Cooperation in Mobile Ad Hoc Networks

Jiangyi Hu
Computer Science Department
Florida State University

January 11, 2005

Abstract

In this report we consider selfish node behavior in ad hoc networks and discuss trust and reputation mechanisms that will stimulate cooperation between nodes. We propose a Locally Aware Reputation system that addresses selfish behavior by using locally available information.

1 Introduction

Mobile ad hoc networks are paradigms for mobile communication in which mobile nodes are dynamically and arbitrarily located in such a manner that communication between nodes does not rely on any underlying static network infrastructure [1]. The communication medium is broadcast and the nodes in a mobile ad hoc network are usually portable mobile devices with constrained resources, such as power, computation ability and storage capacity. Since no fixed infrastructure or centralized administration is available, these networks are self-organized and end-to-end communication may require routing information via several intermediate nodes.

Due to the lack of infrastructure and the limited transmission range of a node in a mobile ad hoc network, a node has to rely on neighbor nodes to route a packet to the destination node. In particular, all network functions are based on the node cooperation. Currently, routing protocols for mobile ad hoc network, such as the Dynamic Source Routing (DSR) [33] and the Ad hoc On Demand Distance Vector Routing Protocol (AODV) [5], are based on the assumption that all nodes will cooperate. Without node cooperation, in a mobile ad hoc network, no route can be established, no packet can be forwarded, let alone any network applications. However, cooperative behavior, such as forwarding other node's messages, cannot be taken for granted.

We can identify two types of uncooperative nodes: *faulty or malicious* and *selfish*. *Faulty/malicious* behavior refers to the broad class of misbehavior in which nodes are either faulty and therefore cannot follow a protocol, or are intentionally malicious and try to attack the system. *Selfishness* refers to noncooperation in certain network operations. In mobile ad hoc networks, the main threat from selfish nodes is dropping of packets (blackhole), which may affect the performance of the network severely. Both *Faulty/malicious* nodes and *selfish* nodes are *misbehaved nodes*.

Due to the ad hoc nature of mobile ad hoc networks, enforcing cooperation in such networks is particularly challenging. The unique characteristics of mobile ad hoc networks raise certain requirements for the security mechanism.

1. Security mechanisms for enforce cooperation in mobile ad hoc networks should be distributed and self-organized. Security mechanisms involving any centralized service may no longer be viable because mobile ad hoc networks are self-organized and they cannot rely on any central authorities or external management.

2. Due to the constraints in bandwidth, computer power, and battery power in mobile devices, mechanisms should not cause undue resource consumption so as to degrade the performance of the network. Thus, there is an application-specific trade-off between security and functionality.

3. The dynamic topology of mobile ad hoc network requires that the security mechanisms be scalable and reliable.

The rest of the report is organized in the following way. Section 2 discusses recent related work. To address the problem of selfishness and to stimulate cooperation, virtual currency based schemes, such as Nuglets and Sprite. Reputation based schemes, such as CONFIDANT, CORE and OCEAN, are also introduced. Section 3 discusses some issues of the reputation methods in general, including the problem of calculating and updating reputation values, tracing faults, and reacting to uncooperative nodes. Section 4 presents the overview of our solution to improve the security of mobile ad hoc networks, followed by a discussion on implementations in Section 5. In Section 6, we conclude and discuss the future work.

2 Related Work

Schemes that stimulate cooperation and mitigate the detrimental effect of uncooperative nodes in mobile ad hoc network can be classified as 1) *virtual currency based schemes* and 2) *reputation based schemes*. Virtual currency schemes [7, 9, 10, 11, 31] use some form of incentive to enforce nodes' cooperation. Nodes get the incentives upon serving the network and use these to gain service from the network. If a node does not have any incentives, it will not get any service from the network. Reputation schemes [14, 17, 18] on the other hand uses the nodes' reputation to mitigate selfish behavior. Nodes maintain the reputation of other nodes based on direct observation or the exchange of reputation messages with other nodes. We will describe schemes under this classification below.

2.1 Virtual Currency Schemes

Since forwarding a message will incur a cost (of energy and other resources) to a node, an uncooperative node will need an incentive in order to forward messages of other nodes. Virtual currency systems [7, 9, 10, 11, 31] use credit or micro payments to compensate for the service of a node. A node receives a virtual payment for forwarding the message of another node, and this payment is deducted from the sender (or the destination node). Two example of such systems are: *Nuglets* [7, 9, 10, 11] and *Sprite* [31].

2.1.1 Nuglets

Buttyan and Hubaux introduced a virtual currency, called *nuglets*, and present a mechanism of charging/rewarding service usage/provision to stimulate cooperation in self-organized mobile ad hoc network [7, 9, 10, 11].

Two models were presented for using the nuglets: *packet purse model*, in which the source of the packet is charged and *packet trade mode*, in which the destination is charged.

In the *packet purse model*, when sending the packet, the source loads it with a number of nuglets sufficient to reach the destination. Each intermediate node takes some nuglets for the forwarding service.

In the *packet trade model*, packets are traded for nuglets by intermediate nodes. Each intermediary node "buys" the packet from the previous node for some nuglets and "sells" it to the next node for more nuglets. In this way, every intermediate node gains nuglets for forwarding and the total cost of forwarding the packet is paid by the destination node.

To implement either the *packet purse model* or the *packet trade model*, tamper-proof hardware is required at each node to prevent the node from illegitimately increasing its own nuglets and to ensure that the correct amount of nuglets is deducted or credited at each node. Mechanisms that use nuglets have some other problems:

2.1.2 Sprite

S. Zhong et al. proposed Sprite [31], a simple, cheat-proof, credit-based system for mobile ad hoc networks. Sprite uses credit to provide incentives for mobile nodes to cooperate and report actions honestly.

The basic idea of their scheme is as follows: a Credit Clearance Service (CCS) is introduced to determine the charge and credit to each node involved in the transmission of a message. When a node receives a message, the node keeps a receipt of the message and later reports it to the CCS when the node has a fast connection with the CCS. Payments and charges are determined from a game theory perspective.

In this scheme, the sender is charged, in order to prevent a denial-of-service attack to the destination by sending it a large amount of traffic. A node that has tried to forward a message is compensated, but the credit that a node receives depends on whether or not its forwarding action is successful. Forwarding is considered successful if and only if the next node on the path reports a valid receipt to the CCS.

Modelling the submissions of receipts regarding a given message as a one-round game, the authors proved the correctness of the receipt-submission system using game theory [31, 32].

2.1.3 Discussion on Virtual Currency Schemes

The basic problem with virtual currency schemes is they either depend on the use of tamper-proof hardware to monitor the increase or deduction of the virtual currency (as Nuglets does), or require a central server to determine the charge and credit to each node involved in the transmission of a message (as Sprite does). Both approaches may not be appropriate for truly mobile ad hoc network scenarios.

Also, they suffer from the location privilege problem [34]. Nodes in different locations of the network will have different chances for earn virtual currency, which may not be fair for all nodes. Usually, nodes at the periphery of the network will have less chance to be rewarded.

2.2 Reputation Based Schemes

Reputation systems are used in many area of electronic transactions, such as eBay and Amazon. Reputation mechanisms are applied to wireless mobile ad hoc network to address threats arising from uncooperative nodes. They rely on neighbor monitoring to dynamically assess the trustworthiness of neighbor nodes and excluding untrustworthy nodes.

Several reputation systems have been proposed to mitigate selfishness and stimulate cooperation in mobile ad hoc network, including:

- CONFIDANT [14]
- CORE [17]
- OCEAN [18]

2.2.1 CONFIDANT

Buchegger and Boudec present a reputation based protocol, called CONFIDANT, for making misbehavior unattractive [14, 15]. CONFIDANT stands for Cooperation Of Nodes: Fairness In Dynamic Ad-hoc Network, it works as an extension to on demand routing protocols.

CONFIDANT aims at detecting and isolating uncooperative nodes, thus making it unattractive to deny cooperation. Nodes rely on passive observation of all packets within a one-hop neighborhood. With CONFIDANT, each node has the following four components: a monitor, a trust manager, a reputation system and a path manager. These components interact with each other to provide and process protocol information.

- *The monitor* is the equivalent of a "neighbor watch", where nodes locally monitor deviating behavior. A node can detect deviation by its neighbor on the source route by listening to the transmission of its neighbor. The monitor reports any suspicious events and any incoming ALARM messages to the trust manager.
- *The trust manager* makes decisions about providing or accepting route information, accepting a node as part of a route, or taking part in a route originated by another node. It consists of the following components:
  - An alarm table containing information about received alarms.
  - A trust table managing trust levels for nodes to determine the trustworthiness of an alarm.
  - A *friends list* containing all the "friends" that the node may sends alarms to.

  ALARM messages contains the type and frequency of protocol violations, are sent by the trust manager of a node to warn others of malicious nodes. Outgoing ALARM messages are generated by the node itself after having experienced, observed, or received a report of malicious behavior. The recipients of these ALARM messages are so-called *friends*, which are administered in a *friends list*. Incoming ALARM messages originated from either outside friends or other nodes, so the source of an ALARM has to be checked for trustworthiness before triggering a reaction.
- *The reputation system* in this protocol manages a table consisting of entries for nodes and their rating. The rating is changed only when there is sufficient evidence of malicious behavior that is significant for a node and

that has occurred a number of times exceeding a threshold to rule out coincidences. To avoid a centralized rating, local rating lists and/or black lists are maintained at each node and potentially exchanged with friends.

- *The path manager* performs the following functions: path re-ranking according to reputation of the nodes in the path; deletion of paths containing malicious nodes, action on receiving a request for a route from a malicious node (e.g. ignore, do not send any reply) and action on receiving request for a route containing a malicious node in the source route (e.g. ignore, alter the source).

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

Buchegger et al. improved the CONFIDANT protocol in [16] to cope with false disseminated reputation information. A trust rating is introduced to represent the trustworthiness of a node. In addition to reputation rating, each node also maintains a trust rating for every other node and first hand information about its neighbors. The first hand information is disseminated, but the reputation rating and trust rating are never published, they are updated accordingly.

Only second hand reputation information that is compatible with the current reputation rating will be accepted. It works as follows. First, whenever a node i makes first hand observation of node j's behavior, it updates its first hand information $F_{i,j}$ and the reputation rating $R_{i,j}$. Second, nodes broadcast their first hand information to their neighbors. For example, node i receives from node k the first hand information $F_{k,j}$ about j. If according to the trust rating of node k, $T_{i,k}$, node k is trustworthy, then $F_{k,j}$ is accepted and used to update $R_{i,j}$ and $T_{i,k}$ is also slightly improved. Otherwise, $R_{i,j}$ is not updated, and $T_{i,k}$ is slightly worsened. A Bayesian approach is used to evaluate both reputation rating and trust rating.

2.2.2 CORE

P. Michiardi et al. proposed a mechanism called CORE (COllaborative REputation mechanism), to enforce node cooperation in mobile ad hoc network [17]. It is a generic mechanism that can be integrated with any network function like packet forwarding, route discovery, network management and location management.

CORE stimulates node cooperation by using a collaborative monitoring technique and a reputation mechanism. In this mechanism, reputation is a measure of someone's contribution to network operations. Members that have a good reputation can use the resources while members with a bad reputation, because they refused to cooperate, are gradually excluded from the community.

CORE defines three types of reputation [17,

28]:

1. *Subjective reputation* is a reputation value which is locally calculated based on direct observation. For example, node A calculates the reputation of a neighbor node B at a given time for a particular function.

2. *Indirect reputation* is second hand reputation information which is established by other nodes. For example, in CORE, node A will accept the indirect reputation of node B from node C. To eliminate an attack where a malicious node disseminates false negative reputation information, only positive reputation information is distributed in CORE.

3. *Functional reputation* is related to a certain function, where each function is given a weight as to its importance. For example, data packet forwarding may be deemed to be more important than forwarding packets with route information, so data packet forwarding will be given greater weight in the reputation calculations.

Each node computes a reputation value for every neighbor using a sophisticated reputation mechanism that differentiates between subjective reputation, indirect reputation and functional reputation.

CORE consists of two basic components: a watchdog mechanism and a reputation table. The watchdog mechanism [24, 17] is used to detect misbehavior nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. The watchdog does this by listening promiscuously to the next node's transmissions. If the next node does not forward the packet, then it is considered as misbehaving.

The reputation table is a data structure stored in each node. Each row of the table consists of four entries: the unique identifier of the entity, a collection of recent subjective observations made on that entity's behavior, a list of the recent indirect reputation values provided by other entities and the value of the reputation evaluated for a predefined function.

2.2.3 OCEAN

S. Bansal et al. proposed an Observation-based Cooperation Enforcement in Ad hoc Networks (OCEAN) [18]. In contrast to CONFIDANT and CORE, OCEAN avoids indirect (second hand) reputation information and uses only direct first-hand observations of other nodes behavior. A node makes routing decisions based solely on direct observations of its neighboring nodes interaction.

In OCEAN, the rating of each node is initialized to Neutral(0), with every positive action resulting in an increment (+1) of the rating, and every negative action resulting in a decrement (-2) of the rating. Once the rating of a node falls below a certain faulty threshold (-40), the node is added to a faulty list. The faulty list represents a list of misbehaving nodes.

OCEAN has five components reside in each node to detect and mitigate misbehavior.

- *NeighborWatch* observes the behavior of the neighbors of a node. It works the same way as watchdog [24]. Whenever misbehavior is detected, NeighborWatch reports to the *RouteRanker*, which maintains ratings of the neighbor nodes.

- *RouteRanker* maintains a rating for each of its neighboring nodes. The rating is initialized to Neutral and is incremented and

decremented based on observed events from the *NeighborWatch* component.

- *Rank-Based Routing* uses the information from *NeighborWatch* to make the decision of selection of routes. An additional field, called the avoid-list, is added to the DSR Route-Request Packet (RREQ) to avoid routes containing nodes in the faulty list.

- *Malicious Traffic Rejection* rejects traffic from nodes which is considered misbehaving. All traffic from a misbehaving node are rejected so that a node is not able to relay its own traffic under the guise of forwarding it on.

- *Second Chance Mechanism* allows nodes previously considered misbehaving to become useful again. A timeout approach is used where a misbehaving node is removed from the faulty list after a fixed period of inactivity. Even though the node is removed from the faulty list, its rating is not increased, so that it can quickly be added back to the faulty list if it continues the misbehavior.

OCEAN focuses on the robustness of packet forwarding: maintaining the overall packet throughput of mobile an ad hoc network with the existence of misbehaving nodes at the routing layer. OCEAN's approach is to disallow any second-hand reputation exchanges. Routing decisions are made based solely on direct observations of neighboring nodes behavior. This eliminates most trust management complexity.

3 Issues with Reputation Methods

As see from section 2.2, although the reputation based schemes applied to mobile ad hoc networks may be different in implementation, they are all composed of essentially three different parts:

1. The calculation and update of reputation values
2. The detection of misbehavior
3. The reaction to uncooperative behavior

3.1 Calculation and Update of Reputation Values

Applied to mobile ad hoc networks, reputation can be defined as one node's perception of another node's performance of some network operation [28]. It is used as a prediction of future quality of service. However, since reputation is not a tangible property, the reputation value has to be explicitly defined. There are some issues that should be considered during the calculation and update of the reputation value.

3.1.1 Trust vs. Reputation

An important concept in network security is trust, interpreted as a relation among entities that participate in various protocols. Trust relations are based on evidence related to the previous interactions of entities within a protocol [19]. A lot of research has been done to evaluate and manage trust in mobile ad hoc networks, such as [19, 20, 21, 22, 23]. Most of the research focuses on establishing an indirect trust relation between two nodes (to exchange public keys or certificates) without previous direct interaction.

We will not discuss this problem in detail here, instead, we will focus on the relationship of trust and reputation in reputation based systems for mobile ad hoc networks.

In most reputation systems [15, 30, 17, 18], reputation value is a metric for trust. A node with a good reputation means it behaves very well and thus is trustworthy, while nodes with bad reputation are uncooperative and not trustworthy.

Buchegger et al. distinguish trust from reputation in [16]. For each node, reputation rating represents how well a node behaves and trust rating represents how honest a node is. Reputation value is used to decide whether the node is regular or misbehaved, while trust rating is used to decide whether the node is trustworthy or not, thus the indirect reputation message from the node is accepted or not.

3.1.2 Direct vs. Indirect Trust (Reputation)

Direct reputation is derived from first hand experience. A node gets such information about another node, usually its one-hop neighbor, by direct observation. For example, node M forwards a message (either a routing message or a data packet) to its next hop neighbor, N, and expects N to further forward the message. M can get first hand information by monitoring whether N correctly participates in the protocol.

Indirect reputation information (also refers to second hand reputation information) is reputation information about a node from other nodes. Such reputation information can be in the form of a blacklist, friends list or a reputation table. It may be first hand information of the sender or maybe transmitted hop-by-hop from the originator.

We can model trust and reputation as follows:

- Direct trust and reputation are based on direct knowledge or observation.
- Trust and reputation may not be symmetric.

  For example, if node A knows that node B to be trustworthy, this does not imply that B knows that A is trustworthy.
- Trust and reputation are usually assumed to be transitive .

  For example, if node A knows that node B is trustworthy and node B knows that node C is trustworthy, then node A can trust C.
- Indirect trust and indirect reputation are based on trust and reputation that link nodes. For example, if node A trusts node B is trustworthy and node B trusts node C, then A trusts C. On the other hand, if node A does not trust node B, then A will not trust C even if B trusts C.
- A node A can know something about another node C from the indirect reputation message if and only if A knows the indirect reputation information is from a trustworthy node B.

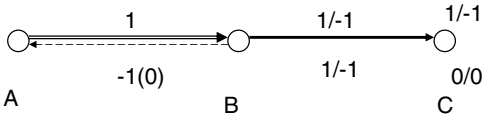

Figure 1: An example of trust and indirect reputation information

In Figure 1, we use 1 to represent trustworthy, -1 for untrustworthy and 0 for unsure of the trustworthiness. The arrowed

double line indicates the direction of trust. The dashed line indicates the transmission of a indirect reputation message and the arrowed line indicates the indirect reputation message.

As show in Figure 1, if A knows B is trustworthy, then A can trust what B said and decide either trust C or not.

If A knows B is not trustworthy or unsure of B's trustworthiness, then A will not trust what B said because B may tell the truth or lie. So A can't decide whether to trust C or not.

- A node A can know something about the indirect reputation message provider, B, if and only if what B says about node C is contrast from what A knows C is.

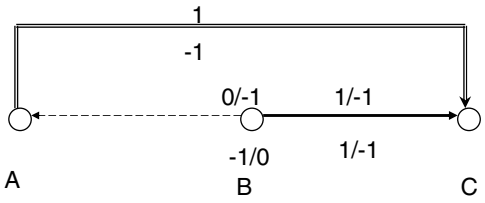

Figure 2: An example of trust and indirect reputation information

As show in Figure 2, if A knows C is trustworthy / not trustworthy, but what B said is contrast from this knowledge, then A knows B must be false accuse / praise C and A will consider B not trustworthy.

If A knows C is trustworthy / not trustworthy and what B said is the same with this knowledge, then B may be trustworthy and telling the truth (as always) or it maybe a liar but is honest this time. A can't decide if B is trustworthy or not.

If A is not sure of C's trustworthiness, then A knows nothing about B or C from the indirect reputation message.

3.1.3 Global vs. Local reputation

Most reputation systems [30, 15, 16, 17] for mobile ad hoc network uses global reputation, in which every node knows reputation of every other node in the network. This is achieved by exchange indirect reputation messages among the network. Since indirect reputation information may be from an untrustworthy node, reputation systems using global reputation information suffer from false rating, either false accusation or false praise [28]. Other issues with global reputation mechanism include:

1. Since each node maintains reputation values of every other node, storing such information requires more storage at each node. Take CONFIDANT and CORE for example, every node has to maintain $O(N)$ reputation information, where N is the number of nodes in the network.

2. Disseminate reputation information greatly increases the volume of network traffic. As for CONFIDANT and CORE, the reputation message distributed during each reputation disseminate period is $O(N^2)$, where N is the number of nodes in the network. Consider mobile ad hoc network, where bandwidth is very limited, this is an important issue.

3. Every time a node receives indirect reputation information, it has to decide whether to accept or not. If the information is accepted, then it is incorporated and the reputation table is updated. This causes additional computation at each node.

4. Reputation information, as data packet, could be modified, replayed or accidentally lost during transmission.

As discussed, global reputation methods are unreliable and complex, distributing reputation information cause additional expense for both the node and the overall network. Is global reputation really necessary? The answer is no. Some may argue that global reputation is helpful if a node moves in the network. But a well-behaved node here does not mean it will behave as well when it moves to another place. For example, a malicious node may build its reputation first and then moves to a certain location to behave maliciously. Furthermore, nodes usually does not care about reputation of distant nodes. They are more concerned about reputation of nodes in their neighborhood, that is, they care more about local reputation rather than global reputation.

OCEAN uses only local reputation, which is based on direct first-hand observations of one-hop neighbors. Any second-hand reputation exchanges are disallowed. According to their simulation, OCEAN achieves a reasonable performance, in terms of network throughput, while being less complex and less vulnerable to false accusations [18]. Compare with global reputation, local reputation mechanism has low cost, is more reliable and more efficient.

3.1.4 Initiate Reputation Value

When a new node enters the network, or a node moves to a new location, where nobody knows about its reputation, an initial reputation value should be given. Each reputation system has a learning period, as the network will not know how a new node will behave.

Assign the lowest possible reputation value to a new node will force it to perform positive work to gain a good reputation, and thus discourage new participants from malicious behavior. But this mechanism may not be feasible in an ad hoc network, where instantaneous connection is required and nodes are more mobile. It may take too much time for a new node to establish its reputation.

Assigning a null value is a reasonable approach and CONFIDANT, CORE and OCEAN all allocate neutral reputations to new nodes [14, 17, 18, 28].

3.1.5 Inconsistent Reputation Value

In reputation systems, different nodes may have different reputation values for the same node. This is called the inconsistent reputation problem. It may be caused by many reasons.

- Nodes may calculate reputation values differently. For example, in both CORE and CONFIDANT, the overall reputation value of a node is a combination of several ratings of its network functions, with weighings applied to these functional reputation values. The reputation mechanisms usually assume that every node will assign the same weights to the functions. This is a potentially inappropriate assumption in a mobile ad hoc network, where nodes with different capabilities and roles are likely to place different levels of importance on different functions [28].

- For different nodes, first hand reputation values of a same node may vary. Every node gets first hand reputation information about nodes that it interact with based on

its own experience. A node may behave differently when interact with different nodes, thus the reputation value for a same node may vary. For example, two nodes, A and B both interact with node C, but A and B may have different reputation values for C. This is possible if C react differently to request from A and B; or A or B may interact with C often than the other.

- Every node deals with the received indirect reputation information based on its own judgement. This also results in the difference. Reputation information accepted by node A may not be accepted by node B because it is either incompatible with B's experience or B does not trust the sender.

None of the reputation methods discussed above require nodes in a mobile ad hoc network to reach a consensus on which nodes misbehave. The problem caused by inconsistent reputation value is that nodes at the network may have different ratings about others. A node may be considered regular by some nodes, while considered misbehaved by others. This makes it hard to distinguish correct reputation ratings from false reputation message, the problem is especially important when indirect reputation message is distributed and global reputation value is calculated.

3.2 Detection of Misbehavior and Tracing Fault

In order for reputation values to be valid, nodes will need a reliable way of detecting good or bad behavior. CONFIDANT, CORE and OCEAN all rely on promiscuous observation for monitoring function operations. However, passive observation presents several weaknesses used within mobile ad hoc network, it might not detect a misbehaving node in the presence of [24]:

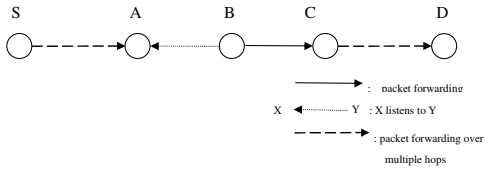

Figure 3: The watchdog mechanism.

1. Ambiguous collision. As shown in Figure 3, an ambiguous collusion is the scenario that packet collision occurs at A while it is listening for B to forward on a packet.

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

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

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

5. Partial dropping, in which a node dropping packets at a lower rate than the configured minimum misbehavior threshold.

3.3 Reaction to Uncooperative Behavior

Once an uncooperative node has been identified, it is isolated and exclude from the network. Usually, neighbors of the uncooperative node refuse

to forward any packets originated from the convicted node, depriving the network services.

However, since the function of a mobile ad hoc network depends on all the participate nodes. The objective is to force the nodes to cooperate and benefit each other [34]. Thus, an uncooperative node should be punished temporally and be given chance to behave normal again. OCEAN uses the "Second Chance Mechanism" to allow nodes previously considered misleading to become useful again [18]. It uses a timeout-based approach where an uncooperative node is accepted by the network after a fixed period of observed inactivity. The rating of the node is not changed, so that it can quickly be detected if the misbehavior continues.

4 Proposed Solution: Locally Aware Reputation System

We propose a simple reputation based scheme, called LARS (Locally Aware Reputation System), to mitigate misbehavior and enforce cooperation. Different from global reputation based schemes, such as CONFIDANT and CORE, our solution uses local reputation only. Each node only keeps the reputation values of all its one-hop neighbors.

To mitigate the detrimental effect of selfish and malicious node, when an uncooperative node is identified, its k-hop neighbors become aware of the misbehavior, where k is a parameter which is adaptive to the security requirement of the network. To avoid false accusation, conviction of the uncooperative node is co-signed by m different (one-hop neighbor) nodes, where $m-1$ is an upper bound on the number of malicious nodes in the one-hop neighborhood. We will discuss the protocol in detail below.

4.1 Model and Assumptions

LARS addresses the problem of node cooperation in self-organized mobile ad hoc networks. In these networks, nodes may not belong to a single authority and do not have common goals. Some nodes may be disruptive and others may attempt to save resources through selfish behavior. In addition, these networks could be self-organizing, meaning that the regular function of networks solely depends on the operation of the end-users.

In our scheme, trust in a node is associated with its reputation value. There are three trust levels and we use a trust value, T, to represent the trustworthiness of a node. A node A considers another node B either

- trustworthy, with $T = 1$,
- untrustworthy, with $T = -1$, or
- trustworthy undecided, with $T = 0$

A trustworthy node is a regular (well-behaved) node that can be trusted. An untrustworthy node is a misbehaved node and should be avoid and deprive of services. A node with undecided trustworthiness is usually a new node in the neighborhood. It may be a regular or a misbehaved node, depending on its future performance.

Every node keeps a reputation table, which associates a reputation value with each of its neighbors. It updates the reputation table based on direct observation only. No global reputation value is calculated, and no indirect reputation message is distributed.

Reputation values R are between a range $R_{min} < R < R_{max}$, and there are two threshold, $R_u > R_{min}$ for untrustworthy and $R_t < R_{max}$

for trustworthy. For a node N with reputation value R and trust value T,

- $T = 1$ (N is trustworthy), if $R_t < R < R_{max}$,
- $T = -1$ (N is not trustworthy), if $R_{min} < R < R_u$,
- $T = 0$ (N is trustworthy undecided), if $R_u < R < R_t$.

A new node, either a node that just entered the network or a node that has moved to a new neighborhood, will be assigned a reputation value between R_u and R_t because its trustworthiness is unknown. A fade factor w is introduced to give less weight to evidence received in the past to allow for reputation fading.

We based our discussion on the following assumptions:

- Each node has a unique, persistent and distinct identity.
- Each node knows its one-hop neighbors.
- Transmission distance of each node is the same.
- Links between nodes are bidirectional.
- Nodes do not have a priori "trust" relationship. Initially, the reputation value of every node is set to a value between R_u and R_t.
- On-demand routing protocols, such as Dynamic Source Routing (DSR) [33] or Ad hoc On-demand Distance Vector Routing(AODV) [5] are used to establish route.
- The number of malicious nodes in a one-hop neighborhood is less than m. That is, for each node, among all the one-hop neighbor nodes, there are at most $m - 1$ malicious nodes.

We use the following notations through the rest of our discussion. Capitalized letters are used to represent nodes, specifically, S stands for the source node, D for the destination node, M is usually a misbehaved node. Let ID_X be the identity of node X, $N(X)$ be the set of one-hop neighbor nodes of node X, and $R_Y(X)$ be the reputation value of node X, in node Y's reputation table.

4.2 Overview

Each node X maintains a reputation value for each of its neighbors in $N(X)$. Based on the direct observation of the neighbors, the reputation values are updated. If the reputation value of a neighbor node, for example M, drops below the untrustworthy threshold R_u, then M is considered misbehaved by X. X will notify its neighbors about M's misbehavior by initiating a WARNING message.

To prevent false accusations and problems caused by inconsistent reputation values, the WARNING message should be signed by m nodes before it can be broadcasted to the k-hop neighborhood. Any nodes within one-hop distance of the misbehaved node, M, can sign the WARNING message if, in its reputation table, M's reputation value has also dropped below the untrustworthy threshold R_u. We will discuss the problem of signing a WARNING message in 5.3.

We assume that a routing path is established using on-demand routing protocol, such as DSR [33] or AODV [5]. Misbehavior is possible during the process of routing. To deal with this, routing requests can be flooded to ensure the establishment of a route from the source node S to

the destination node D if there exists one. We assume that the source node and the destination node are not malicious and focus our discussion on the misbehavior (of intermediate nodes) during communication between two trusted nodes.

Consider the scenario in which S sends a message to D using a multi-hop path. Every time an intermediate node, I, forwards the message, its neighbor nodes, $N(I)$, can overhear the forwarding and keep a record of the message sent and set a timer. Due to collisions and other coincidence, not all nodes in $N(I)$ hear the forwarding.

If the message reaches D, then D returns an acknowledgement, *ack*, back to S along the reverse of the same path. If S gets the *ack* from D within a certain time period, it is confirmed that the communication is successful, otherwise, it initiates a trace process to identify a misbehaved node. A special *trace* packet is initiated by S and sent along the same path. Neighbors of nodes along the path which already have a record of the message will participate in the trace process. For every neighbor node, if a *trace* is received before the timer has expired, then it broadcasts the record and helps to find the misbehaved node, otherwise, the record is discarded. We will discuss the trace algorithm in 4.4.

4.3 Calculation and Update of Reputation Value

LARS uses local reputation value, where each node maintains only reputation values of its one-hop neighbors. The reputation value is updated based only on its direct observation of the neighbors, no second hand reputation information is exchanged and integrated. Such scheme has certain advantages:

- Since the neighbors are all within direct communication distance, the reputation information can be derived directly from first hand observation. No indirect reputation information is necessary in the calculation and update of the reputation value, thus avoiding the complexion discussed in 3.1.2 and 3.1.3. The reputation value is accurate and the calculation is simple, easy and fast.

- No second hand reputation information is distributed, thus eliminating the additional network traffic caused by the distributed reputation systems.

- Each node only keeps the reputation values of its neighbor nodes. Compared to global reputation systems, in which each node has to keep reputation value of all the nodes, our scheme saves a lot of storage at each node.

Every time a node participates in the network protocol, its one-hop neighbors update its reputation value accordingly. If the participation is positive, then the reputation value is increased, otherwise, the reputation value is decreased.

Suppose I behaves regularly. Then every time it forwards a message, its one-hop neighbors observe the normal behavior and increase its reputation value by μ. That is,

$$R_X(I) = R_X(I) + \mu, X \in N(I)$$

Suppose M is a malicious node and drops the message from its previous node, N. There are several different cases. We use the figures below to illustrate each of them. In these figures, a circle describes the one-hop neighborhood of a node, a solid vector describes the forwarding of a message, a dotted vector describes the forwarding of a *trace*, a dashed vector describes non forwarding (either a message or a *trace*). We do not

show the forwarding of *ack* in the figures. M's neighbors will update M's reputation as follows:

- Case 1: N has sent a message to M, but M has failed to forward it.

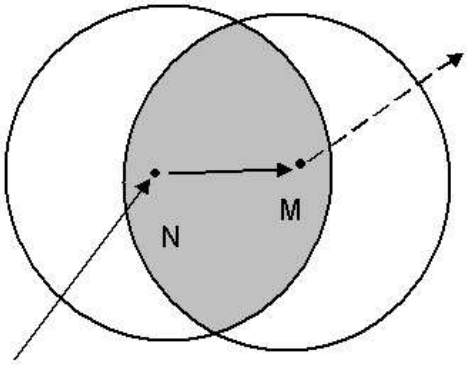

Figure 4: N has sent a message to M, but M has failed to forward it

As shown in Figure 4, for every node X in the shaded area, which is a one-hop neighbor of both N and M, X detects the misbehavior and reduces the reputation value of M by α, where $\alpha > \mu$. That is,

$$R_X(M) = R_X(M) - \alpha, X \in N(M) \cap N(N)$$

- Case 2: M has not forwarded the message, but forwarded the *trace*.

  As shown in Figure 5, for every node X, which is a one-hop neighbor of both N and M, X reduces the reputation value of M by α.

  For every node X, which is a one-hop neighbor of M, but not a one-hop neighbor of N, X realizes that M has not forwarded the message when it gets the *trace*, then X will

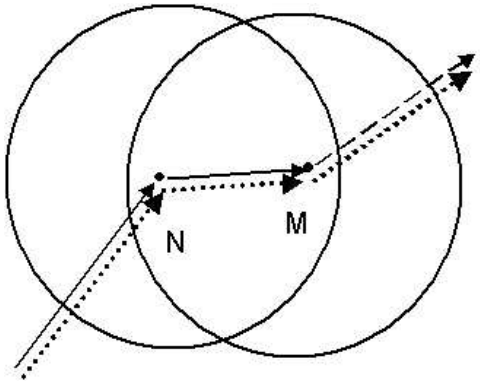

Figure 5: M has not forwarded the message, but forwarded the *trace*

reduce M's reputation value by α. Thus, all one-hop neighbors of M reduce its reputation by α. That is,

$$R_X(M) = R_X(M) - \alpha, X \in N(M)$$

- Case 3: M has not forwarded the message, and has also dropped the *trace*.

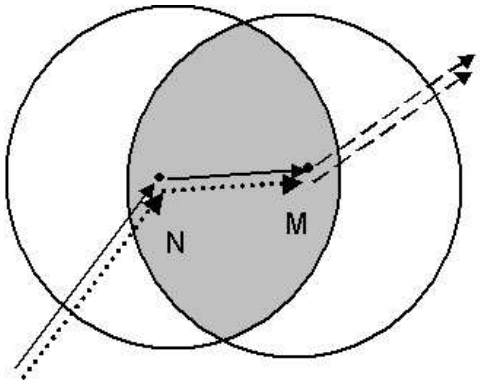

Figure 6: M has not forwarded the message, and has also dropped the *trace*.

As shown in Figure 6, for every node X,

which is a one-hop neighbor of M, but not a one-hop neighbor of N, X will not observe M's misbehavior and will not update M's reputation value.

For every node X in the shaded area, which is a one-hop neighbor of both N and M, X detects that both the message and the *trace* were dropped, and reduces M's reputation value by β, where $\beta > \alpha$. That is,

$$R_X(M) = R_X(M) - \beta, X \in N(M) \cap N(N)$$

- Case 4: M has not forwarded the message, and a neighbor, M', colluded with M and sent a forged record.

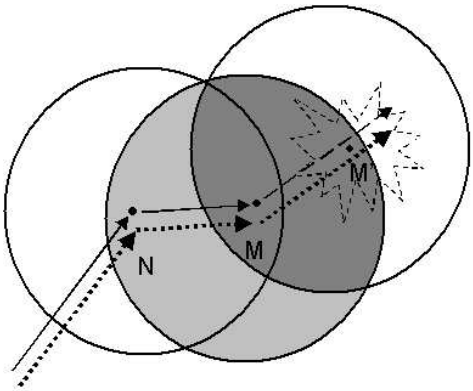

Figure 7: M has not forwarded the message, and a neighbor, M', colluded with M and sent a forged record.

As shown in Figure 7, for every node X, which is a one-hop neighbor of both M and M', X detects the cheating and reduces both M and M''s reputation by γ, where $\gamma > \beta > \alpha$.

For all other one-hop neighbors of M, we get the same as case 2 and the reputation value of M is reduced by α. That is,

$$R_X(M) = R_X(M) - \gamma, X \in N(M) \cap N(M')$$

$$R_X(M') = R_X(M') - \gamma, X \in N(M) \cap N(M')$$

$$R_X(M) = R_X(M) - \alpha, X \in N(M) - N(M')$$

We have $\alpha > \mu$, so that the punishment is greater than the reward, to encourage cooperation. This makes it hard for a node to built its reputation and then misbehave. Even if a node has a good reputation, when it misbehaves, its reputation value will drop quickly (faster than can built it).

Also, we treat different misbehavior types differently. If only the message is dropped, as discussed in case 1 and case 2, it might be an accidental misbehavior. But if both the message and the *trace* are dropped by a node, as in case 3, then it is possibly an intentional misbehavior and it is punished more severely. If collusion and cheating occur, as in case 4, then the greatest punishment is applied.

In all cases, $R_X(M)$ is updated by w, the fading factor, after a certain time interval (the fading timer), to give less weight on past experience. That is,

$$R_X^{t1+\Delta}(M) = wR_X^{t1}(M)$$

where Δ is the fading timer and $R_X^{t1}(M)$ is the reputation value of M at time $t1$ at node X's reputation table.

4.4 The Trace Algorithm

As discussed in 4.2, if the source node, S, does not get the acknowledgement from the destination node D within a certain time period, then a *trace* is sent along the path and the fault tracing process starts. The tracing algorithm traces the fault to a misbehaved node which caused the

disruption of communication. We now describe our tracing algorithm in detail.

When an intermediate node, I, gets the *trace*, it forwards the *trace*. The neighbors of node I respond with the kept record if I had forwarded the message. The response is broadcasted so any nodes that have received the *trace* will get it and can further broadcast it upstream. Any nodes that didn't get a *trace* ignore the response. Broadcasting is used in forwarding the response to make sure the source will get the response and identify the misbehaved node. Finally, the response is broadcasted to the source node. From the responses, the source node can identify the misbehaved node and will try to use alternative path to avoid the misbehaved node. We will discuss how the trace algorithm works using an example below.

When there is no misbehavior, a message is forwarded hop by hop from S to the destination node D. Neighbor nodes of a node along the route keep records of the forwarded message and wait for time $T = 2n\tau$, where n is the number of hops from S to D, τ is the upper bound of the one-hop round trip. If no *trace* message is received before the timer expire, then the nodes discard the record. If the destination node, D, receives the message and verifies it is valid, then an acknowledgement is sent back S.

If D does not get a message, then S gets no acknowledgement within time $T = 2n\tau$ and it starts the trace process by sending a trace message along the same path. Every intermediate node on the route forwards the trace when it receives the trace. A node will not drop a trace message because if it does so, then its neighbors will not get the trace message and will not reply with a response. As show in Figure 5, assume node M is either a malicious or a selfish node that has dropped the message. Node N forwards the *trace* to node M. The neighbors of N, overhear the *trace* message will broadcast the record verifying that N had forwarded the message. Any upstream node that gets the record will broadcast it upstream. Since M had not forwarded the message, then no response will be generated from neighbors in vicinity of M. Finally, S finds out that N is the last node that has forwarded the message and the next node of N, node M, is the misbehaved one.

4.5 Reaction to Uncooperative Behavior

If a node's reputation value drops below the threshold, R_u, then it is considered misbehaved and a WARNING message about the node is generated.

Before the WARNING message is broadcasted to the neighborhood, it should be signed by m nodes, where $m - 1$ is the upper bound of malicious nodes in a one-hop neighborhood. This ensures the trustworthiness of the WARNING message and is robust against false accusation. There are two reasons for this requirement:

- First, due to the problem of inconsistent reputation value as discussed in 3.1.5, different nodes may have different reputation values for a same node, thus a misbehaved node considered by node A may be considered regular by node B. But if m nodes in the one-hop neighborhood of a node agree it is misbehaved, then it is convicted.

- Second, requiring m nodes to sign a WARNING message also prevents nodes from false accusation. Since the number of malicious nodes is bounded by m, malicious nodes cannot collude to forge a valid WARNING message to frame a regular node.

After a WARNING message is verified, it is broadcasted to the k-hop neighborhood of the misbehaved node, so that all the k-hop neighbor nodes become aware of the misbehavior and deny service for the misbehaved node.

But the misbehaved node is not excluded from the network forever. The objective is to let the misbehaved node learn by punishment and behave well in the future. After a time-out period, it is accepted but with the reputation value unchanged so that it has to rebuild its reputation again by good cooperation.

4.6 Optimization

We optimize the scheme as follows:

- To save space and energy consumption in the neighboring nodes, in a dense network, only k out of n neighbors keep a record of the forwarded message, where n is the number of neighbors of the node and k is a random number between 1 to n. Instead of involving every neighbor of a node in the monitoring, this scheme randomly chooses some neighbors, called *probes*, to monitor the message forwarding of a node. Every time a node forward a message, k out of n neighbor nodes will be randomly selected as *probe*s to monitor the forwarding.

  The randomization has two advantages. First, being a *probe* means providing services for the network society, thus more space, time and energy consumption. Choosing random *probe* every time gets every node involved and it is fair. Second, the randomization will select different *probe* every time, thus eliminates the cheating and colluding problem which will caused by fixed *probe*s.

  Every *probe* keeps a record of the message sent and set a timer. If before the timer expires, a *trace* is received, then the *probe* broadcasts the record and helps to find the misbehaved node; otherwise, the record is discarded.

- We can further eliminate bandwidth usage as follows. An upstream node does not forward the responses from the neighbor *probes* immediately; instead, it keeps the response for a certain time period $T = m\tau$, where m is the number of hops from the node to the destination node, τ is the upper bound of the one-hop round trip. When the timer expires, the node then broadcasts the latest response, which may from the *probe* of the next node or from another downstream *probe*.

5 Implementation

We now describe the protocol in detail. Let ID_X be the identity of node X, *seq* the sequence number of the message and is incremented every time a new message is sent, t the timestamp of the message, m the message, and $MAC(m)$ the keyed MAC (Message Authentication Code) with key K_{SD}, which is shared by S and D. To simplify the discussion, we assume privacy is not required and no need to encrypt the packet.

Let $Route(S, D) = (S = X_0, X_1, X_2, \cdots, X_i, \cdots, X_n = D)$ be the route used. *Probes* around node X_i are denoted by $P(X_i) = \{P_{i1}, P_{i2}, \cdots, P_{ik}\}$. Message sent from S is $pkg_s = [ID_S, ID_D, seq, t, m, MAC(m)]$, trace message is $trace = [ID_S, ID_D, seq]$, a *probe* around node X_i keeps a record $R_i = [ID_S, ID_D, seq, MAC(m), X_i, X_{i+1}]$.

5.1 Probe Selection Algorithm

As stated in 4.6, every time a message is forwarded by a node, random neighbor nodes will be served as *probes* to monitor the forwarding. The *probes* are selected as follows. Every neighbor node overhearing the forwarding finds out if it is selected as *probe* by using a predefined hash function [36] h over the sequence number seq, the timestamp of the message, t and its own identity. The result of the hash function is either a 0 or a 1. If the node gets 1, then it will server as a *probe*, otherwise, it ignores the message. That is, for every neighbor node X,

- selected as *probe*, if $h(seq, t, ID_X) = 1$;
- not selected as *probe*, if $h(seq, t, ID_X) = 0$.

5.2 The Trace Algorithm

The following pseudo code show how the trace algorithm works.

- Source node S:
  1. S send to X_1: pkg_s
  2. set timer $T = n\tau$
  while timer not expired
  If a valid acknowledgement from D is received
  Then stop the timer and return success
  Else
  Initial a trace message $trace$
  set timer $T = n\tau$
  while timer not expired
  If a valid response R_i is received
  Then stop timer and return $misbehaved(X_{i+1})$

- For every intermediate node $X_i(1 < i < n)$:
  X_i send to X_{i+1}: pkg_s or $trace$

- For every *probe* node $P_{ij}(1 < i < n, 1 < j < k)$:
  1. If overhear message forwarding from node X_i
  Then keep a record R_i and set timer $T = n\tau$
  2. While timer not expired
  If a *trace* message from S is received
  Then stop timer and broadcast record R_i
  $R = R_i$
  set timer $T = (n - i)\tau$
  While timer not expired
  If $R_j(i < j)$ is received
  Then $R = R_j$
  Broadcast R
  3. Discard the record

5.3 Locally Aware Reputation

We use the threshold cryptography scheme to sign the WARNING message about a misbehaved node. A (k,n) threshold scheme [35] divides the secret S into n pieces: $S1, S2, \cdots, Sn$ in such a way that:

- knowledge of any k or more Si pieces makes S computable;
- knowledge of any $k - 1$ or fewer Si pieces leaves S completely undetermined (in the sense that all its possible values are equally likely).

Our solution makes use of the polynomial secret sharing [35] and uses a (n,m) threshold scheme, where n is the number of one-hop neighbor nodes and m is the upper bound of the malicious node in a one-hop neighborhood. Each node X holds a personal RSA key pair, $< SK_X, PK_X >$, where SK_X is X's private/ secret key and PK_X is X's public key. The secret key of node X, specifically the signing key SK_X, is shared among all X's n one-hop neighbor nodes to a random polynomial of order $m-1$, where $m-1$ is the upper bound of malicious nodes in the one-hop neighbor nodes

of X. However, SK_X is not visible, known or recoverable by any node.

In our design, a node M is considered untrustworthy if any m one-hop neighbor nodes claim so. Since SK_M is shared among n one-hop neighbors of M, any such secret share holders notice M's misbehavior can sign the WARNING message. A WARNING message signed by SK_M can be verified by the well-known public key PK_M.

For each one-hop neighbor of node M, if it finds out that M's reputation value drops below the untrustworthy threshold, then it will generate a WARNING message about M and a "partial" signature on the message by applying its share of SK_M. It then broadcasts the WARNING message along with the partial signature. Any node that receives the same WARNING message from m different nodes can verify the validity of the message by combining m partial signatures together to generate the full signature.

If the WARNING message is verified, it is then broadcasted to the k-hop neighborhood, thus M's k-hop neighbors become aware of its misbehavior and refuse to server for it.

5.4 Simulation and Performance analysis

The objective of simulation and performance analysis is to determine the impact of Locally Aware Reputation System (LARS) on metrics as described below in a mobile ad hoc network where a part of the nodes act uncooperative. We will simulate the protocol using entity mobility models, assuming that in group mobility models, nodes in a group will cooperation with each other without any enforcement.

For all the metrics, we want to investigate the scalability in terms of number of nodes, fraction of uncooperative nodes, and mobility. Using DSR [33] as a reference, we will consider the following performance metrics of LARS:

- **Data Packet Delivery Ratio:** ratio of the number of data packets delivered to the destination nodes divided by the number of data packets transmitted by the source nodes. The data delivery ratio is directly influenced by packet loss, which may be caused by general network faults or uncooperative behavior.
- **Protocol Overhead:** the ratio of number of extra message (such as *trace* and record replied in the trace process, WARNING messages) divided by number of all the message transmitted.
- **End-to-end Delay:** the time needed to send a packet successfully from the source node to the destination node. The highest, lowest and average end-to-end delay will be considered.

DSR [33] is an on demand source routing protocol for mobile ad hoc network. Each packet carries the full path (a list of intermediate nodes) that the packet should be able to traverse in its header. A route to a destination is requested only when there is data to send to that destination, and a route to that destination is unknown or expired.

The simulation will be implemented on GloMoSim [38], a library-based sequential and parallel simulator for wireless mobile ad-hoc networks. We will first analyze a regular well-behaved DSR network; second, we introduce some uncooperative behavior to the regular DSR network and analysis the performance; then we enhanced DSR with the LARS protocol and compare the performance with the regular DSR.

Different mobility models will be used in the simulation.

References

[1] E.M.Belding-Royer and C.K.Toh. A review of current routing protocols for ad-hoc mobile wireless networks. IEEE Personal Communications Magazine, pages 46-55, April 1999.

[2] C. E. Perkins, Ad Hoc Networking, Addison-Wesley, 2000.

[3] L.Zhou and Z. Hass. Securing ad hoc networks. IEEE Network, 13(6), pages 24-30, November/December 1999.

[4] C.E. Pekins and P.Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing, Proceedings of INFOCOM '97, April 1997.

[5] Charles E. Perkins and Elizabeth M. Royer, Ad hoc On-Demand Distance Vector Routing, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100.

[6] H. Luo, P.Zerfos, J.Kong, S. Lu and L.Zhang, Self-securing Ad Hoc Wireless Networks, 7th IEEE Symposium on Computers and Communications (ISCC'02), July 2002, Italy.

[7] L.Buttyan and J.-P. Hubaux, Enforce Srvice Availability in Mobile Ad-Hoc WANs,In proceedings of MobiHoc, 2000

[8] Jean-Pierre Hubaux, Levente Buttyan, Srdjan Capkun, The Quest for Security in Mobile Ad Hoc Networks, Proceedings of the 2001 ACM International Symposium on Mobile ad hoc networking and computing, 2001

[9] J.-P. Hubaux and L.Buttyan, Toward Mobile Ad-Hoc Wans: Terminodes, Technical Report No. DSC/2000/006, Swiss Federal Institute of Technology, Lausanne, July 2000.

[10] L.Buttyan and J.-P. Hubaux, Stimulating Cooperation in Self Organizing Mobile Ad Hoc Networks, Technical Report No. DSC/2001/046, Swiss Federal Institute of Technology, Lausanne, August 2001.

[11] L. Buttyan, J.-P. Hubaux, Nuglets: a Virtual Currency to Stimulate Cooperation in Self Organized Mobile Ad Hoc Networks, Technical report No. DSC/2001.

[12] Naouel Ben Salem, Levente Buttyan, Jean-Pierre Hubaux, Markus Jakobsson, A charging and rewarding scheme for packet forwarding in multi-hop cellular networks, in proceedings of MobiHoc 2003, pp. 13-24.

[13] Hugo Miranda and Luis Rodrigues, Preventing Selfishness in Open Mobile Ad Hoc Netwroks, In Proceedings of 7th CaberNet Radicals Workshop, Bertinoro, Forli, Italy, October 2002.

[14] Sonja Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes — Fairness In Dynamic Ad-hoc NeTworks. In Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), Lausanne, CH, June 2002.

[15] Sonja Buchegger, Jean-Yves Le Boudec, Coping with False Accusations in Misbehavior Reputation Systems for Mobile Ad-hoc, EPFL Technical Report IC/2003/31.

[16] Sonja Buchegger, Jean-Yves Le Boudec, A Robust Reputation System for P2P and Mobile Ad-hoc Networks, In Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems,2004.

[17] Pietro Michiardi, Refik Molva, Core: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks, IFIP-Communicatin and Multimedia Securtiy Conference 2002.

[18] S. Bansal and M. Baker, Observation-based Cooperation Enforcement in Ad Hoc Networks,http://arxiv.org/pdf/cs.NI/0307012, July 2003.

[19] George Theodorakopoulos and John S. Baras, Trust Evaluation in Ad Hoc Networks, In Proceedings of the 2004 ACM workshop on Wireless security, 2004.

[20] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In Proceedings of the 1996 IEEE Symposium on Security and Privacy, pages 164-173, 1996.

[21] S. Buchegger and J. Y. Le Boudec. The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In Proceedings of WiOpt '03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Sophia-Antipolis, France, March 2003.

[22] R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, pages 229-242, Jan. 1998.

[23] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In DARPA Information Survivability Conference and Exposition, January 2000.

[24] Sergio Marti, T.J.Giuli, Kevin Lai, and Mary Baker, Mitigating routing misbehavior in mobile ad hoc networks, in proceedings of MOBICOM 2000, pages 255-265, 2000.

[25] Pietro Michiardi, Refik Molva, Game theoretic analysis of security in mobile ad hoc networks, Research Report RR-02-070, April 2002.

[26] Pietro Michiardi and Refik Molva,Prevention of Denial of Service Attacks and Selfishness in Mobile Ad Hoc Networks, Reseach Report RR-02-063, January 2002.

[27] Vikram Srinivasan, Pavan Nuggehalli, Carla-Fabiana Chiasserini and Ramesh Rao, Cooperation in Wireless Ad Hoc Networks, in Infocom 2003.

[28] P. Yau and C. J. Mitchell, Reputation methods for routing security for mobile ad hoc networks, in: Proceedings of SympoTIC '03, Joint IST Workshop on Mobile Future and Symposium on Trends in Communications, Bratislava, Slovakia, October 2003, IEEE Press, 2003, pages 130-137.

[29] P. Yau and C. J. Mitchell. Security vulnerabilities in ad hoc networks. In The Seventh International Symposium on Communication Theory and Applications, July, 2003,

Ambleside, Lake District, UK, HW Communications Ltd, July 2003, pages 99-104.

[30] Sonja Buchegger, Jean-Yves Le Boudec, Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks, in 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands, Spain, January 2002.

[31] Sheng Zhong, Jiang Chen, and Yang Richard Yang, Sprite: A simple, Cheat-proof, Credit-based System for Mobile Ad hoc Networks, in Proceedings of IEEE Infocom '03, San Francisco, CA, April 2003.

[32] Pietro Michiardi, Refik Molva, Game theoretic analysis of security in mobile ad hoc networks, Research Report RR-02-070, April 2002.

[33] Dave B. Johnson and David A. Maltz. The dynamic source routing protocol for mobile ad hoc networks. Inter- net Draft, Mobile Ad Hoc Network (MANET) Working Group, IETF, October 1999.

[34] Yongwei Wang, Venkata C. Giruka, Mukesh Singhal, A Fair Distributed Solution for Selfish Nodes Problem in Wireless Ad Hoc Networks,Ad-Hoc, Mobile, and Wireless Networks: Third International Conference, ADHOC-NOW 2004, Vancouver, Canada, July 22-24, 2004. Proceedings Pages 211-224.

[35] A. Shamir, How to share a secret, Communications of ACM, 1979.

[36] Bruce Schneier, Applied Cryptography - Protocols, Algorithms, and Source Code in C, John Wiley & Sons, 1996.

[37] Yvo Desmedt, Threshold Cryptography, European Trans. on Telecommunications, 5(4), pages 449-457, July-August 1994.

[38] Xiang Zeng, Rajive Bagrodia, and Mario Gerla, GloMoSim: A library for parallel simulation of large-scale wireless networks, Proceedings of the 12th Workshop on Parallel and Distributed Simulations, PADS '98, May 26-29, in Banff, Alberta, Canada, 1998.