-reporting nodes, respectively. Because of the false reports, some of the cooperative nodes are rated with low reputations. Comparing Fig. 10(a) and (b), we see that as the number of the false-reporting nodes increases, the number of low reputations each node received increases.

In *Reputation*, a node exchanges its reputation observations with its neighbors and calculates the average as the node's global reputation. To make the global values of a given node in different nodes the same, we used broadcasting to ensure that each node receives others' local reputations. Fig. 11 shows the global reputation of each node in *Reputation* and ARM. *Reputation* exhibits a 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 nodes' reputation. Some reputations are not exactly 1 because some cooperative nodes may drop packages due to interference during transmissions.

Comparing Fig. 11(a) with (b), we find that having more false-reporting nodes in the system generates a greater variance in node reputations in *Reputation*. However, it does not have much influence on node reputations in ARM. More false reports incur higher inaccuracy of node reputations in the final global reputation calculation in *Reputation*. By taking advantage of the DHT, ARM efficiently gathers all local reputations of each node and filters out the false reports. It calculates node global reputation based on the justified reports.

E. Performance in Collusion Resilience

In this test, colluders drop received packets, misreport the reputation of their neighboring benign nodes, and collude with each other. We conducted experiments for both nongroup and group collusion. In the former, the colluders move individually and report high reputations for each other when meeting together. In the latter, all colluders in a group move together as a group and always rate each other highly. We consider collusion where colluders drop received packets with probability 0.3, and falsely report low reputations that are randomly chosen in [0.3, 0.4] for their neighboring cooperative nodes, and higher reputations are randomly chosen in [0.9, 1] for other colluders.

Fig. 12(a) and (b) shows node local reputations in a defenseless system with five and 10 colluders, respectively. It shows that a certain portion of nodes receives low R_l. These low R_l's 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_l's is proportional to the number of colluders in the system. This is because although colluders can increase their R_l's through collusion, they cannot always meet and collude with each other in the nongroup collusion. Therefore, more colluders lead to lower R_l's in the system.

Fig. 13 shows the global reputation of each node in *Reputation* and ARM in nongroup collusion. *Reputation* exhibits a larger variance than ARM in the reputation for cooperative nodes. This is because *Reputation* includes the false reports for

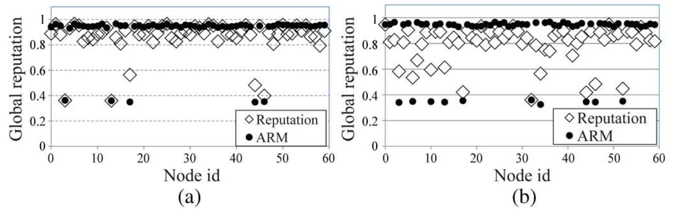


Fig. 13. Reputations in defenseless system with nongroup collusion. (a) Defensive system (five colluders). (b) Defensive system (10 colluders).

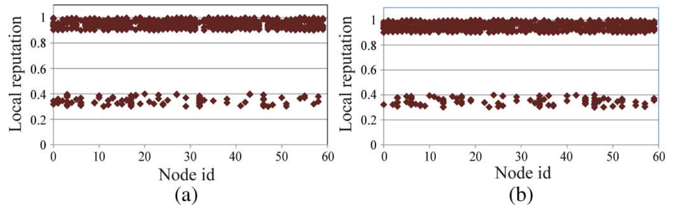


Fig. 14. Reputations in defenseless system with group collusion. (a) Defenseless system (five colluders). (b) Defenseless system (10 colluders).

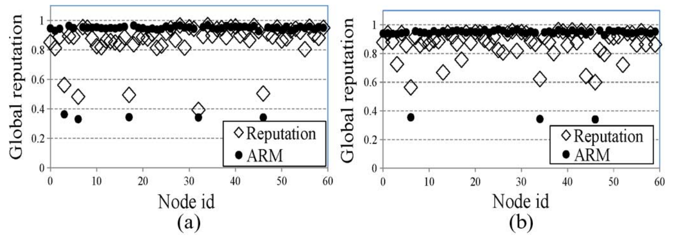


Fig. 15. Reputations in defensive system with group collusion. (a) Defensive system (five colluders). (b) Defensive system (10 colluders).

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 largely deviate from the 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_g of colluders since some colluders have high R_g's. This is because the colluders report high R_l's for each other when they meet. *Reputation* takes these false reports into account, while ARM filters them out when calculating R_g.

Fig. 14 shows the local reputations for group collusion in a defenseless system. Compared to Fig. 12, Fig. 14 has much fewer low node R_l's due to two reasons. First, in the group node collusion, the colluders can always report high reputations for each other to increase their own reputation. Second, more colluders generate lower R_l's for cooperative nodes. Fig. 15(a) and (b) shows the global reputation with group collusion in *Reputation* and ARM. When the number of colluders is five, even though 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_g's than *Reputation* for both cooperative nodes and colluders. 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 because the majority of the neighbors of a colluder are colluders and the false reports from the colluders overwhelm the reports from benign nodes.

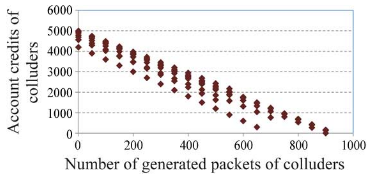


Fig. 16. Credits of colluders.

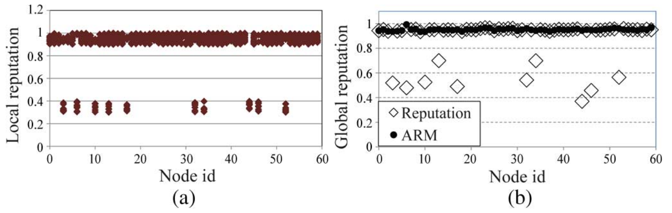


Fig. 17. Node reputation in adverse environment. (a) Defenseless system. (b) Defensive system.

The account management in ARM helps detect the colluders as shown in Fig. 16.

Fig. 16 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. In ARM, nodes earn credits by forwarding packets for others. Although the colluders can keep high R_g's by rating each other highly and receiving the fraudulent benefits of low a service price, they will ultimately use up their credits as they generate more packets, and finally are detected as uncooperative nodes by deficit accounts.

F. *Performance in Misreport Resilience*

We also tested whether ARM can accurately calculate node reputation with misreports due to an adverse environment with interfering background noise. We increased the background noise in 10 randomly chosen regions. All nodes in the system are cooperative. Fig. 17(a) shows node local reputations in the defenseless system. The low reputations are caused by misreports due to background noise. It is interesting to find that the nodes with low reputation are clustered. This is because, in the adverse environment, physically close nodes experience the interference noise at the same time. Fig. 17(b) shows that ARM can accurately reflect nodes' reputation in the adverse environment, while *Reputation* cannot accurately reflect some nodes' reputations. ARM collects all reports in the system using its DHT, identifies the cooperative nodes in the adverse environment by analyzing the clustering features of the nodes with low reputations, and then filters their reports. Although uncooperative nodes may cluster together to pretend that they are suffering noise interference, they will eventually be detected by account deficit. Since *Reputation* considers all reports for reputation calculation, nodes in the interfering area receive low reputations.

G. *Performance in Reputation Manager Auditing*

We run simulation for 20 times and used the average as the final experimental results. Fig. 18 shows malicious manager detection rate in a local reputation calculation with different number of malicious managers in the system. We see that the manager auditing algorithm can effectively detect the malicious

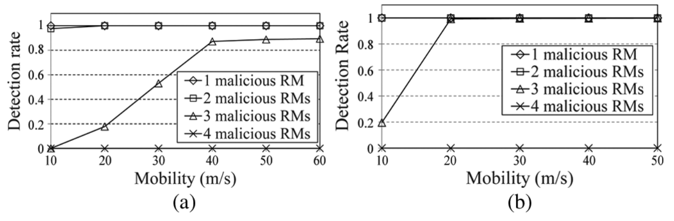


Fig. 18. Malicious reputation manager detection in local reputation collection. (a) Update period $= 10$ s. (b) Update period $= 20$ s.

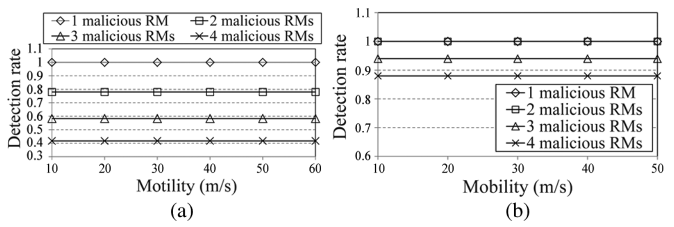


Fig. 19. Malicious reputation manager detection in global reputation collection. (a) Num. of RMs of a node $c = 2$. (b) Num. of RMs of a node $c = 4$.

managers when the number of malicious managers is small. However, as the number of malicious managers increases, the probability that a node meets a malicious manager increases. More malicious managers report false reputation of a node, leading to a decreased detection probability.

We also see that as the average mobility of the nodes increases, the detection probability increases. Higher mobility enables a node to meet more managers in a reputation update period, which helps detect malicious manager based on reputation reports from more managers. Comparing Fig. 18(a) and (b), we find that as the reputation update period increases from 10 to 20 s, the detection rate increases. A longer reputation update period enables a node to meet more managers during the period, which helps detect malicious manager based on reputation reports from more managers.

Fig. 19 shows the malicious manager detection rate in global reputation calculation versus node mobility with different number of malicious nodes in the system. We see that as the number of malicious managers increases, the malicious manager detection rate decreases. The increased number of the malicious managers increases the probability that a node has a malicious owner manager. We also see that node mobility does not affect the detection rate because a node finds malicious managers from the returned reputations from its owner managers, which is not affected by node mobility. Comparing Fig. 19(a) and (b), we find that as the number of managers of one node increases, the malicious manager detection rate increases. This is because given a constant number of malicious managers in the system, as more managers manage the reputation value of a node, the ratings from malicious managers can be more easily identified.

V. Conclusion

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

References

[1] "The state of the smartphone market," 2008 [Online]. Available: http://www.allaboutsymbian.com/

[2] Juniper Research, Basingstoke, U.K., "Next generation smartphones players, opportunities & forecasts 2008–2013," Tech. rep., 2009.

[3] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. MobiCom*, 2000, pp. 255–265.

[4] P. Michiardi and R. Molva, "Core: A collaborative reputation mechanism to enforce node cooperation in MANETs," in *Proc. CMS*, 2002, pp. 107–121.

[5] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol," in *Proc. MobiHoc*, 2003, pp. 226–236.

[6] Q. He, D. Wu, and P. khosla, "SORI: A secure and objective reputation-based incentive scheme for ad-hoc networks," in *Proc. IEEE WCNC*, 2004, pp. 825–830.

[7] T. Anantvalee and J. Wu, "Reputation-based system for encouraging the cooperation of nodes in mobile ad hoc networks," in *Proc. IEEE ICC*, 2007, pp. 3383–3388.

[8] B. Zong, F. Xu, J. Jiao, and J. Lv, "A broker-assisting trust and reputation system based on artificial neural network," in *Proc. SMC*, 2009, pp. 4710–4715.

[9] M. T. Refaei, L. A. DaSilva, M. Eltoweissy, and T. Nadeem, "Adaptation of reputation management systems to dynamic network conditions in ad hoc networks," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 707–719, May 2010.

[10] S. Buchegger and J. Y. LeBoudec, "A robust reputation system for mobile ad-hoc networks," in *Proc. P2PEcon*, 2004, pp. 1–6.

[11] J. Mundinger and J. Le Boudec, "Analysis of a reputation system for mobile ad-hoc networks with liars," *Perform. Eval.*, vol. 65, no. 3–4, pp. 212–226, 2008.

[12] J. Luo, X. Liu, and M. Fan, "A trust model based on fuzzy recommendation for mobile ad-hoc networks," *Comput. Netw.*, vol. 53, no. 14, pp. 2396–2407, 2009.

[13] R. Akbani, T. Korkmaz, and G. V. Raju, "Emltrust: An enhanced machine learning based reputation system for manets," *Ad Hoc Netw.*, vol. 10, no. 3, pp. 435–457, 2012.

[14] X. Wang, L. Liu, and J. Su, "RLM: A general model for trust representation and aggregation," *IEEE Trans. Services Comput.*, vol. 5, no. 1, pp. 131–143, Jan.–Mar. 2012.

[15] M. Jakobsson, J. Hubaux, and L. Buttyan, "A micropayment scheme encouraging collaboration in multi-hop cellular networks," in *Proc. Financial*, 2003, pp. 15–33.

[16] L. Buttyan and J. P. Hubaux, "Enforcing service availability in mobile ad-hoc WANs," in *Proc. MobiHoc*, 2000, pp. 87–96.

[17] L. Buttyan and J. P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc network," *Mobile Netw. Appl.*, vol. 8, no. 5, pp. 579–592, 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. IEEE INFOCOM*, 2003, pp. 1987–1997.

[19] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring, "Modelling incentives for collaboration in mobile ad hoc networks," in *Proc. WiOpt*, 2003, pp. 78–85.

[20] L. Anderegg and S. Eidenbenz, "Ad hoc-VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents," in *Proc. MobiCom*, 2003, pp. 245–259.

[21] H. Janzadeh, K. Fayazbakhsh, M. Dehghan, and M. S. Fallah, "A secure credit-based cooperation stimulating mechanism for MANETs using hash chains," *Future Generation Comput. Sys.*, vol. 25, no. 8, pp. 926–934, 2009.

[22] Z. Li and H. Shen, "Game-theoretic analysis of cooperation incentive strategies in mobile ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 8, pp. 1287–1303, Aug. 2013.

[23] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[24] K. Balakrishnan, Deng, and P. Varshney, "TWOACK: Preventing selfishness in mobile ad hoc networks," in *Proc. IEEE WCNC*, 2005, pp. 2137–2142.

[25] P. Dewan, P. Dasgupta, and A. Bhattacharys, "On using reputations in ad hoc networks to counter malicious nodes," in *Proc. ICPADS*, 2004, pp. 665–672.

[26] S. Bansal and M. Baker, "Observation-based cooperation enforcement in ad hoc networks," CS Dept., Stanford Univ., Stanford, CA, USA, Tech. Rep., 2003.

[27] K. Liu, J. Deng, P. K. Varshney, and K. Balakrishnan, "An acknowledgment-based approach for the detection of routing misbehavior in MANETs," *IEEE Trans. Mobile Comput.*, vol. 6, no. 5, pp. 536–550, May 2007.

[28] J. J. Jaramillo and R. Srikant, "DARWIN: Distributed and adaptive reputation mechanism for wireless networks," in *Proc. MobiCom*, 2007, pp. 87–98.

[29] Z. Li, H. Shen, and K. Sapra, "Leveraging social networks to combat collusion in reputation systems for peer-to-peer networks," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1745–1759, Sep. 2013.

[30] W. Wang, S. Eidenbenz, Y. Wang, and X. Y. Li, "OURS: Optimal unicast routing systems in non-cooperative wireless networks," in *Proc. MobiCom*, 2006, pp. 402–413.

[31] S. Zhong and F. Wu, "A collusion-resistant routing scheme for noncooperative wireless ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 582–595, Apr. 2010.

[32] J. Choi, K. Shim, S. K. Lee, and K. L. Wu, "Handling selfishness in replica allocation over a mobile ad hoc network," *IEEE Trans. Mobile Comput.*, vol. 11, no. 2, pp. 276–291, Feb. 2012.

[33] WiMAX Forum, "Requirements for WiMAX peer-to-peer (P2P) services," Tech. rep., 2012.

[34] C. Perkins, E. Belding-Royer, and S. Das, "RFC 3561: Ad hoc on demand distance vector (AODV) routing," 2003.

[35] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Comput.*, vol. 353, no. 5, pp. 153–181, 1996.

[36] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web," in *Proc. STOC*, 1997, pp. 654–663.

[37] G. A. Dirac, "Some theorems on abstract graphs," *Proc. London Math.*, vol. 3, no. 2, pp. 69–81, 1952.

[38] L. Ying, S. Yang, and R. Srikant, "Optimal delay–throughput tradeoffs in mobile ad hoc networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 4119–4143, Sep. 2008.

[39] "The network simulator NS-2," [Online]. Available: http://www.isi.edu/nsnam/ns/

[40] "Delay Tolerant Networking Research Group," [Online]. Available: http://www.dtnrg.org/wiki/Home

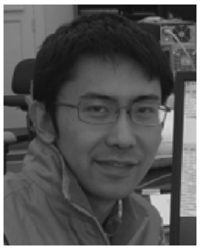
[41] V. Bernardo, M. Curado, T. Staub, and T. Braun, "Towards energy consumption measurement in a cloud computing wireless testbed," in *Proc. NCCA*, 2011, pp. 91–98.

[42] Z. Li and H. Shen, "A hierarchical account-aided reputation management system for large-scale MANETs," in *Proc. IEEE INFOCOM*, 2011, pp. 909–917.

Haiying Shen (M'07–SM'13) received the B.S. degree in computer science and engineering from Tongji University, Shanghai, China, in 2000, and the M.S. and Ph.D. degrees in computer engineering from Wayne State University, Detroit, MI, USA, in 2004 and 2006, respectively.

She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and cloud computing.

Dr. Shen is a Microsoft Faculty Fellow of 2010 and a member of the Association for Computing Machinery (ACM).



Ze Li received the B.S. degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2007, and the Ph.D. degree in computer engineering from Clemson University, Clemson, SC, USA, in 2012.

He is currently a Data Scientist with the MicroStrategy Incorporation, Tysons Corner, VA, USA. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks.