Reputation-based System for Encouraging the Cooperation of Nodes in Mobile Ad Hoc Networks

Tiranuch Anantvalee and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, Florida 33431
Email: {tanantva@, jie@cse.}fau.edu

***Abstract*— In a mobile ad hoc network, node cooperation in packet forwarding is required for the network to function properly. However, since nodes in this network usually have limited resources, some selfish nodes might intend not to forward packets to save resources for their own use. To discourage such behavior, we propose a reputation-based system to detect selfish nodes and respond to them by showing that being cooperative will benefit them more than being selfish. In this paper, besides cooperative nodes and selfish nodes, we introduce a new type of node which is a suspicious node. These suspicious nodes will be further investigated and if they tend to behave selfishly, we could take some actions against them like we do to selfish nodes to encourage them to be cooperative. We introduce the use of a state model to decide what we should do or respond to nodes in each state. In addition, we introduce the use of a timing period to control when the reputation should be updated and to use as a timeout for each state.**

Keywords: Ad hoc networks, cooperation, reputation, selfish

I. Introduction

A mobile ad hoc network (MANET) is a self-configuring network that is formed automatically by a collection of mobile nodes without the help of a fixed infrastructure or centralized management. In order for a node to communicate to a node that is out of its radio range, the cooperation of other nodes in the network is required; this is known as multi-hop communication. MANETs was originally developed for military purposes which all nodes belong to the same authority which they all must cooperate to achieve their goals. Currently, MANETs have been developing rapidly and are increasingly being used in many civilian applications since setting up such networks can be done without the help of any infrastructure or interaction with a human. However, each of these devices belongs to different authorities or belongs to individuals. In such networks, we cannot always assume that they cooperate. Thus, they might intend not to cooperate in order to disrupt the network or to save batteries for their own uses.

The simulations in [1], [5] showed that the presence of only a few misbehaving nodes can dramatically degrade the performance of the entire system. Much research has been done to cope with this cooperation problem. The research can be divided into three main categories: secure routing (such as Ariadne [8], SRP [9], and SEAD [10]), economic incentives (such as Nuglets [12] and SPRITE [11]), and reputation-based system (such as Watchdog and Pathrater [1], CONFIDANT [5], CORE [2], OCEAN [3], and SORI [4]).

This work was supported in part by NSF grants ANI 0073736, EIA 0130806, CCR 0329741, CNS 0422762, CNS 0434533, and CNS 0531410.

In this paper, we address the problem of selfish nodes by using a reputation approach called reputation management system (RMS). The main purpose of this system is to detect and punish selfish nodes. By punishing these nodes, we want to show that being selfish will not benefit them. Instead, being cooperative has a better chance to increase their benefit. This RMS can be used as an extension to any source routing protocols, and we chose dynamic source routing (DSR) for our simulation implementation.

An RMS runs independently on every node and monitors transmissions that it can overhear, even ones it is not involved in, which is known as a watchdog mechanism [1]. Reputation values are given according to the results from the watchdog and reputation regarding one node is different from one neighbor to another. Therefore, RMS allows the propagation of these reputation values. Unlike other previous works, we limit the propagation to only when it is necessary, e.g., when it is requested or when detecting selfish nodes. If no selfish behavior is detected, no information is propagated. This results in less transmissions, less overhead, and less energy consumptions.

The propagation of reputation values may be used to attack a legitimate node by accusing it as a selfish node. To alleviate this problem, only second-hand information that is compatible with a node's own information will be used. Even if the information is used, it should only slightly influence the resulting reputation values. Once we determine that a node is selfish, the punishment is carried out to penalize the selfish node. To receive a better service, we also avoid routing through selfish nodes by selecting a route without them. In addition, to obtain only routes free of them, selfish nodes may be added to an avoid list, an additional field, when sending a route request.

When using reputation, most of the existing systems use a binary threshold to determine whether a node is well-behaving or selfish. In our system, we use two thresholds to categorize nodes into three categories. If the reputation value is above the first threshold or below the second threshold, it is *cooperative* or *selfish*, respectively. The third type of node we introduce in this paper is a *suspicious* node, whose reputation value falls between two thresholds. We cannot determine right away whether they are cooperative or selfish, so they should be

further investigated. If they tend to behave selfishly, we also take some actions against these suspicious nodes to encourage them to be cooperative.

In addition to using two thresholds, we also introduce the use of a state model to determine the states of other nodes and to decide what should be done in each state. What state a node is put into depends on its reputation rating, which is compared against two thresholds, and its previous state. Moreover, we also introduce the use of a timing period to control when the reputation should be updated and to be used as a timeout for each state.

The remainder of this paper is organized as follows. Section II provides related works in node cooperation in MANETs using reputation approach. The assumptions and overview of the proposed system is presented in Section III. The detailed description and the detection mechanism are presented in Section IV and V respectively, followed by the simulation result which was implemented on the JiST/SWANS simulator in Section VI. Then, the conclusion is drawn in Section VII.

II. Related Works

Marti, *et al.* [1], proposed two techniques, *watchdog* and *pathrater*. The watchdog promiscuously listens to the transmission of the next node in the path for detecting misbehavior. The pathrater keeps the ratings for other nodes performs route selection by choosing routes that do not contain selfish nodes, and have the highest average node rating. However, these selfish nodes are not punished, but rather rewarded, as their packets continue to be forwarded by others while they do not have to forward for any node.

Michiardi and Molva [2] proposed CORE, which also uses the watchdog mechanism to observe neighbors, and aims to detect and isolate selfish nodes. The node reputation is heavily weighted towards past reputation, therefore, cooperative nodes with low battery condition would not be detected as misbehaving nodes right away. However, only positive indirect reputation is allowed in this system to avoid false accusation and denial of service attacks. In our system, a node receives indirect reputation from other nodes, but the decision whether or not to use it depends on the deviation from direct information.

Bansal and Baker [3] proposed an extension to the DSR protocol called OCEAN, which also considers selfish behavior. Each node maintains the ratings for neighbors who directly interact with it. To avoid trust management complexity and false accusation, these ratings are not propagated to any other node. A neighbor whose reputation value is less than the faulty threshold is put into a faulty list, whose traffic will be rejected.

SORI, proposed by He, Wu and Khosla [4] also addressed the problem of selfish nodes. The first-hand reputation is based on the ratio of the number of packets sent to be forwarded to the number of packets that have been forwarded. This reputation is updated periodically and is broadcast to neighbors only when it significantly changes. When combining, the second-hand reputation is weighted by credibility, which is the first-hand reputation of the sender node. As a punishment to selfish nodes, packets originating from selfish nodes are probabilistically dropped.

Buchegger and Le Boudec [5] proposed CONFIDANT, an extension to the source routing protocol. CONFIDANT deals not only with selfish behavior, but also several types of misbehavior such as silent route change or frequent route updates. When misbehaving nodes are detected, it sends an alarm to other nodes in the network, defined as friends, to isolate misbehaving nodes from the network. This original CONFIDANT requires predetermined trust to evaluate the trustworthiness of the received alarms. The authors have improved CONFIDANT with an adaptive Bayesian reputation and trust system [6]. In [6], the first-hand reputation is locally broadcast to exchange information with neighbors periodically. In contrast, our system only broadcasts reputation when requested or when selfish nodes are detected. [6] also involves a trust metric, which is a measure of the node's previous behavior in reporting second-hand information. Our system uses a similar trust metric.

III. Reputation Management System (RMS)

A. Assumptions

The system model presented in this paper is based on the following assumptions.

- Each node has a unique id and it cannot be spoofed.
- The network is dense enough so that each node has at least two one-hop neighbors.
- A wireless interface of each node supports promiscuous mode operation: a node always listens to every transmission within its one-hop neighborhood although it doesn't involve in those transmissions.
- Links are bidirectional. At time t, if node B can receive a message from node A, node A should be able to receive a message from node B at time t as well.
- An antenna used on each node is an omni-directional antenna which enables its transmission to be monitored by its one-hop neighbors.
- Each node is independent from each other, no collusion.
- We do not consider malicious nodes, only selfish nodes seeking to conserve their own resource.

B. RMS Overview

The reputation management system (RMS) is an extension to the source routing protocol which runs independently on each node. Each node (a monitoring node) hosts a watchdog to monitor the forwarding behaviors of its neighbors (monitored nodes), then assigns the reputation values (ratings) for each of them according to the observed behaviors. We will refer to this reputation value as the reputation value in forwarding packets (RF). This RF value should be updated periodically to reflect the current behavior of a node. Thus, the timing period (τ) is introduced to serve this purpose so that the RF value is updated only at the end of each period. Additionally, when updating, it should take into account the past reputation to avoid false detections due to link breaks or the lack of resources.

Since RMS runs independently on each node, an RF value regarding one node is different from one neighbor to another.

Because of this, RMS allows the propagation of this RF value. Unlike [6] which publishes the reputation periodically, we limit the propagation to only when it is necessary, e.g. when it is requested, or when a node is being investigated. In addition, only reputations of requested nodes or detected nodes are propagated to the requestors or neighbors, respectively. Moreover, since we use a timing period to control when to update the RF value, the propagation, if any, will occur only at the end of the period after the update as well. Obviously, this results in less transmissions, less overhead, and less energy consumptions.

However, this propagation may be used to attack a legitimate node by accusing it as a selfish node. To alleviate this problem, the decision whether or not to allow the received RF to be used in calculating the final RF value is made by first comparing the received rating with the node's own RF rating. If the comparing result is in the acceptable range, then the new value will be integrated. In addition, even if it's used, it should only slightly affect the resulting RF value because we always believe in our own observation more than others' observation.

In the real world, when we receive recommendations from several persons, which recommendation we believe more depends on that person's reputation in telling the truth. Similarly, after deciding which received RF values we will use, we will also weigh which received RF should have more effect on our resulted RF. Thus, we introduce the second reputation value: the reputation in being a referrer (RR). This reputation value represents how trustworthy a referring node is, comparing to other referring nodes, when giving the information regarding other nodes. This RR value changes when a received RF value is compared with the existing RF value. If the comparing result is in the acceptable range, RR is increased, otherwise, it is decreased.

Generally, the RF value can represent the forwarding behavior of a monitored node. Whether it is considered cooperative or selfish depends on its RF value, compared against a predefined threshold (TH). Basically, if $RF < TH$, it means that the monitored node behaves selfishly and should be punished, otherwise it's well-behaving. Using one threshold may be too harsh to determine the behavior of the node. Therefore, in our system, we introduce the use of two thresholds: TH_{coop} and $TH_{selfish}$. If $RF \geq TH_{coop}$ or $RF < TH_{selfish}$, it is straightforward that the monitored node is cooperative (`cooperative` state) or selfish (`punished` state since it should be punished) respectively. However, if $TH_{selfish} \leq RF < TH_{coop}$, the behavior of the monitored node cannot be determined immediately. We categorize these nodes as suspicious nodes. For example, a monitoring node might misunderstand a monitored node due to the limitations described in [1], so the RF value of the monitored node is below TH_{coop}. Another possible reason is, it might forward only the necessary amount of packets to stay above $TH_{selfish}$. Therefore, punishment should not be limited to only the monitored nodes whose RF values are below $TH_{selfish}$, but also for monitored nodes whose RF values fall between these two thresholds for more than a certain period of time. Nonetheless, the level of punishment should be different from the node whose RF value is below $TH_{selfish}$. For these reasons, there should be more states than just `cooperative` and `punished`. As a solution, we introduce the use of the state model to control the state transition of a monitored node based on the comparison of the RF value against two thresholds. Note that the state is determined only after the update of the RF value, which occurs at the end of period only.

When a node is considered to be selfish, it is put into the `punished` state. As described above, we also punish nodes who are suspicious and tend to behave selfishly, but with different levels of punishment. Thus, we use $PLEVEL$ to determine the punishment level which varies from 1 to 4 (maximum punishment). The selfish node will be given chances to behave cooperatively so that it can recover from the `punished` state. But if it continues to behave selfishly and is put into the `punished` state with $PLEVEL = 4$ for more than a certain number of times, it will be put into the `blacklisted` state and will never recover from this state. The punishment for nodes in the `blacklisted` state is like that for nodes in the `punished` state with $PLEVEL = 4$, except that none-`blacklisted` nodes can change their behavior in the `punished` state and recover to other states.

In addition, since states could tell us, to some degree, how nodes have behaved, we also use states to determine how we should respond to nodes in each state. Therefore, once a node is put into one state, it should stay in that state and should be monitored for some more periods of time. Since we already use the timing period (τ) to control when the RF value is updated, it will also be used to define how long or how many more periods (a multiplication of τ) the node should stay in one state before the next state of the node is determined.

IV. RMS Components

The reputation management system (RMS) is comprised of four main components: the monitoring module, the reputation manager, the response module and the communication module. These components reside on every node in the network.

A. The Monitoring Module

The monitoring module uses a watchdog mechanism to monitor the packet-forwarding behavior for each of its neighbors by keeping track of two counters for each neighbor. Let M be the monitoring node and m be the monitored node.

- $SBF(M, m)$ denotes the number of packets that should be forwarded by m observed by M
- $ABF(M, m)$ denotes the number of packets that have actually been forwarded by m observed by M

Whenever node m receives a packet which is supposed to be forwarded, either from node M, or from another neighbor and node M overhears the transmission, node M stores the packet in its buffer, sets the timeout, and increases the SBF counter by one. If node m forwards the packet, the packet is removed from the buffer and the ABF counter is increased by one. Otherwise, the packet is removed when time reaches the timeout.

These two counters for each neighbor are counted over a fixed timing period (τ). At the end of each period, the

monitoring module sends these counters to the reputation manager, resets them, and starts counting from zero again.

B. The Reputation Manager

The reputation manager is responsible for maintaining a reputation record for each neighbor by keeping them in the reputation table. Each field in the reputation record is described below.

- $NODE$: Node ID of a monitored node m
- RF: Reputation value in forwarding packets that the monitoring node has for the monitored node. The value range is [0,1]
- RR: Reputation value in being a referrer that the monitoring node has for the monitored node, the value range is [0,1]
- $STATE$: The state of a monitored node m determined by node M, which can be `cooperative`, `suspected`, `inspecting`, `punished`, and `blacklisted`
- $STATE\_TS$: The last updated time of $STATE$
- $STIME$: Time to determine the state of a monitored node after updating RF value
- $PLEVEL$: The level of punishment which is used only when $STATE$ is `punished`. The value is 1, 2, 3, or 4.
- $COUNT$: The number of times that the node has been put into `punished` state with $PLEVEL = 4$

At the end of every period ($t = t_0 + \tau, t_0 + 2\tau, t_0 + 3\tau, \ldots$, where t_0 is the time when a monitoring node joins the network), the reputation manager updates each reputation record according to the data received from the monitoring module (two counters) and the communication module (RF values from other nodes in REP_REP or ALARM, explained later in Section IV-C). The update process can be divided into three steps: 1) Calculating RF and RR values. 2) Determining $STATE$, $PLEVEL$, and $STIME$. 3) Informing the response module to punish selfish nodes and the communication module to send REP_REQ or ALARM when necessary. We will describe each step by assuming that M is the monitoring node and m is the monitored node.

1) Calculating RF and RR Values: As mentioned earlier, the RF value for each monitored node m can be obtained by directly monitoring the forwarding behavior of node m and by combining its RF value with the RF values from other nodes. In other words, the value of RF depends on inputs from two sources: 1) two counters from the monitoring module of the node itself and 2) the RF values received in REP_REP and ALARM messages from neighbors (via the communication module). Therefore, at the end of each period, if there is at least one input, the new RF value that node M has for node m, $RF(M, m)$, is calculated by using the formula shown below:

$$RF(M, m) = \beta \cdot RFO(m) + (1 - \beta) \cdot RFR(m) \tag{1}$$

where $RFO(m)$ is the reputation value regarding node m calculated from the information OBSERVED by the node itself. Whereas $RFR(m)$ is the reputation value regarding node m calculated from the information RECEIVED from other nodes, which is the RF values from REP_REP and ALARM messages. β is a self-belief factor for how much the node believes in itself and ranges from 0 to 1.

From Equation 1, if there are inputs from two sources, the new RF value should be calculated based on both inputs. In the case where there is only input from neighbors, the current RF value that is stored in the table, $RF_{table}(M, m)$, should be also brought into account when calculating the new RF value by using $RFO(m) = RF_{table}(M, m)$. This is because the new RF value should not be changed completely to what neighbors say. It should still be related to the current RF value that has been accumulated over time.

Let us now consider the value of $RFO(m)$. This value depends on the observation of the monitoring module, which sends those two counters for node m to the reputation manager. $RFO(m)$ is calculated by:

$$RFO(m) = \alpha \cdot RF_{table}(M, m) + (1 - \alpha) \cdot \frac{ABF(M, m)}{SBF(M, m)} \tag{2}$$

Since there could be nodes that have been cooperative for a long time but happen to be out of power, these nodes should not be mistaken for being selfish. For this reason, the current RF value, $RF_{table}(M, m)$, should not be entirely discarded, as it can represent the past behaviors of the node. α is introduced as a past behavior factor which is used to judge nodes based on past behaviors. Hence, the more we want to give priority to the past behaviors, the more the value of α should be. However, when there is no existing record for m, the value of $RFO(m)$ would be only the ratio of $ABF(M, m)$ to $SBF(M, m)$.

Now, let us consider the value of $RFR(m)$. When combining the received RF values, the RR values of the nodes that sent those RF values should be brought into consideration. The RR value represents how trustworthy a referring node is, compared to other referring nodes, when giving information regarding other nodes. $RFR(m)$ is calculated by:

$$RFR(m) = \frac{\sum_{i=1}^{n} (RR(M, N_i) \cdot RF(N_i, m))}{\sum_{i=1}^{n} RR(M, N_i)} \tag{3}$$

where n is the number of neighboring nodes who sent the REP_REP or ALARM messages containing RF value regarding node m, $RF(N_i, m)$, to node M. Note that RF values in ALARM messages will be used in Equation 3 at the end of whichever period they are received, but RF values in REP_REP messages will be used only when time reaches $STIME$ and the $STATE$ of node m is `suspected`, as explained later in Section V.

However, the choice of whether or not to include the received RF value in Equation 3 depends on the value of $\Delta RF(N_i)$ which is calculated by:

$$\Delta RF(N_i) = |RF(N_i, m) - RF_{table}(M, m)| \tag{4}$$

If $\Delta RF(N_i) < \delta$, the $RR(M, N_i)$ will be increased by γ and the received $RF(N_i, m)$ will be used in Equation 3. On the contrary, if $\Delta RF(N_i) \geq \delta$, the $RR(M, N_i)$ will

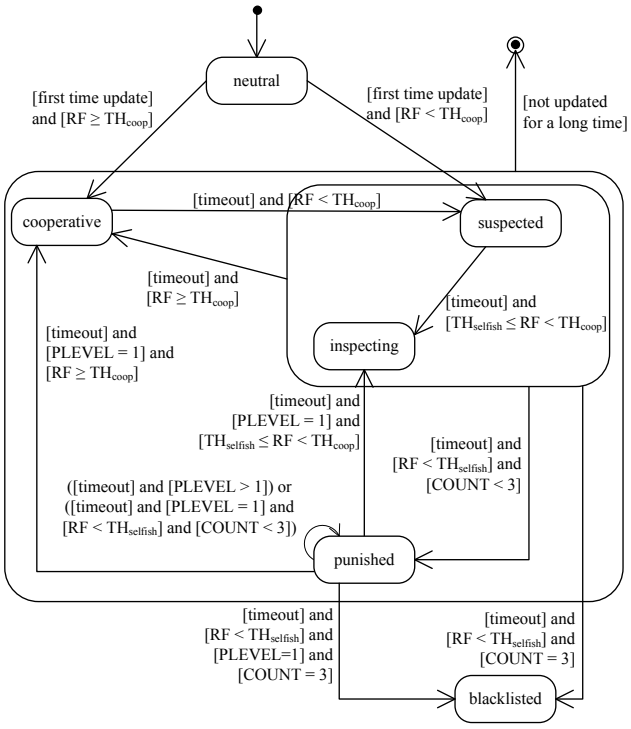

Fig. 1. State diagram of a monitored node

be decreased by γ and the received $RF(N_i, m)$ will be disregarded. However, if node M never had a reputation record for node m before, M will ignore all received ALARMs regarding m since M has no $RF_{table}(M, m)$ to compare with the received $RF(N_i, m)$.

2) Determining $STATE$, $PLEVEL$, and $STIME$: After calculating the new value of $RF(M, m)$, the state of node m will be determined according to this new value. A node can be in one of the following five states: `cooperative`, `suspected`, `inspecting`, `punished`, and `blacklisted`. As mentioned earlier, two thresholds are used to determine which state the node is in. Figure 1 illustrates the state diagram of a monitored node. Note that timeout means when time reaches $STIME$.

If there is no reputation record for node m, the $STATE$ of node m is always considered as `neutral`. Once there are counters from the monitoring module for m, at the end of that period, m will be put into `cooperative` or `suspected` according to its RF value. But if there is a reputation record for node m in the reputation table, the $STATE$ of node m is determined only when time reaches $STIME$. Assume that the time at the end of the current period is t_{20} and the $STIME$ is also t_{20}. Table I shows an example of how the next $STATE$ of a monitored node is determined and how the value of $PLEVEL$ and $STIME$ are changed.

$STIME$ is used as a state timeout to keep the monitored node in one $STATE$ for a certain period of time. For example, when the state of a node is `inspecting`, if $RF(M, m)$ is between TH_{coop} and $TH_{selfish}$, node m will be put into `punished` with $PLEVEL = 1$ for 10 more periods ($t_{20} + 10\tau$). During that time, node M continues monitoring node m. When time $t = t_{20} + 10\tau$, after the RF of node m is calculated according to Section IV-B.1 above, the $STATE$ of node m is then determined.

3) Informing the Response Module and Communication Module: Once the reputation manager has finished updating all reputation records, it then checks the reputation table to see whether any node whose $STATE$ has just been changed to `suspected`, or `punished` or `blacklisted` within that period (i.e., from the previous example, $STATE\_TS = t_{20}$).

- The reputation manager will inform the response module of nodes whose states are changed to `punished` (or whose $PLEVEL$ is changed) or `blacklisted`, so these nodes will be responded to accordingly.
- The reputation manager will inform the communication module to send REP_REQ or ALARM to its neighbors for nodes whose states are changed to `suspected` or `punished` with $PLEVEL = 4$, respectively.

C. The Communication Module

The communication module acts as an interface for the RMS to communicate to neighbors' RMS. The main purpose of this module is to exchange reputations with immediate neighbors through three types of messages:

- **REP_REQ Message**: This message is sent to neighbors in order to ask for RF values for nodes whose states have changed to `suspected`. Upon receiving REP_REQ, the communication module asks the reputation manager to return the RF values of nodes in REP_REQ.
- **REP_REP Message**: When the communication module receives the returned RF values from the reputation manager, it constructs the REP_REP message and sends it back to the requestor node. Upon receiving the REP_REP, the communication module forwards the received RF values to the reputation manager.
- **ALARM Message**: This message is sent to neighbors when there is a node whose state is changed to `punished` with $PLEVEL = 4$. Upon receiving ALARM, the communication module forwards the received RF values to the reputation manager.

D. The Response Module

The response module participates in route selection and route discovery processes. When there are several paths to a destination, the response module helps to select a route that has the best average reputation value and free of selfish nodes, `blacklisted` or `punished` with $PLEVEL = 4$ nodes. For nodes whose $PLEVEL$ are 1, 2, or 3, we still route packets through these nodes and observe them to give them chances to change their behaviors.

The response module also takes part in a route discovery process. When a node sends out a route request, it will append to the message header an avoid list, an additional field containing `blacklisted` and `punished` with $PLEVEL = 4$ nodes. If a node receives a route request with an avoid list containing itself, it then does not forward that route request. Thus, a route reply received at the source node will not contain

TABLE I
STATE TRANSITION OF A MONITORED NODE

current				next			
$STATE$	$PLEVEL$	$COUNT$	RF	$STATE$	$PLEVEL$	$COUNT$	$STIME$
neutral	-	any	$RF \geq TH_{coop}$	cooperative	-	same	$t_{20} + \tau$
neutral	-	any	$RF < TH_{coop}$	suspected	-	same	$t_{20} + \tau$
cooperative	-	any	$RF \geq TH_{coop}$	cooperative	-	same	$t_{20} + \tau$
cooperative	-	any	$RF < TH_{coop}$	suspected	-	same	$t_{20} + \tau$
suspected	-	any	$RF \geq TH_{coop}$	cooperative	-	same	$t_{20} + \tau$
suspected	-	any	$TH_{selfish} \leq RF < TH_{coop}$	inspecting	-	same	$t_{20} + 5\tau$
suspected	-	3	$RF < TH_{selfish}$	blacklisted	-	same	∞
suspected	-	< 3	$RF < TH_{selfish}$	punished	4	$+1$	$t_{20} + 30\tau$
inspecting	-	any	$RF \geq TH_{coop}$	cooperative	-	same	$t_{20} + \tau$
inspecting	-	any	$TH_{selfish} \leq RF < TH_{coop}$	punished	1	same	$t_{20} + 10\tau$
inspecting	-	3	$RF < TH_{selfish}$	blacklisted	-	same	∞
inspecting	-	< 3	$RF < TH_{selfish}$	punished	4	$+1$	$t_{20} + 30\tau$
punished	4	any	any	punished	3	same	$t_{20} + 10\tau$
punished	3	any	any	punished	2	same	$t_{20} + 10\tau$
punished	2	any	any	punished	1	same	$t_{20} + 10\tau$
punished	1	any	$RF \geq TH_{coop}$	cooperative	-	same	$t_{20} + \tau$
punished	1	any	$TH_{selfish} \leq RF < TH_{coop}$	inspecting	-	same	$t_{20} + 5\tau$
punished	1	3	$RF < TH_{selfish}$	blacklisted	-	same	∞
punished	1	< 3	$RF < TH_{selfish}$	punished	4	$+1$	$t_{20} + 30\tau$

such nodes. Another case is when a node receives a route request originated from selfish nodes. Since the punished state is divided into four levels, the response module may take different actions for nodes in each level. When receiving their requests, only some proportion, depending on the $PLEVEL$ of the requestors, of their requests are forwarded and replied to. The response for nodes in blacklisted state is like that for nodes in punished state with $PLEVEL = 4$. In our implementation, for example, 75%, 50%, and 25% of their requests are forwarded and replied to when $PLEVEL$ is 1, 2, and 3, respectively. For $PLEVEL = 4$ and blacklisted nodes, all of their requests are not forwarded nor replied to.

V. STATE FLOW: DETECTION MECHANISM

In this section, refer to Figure 1 and Table I, we will describe how the state is changed from one state to another and what being done in each state. Our implementation in the next section is also based on Figure 1 and Table I.

Whenever node m becomes known to node M, as an intermediate node in a route to a destination, as a neighbor, or as a node who requests for a service, it is positively considers to be cooperative and treated like it is in cooperative state. However, its state is neutral since there is no reputation record for m. The state of m changes, to either cooperative or suspected, only when $RF(M, m)$ is calculated based on two counters from the monitoring module.

If m is in the cooperative state, it is cooperative and nothing needs to be done. If m is in the neutral state, it is also treated as if it is in the cooperative state. Whenever its RF value falls below TH_{coop}, it'll be put in the suspected state in order to gather more information from neighbors.

When m is put into the suspected state, M sends out REP_REQ and waits for REP_REP. Once the end of a period reaches $STIME$, M combines received RF values in REP_REP with its own RF value regarding m. The combined result is then used to determine the state of m according to Table I.

In the inspecting state, M is doing the further inspection, for example, whether m is trying to forward only the necessary amount of packets just to be above $TH_{selfish}$. When this inspecting period comes to an end and m's RF value is still in between the two thresholds, it will definitely be punished by putting it into the punished state with $PLEVEL = 1$. Otherwise, its state will be changed to cooperative or punished with $PLEVEL = 4$ according to the RF value.

In the punished state, m is considered selfish and is being punished. In this state, $PLEVEL$ is used to define different levels of punishments or responses. In our implementation in the next section, when $PLEVEL = 4$, m will be excluded from all network activities: avoiding routing packets through m, ignoring requests from m, and adding m to an avoid-list when sending a route request. Although m is offered a chance to rejoin the network after a predefined period by using $STIME$, it is first allowed to rejoin the network with some limitation, so it is still in the punished state with different $PLEVEL$ which is decreased gradually. Since $STIME$ is used as a timeout, once the time reaches $STIME$, $PLEVEL$ is decreased and the new $STIME$ is set. When $PLEVEL = 1$ and the time reaches $STIME$, m's state is determined according to its RF value as shown in Table I.

Whenever m is put into the punished state with $PLEVEL = 4$, $COUNT$ is increased by 1. In our implementation, we allow a node to be put into the punished state with $PLEVEL = 4$ only three times. If node m's RF is less than $TH_{selfish}$ for the fourth time, m will be put into the blacklisted state and it will never be given any chance to recover from this state.

VI. SIMULATION

To evaluate the performance of our system, we implemented RMS using JiST/SWANS simulator [13], [14]. We integrated

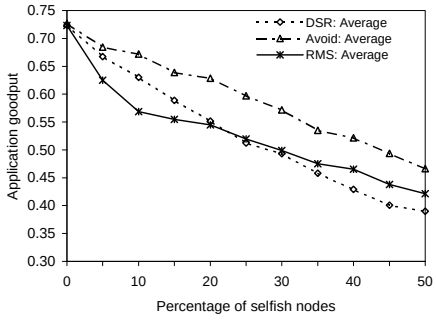

Fig. 2. Application goodput of all nodes

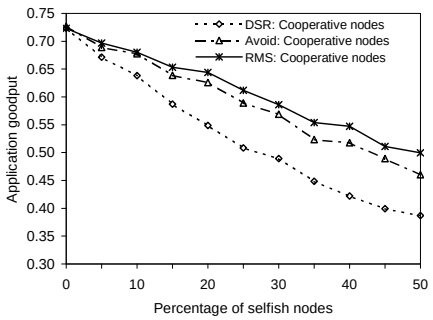

Fig. 3. Application goodput of cooperative nodes

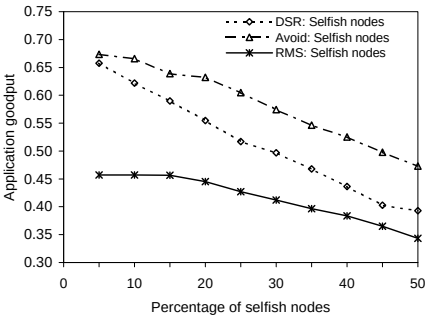

Fig. 4. Application goodput of selfish nodes

RMS as an extension to the Dynamic Source Routing protocol (DSR). The results are compared against the regular DSR network without our extension. Note that in our simulation, we assume that RMS messages are sent out correctly as specified. They are not used to attack the legitimate nodes.

Metrics used in our simulation results are:

- **Application Goodput**: This is the average ratio of the total number of packets received at destination to the total number of packets intended to send, at the application level, by all nodes. The application goodput of cooperative nodes means the average ratio of the total number of packets received at destination to the total number of packets intended to send at the application level by cooperative nodes only. The application goodput of selfish nodes is defined in the same manner.
- **Routing Goodput**: This is the average ratio of the total number of packets received at destination to the total number of packets originated by all nodes at the routing layer. Unlike the application goodput, packets are counted only when they are being sent out. If there is no available route to the destination, they are not counted.
- **Number of Packets Dropped**: This is the average number of packets dropped by selfish nodes taken from 20 simulation runs.

A. Simulation Setup

The simulated network consists of 50 wireless nodes deployed in a field of 1000 x 1000 square meters. The random waypoint is chosen as a mobility model. Each node is first randomly placed in the field, waits for the pause time (0 second in our simulation), then moves to another random position with a speed chosen between 1 to 10 m/s. Every 10 seconds during the simulation, ten new source and destination pairs are randomly selected, therefore, every node has chances to be both a source and a destination. The constant bit rate (CBR) is selected as our traffic model with a rate of 4 packets per second. The run time for each simulation is 1000 seconds. The parameters used for our RMS extension in the simulations are: $\tau = 5$ second, $\alpha = 0.8$, $\beta = 0.8$, $\delta = 0.4$, $\gamma = 0.05$, $TH_{coop} = 0.4$, and $TH_{selfish} = 0.1$. The results presented here are the average from 20 simulation runs with different seeds. The seeds influence the placement and the movement of the nodes, and the selection of source-destination pairs.

B. Simulation Results

For the simulation results, we graph three curves for DSR, Avoid and RMS. (1) ***DSR***: This is a result for the original DSR with no extension. (2) ***Avoid***: This is a result for DSR with RMS enabled. However, no reputation exchange or punishment mechanism is executed. RMS only helps the source node to select a route that has the best average reputation value and does not contain nodes whose states are `blacklisted` or `punished` with $PLEVEL = 4$. (3) ***RMS***: This is a result for implementing all of our extensions to the original DSR. Besides avoiding routing through selfish nodes, it also does not forward or reply to requests from them. Nodes also exchange RMS messages through their communication modules. In addition, an avoid list containing `punished` with $PLEVEL = 4$ nodes is added to the DSR header when sending a route request. To limit the size of the header, a maximum of five nodes can be added in the avoid list in the simulation.

Figure 2-4 shows the comparison of the application goodput of the original DSR and the DSR with our RMS extensions when the percentage of selfish nodes varies from 0 to 50 percent. When nodes only avoid routing through `punished` nodes with $PLEVEL = 4$, the application goodput for all nodes increases including that of the selfish. Although this results in the best application goodput for the overall system, not only are selfish nodes not punished, they do not have to relay traffic as they are avoided.

When we implement all of our extensions to DSR, not only are selfish nodes avoided, requests from them are not forwarded nor replied to. Thus, their packets could not be sent out since there is no route to the destination. Although the application goodput of all nodes is decreased when using our extension, Figure 2, 3 and 4 show that the application goodput of cooperative nodes is increased while that of selfish nodes is noticeably decreased. The differences in the application goodput of cooperative nodes and that of selfish nodes are 0.25 and 0.15 when the percentages of selfish nodes are 5% and 50%, respectively. Since being selfish is a worse strategy than being non-selfish, an intelligent player will take the dominant strategy of being non-selfish.

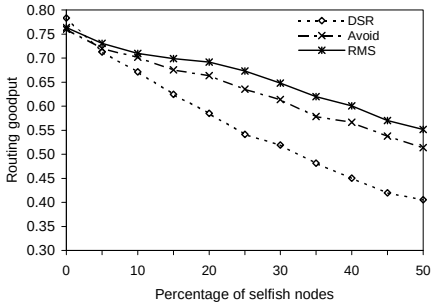

Fig. 5. The routing goodput

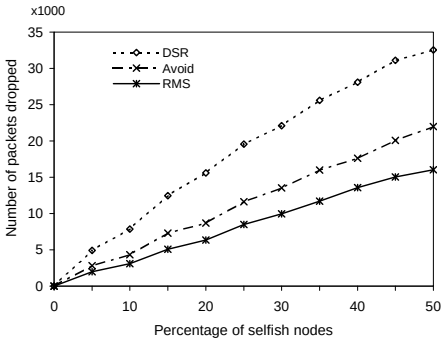

Fig. 6. The number of packets dropped by selfish nodes

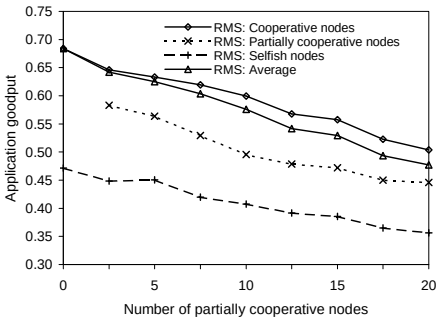

Fig. 7. Application goodput of nodes in the presence of 5 selfish nodes

Figure 5 shows the result of the routing goodput. Once the packets leave the sender nodes, the chances that they will be received at the destination nodes is increased when using our extension. The routing goodput is improved when avoiding routing through selfish nodes, but it is improved more when nodes exchange reputation, which help nodes to detect selfish nodes faster and avoid them. In the presence of 50% of selfish nodes, the routing goodput increases from 0.40 to 0.55.

Figure 6 shows the number of packets dropped by selfish nodes. With our extension, when selfish nodes are detected, we route around them, so the number of packets dropped by selfish nodes is decreased.

We also evaluated our system in the presence of nodes who forward only the necessary amount of packets so that they are not detected as selfish. This means that they try to keep their reputation in between the two thresholds which we categorize them as suspicious nodes. To refer to this type of nodes, we use the term "partially cooperative". In the simulated network, there are three types of nodes: cooperative nodes, partially cooperative nodes, and selfish nodes. Every simulation has 5 selfish nodes and the number of partially cooperative nodes varies from 0 to 20. The number of total nodes is still 50 nodes. Using the parameters in Section VI-A, we configured partially cooperative nodes to drop 75% of data packets. Thus, their reputation should be above $TH_{selfish}$ but below TH_{coop}. Figure 7 shows the result of the application goodput for all types of nodes in the simulation when using DSR with our extension compared to the one without. The application goodput of partially cooperative nodes is as expected, more than the application goodput of selfish nodes, but less than that of cooperative nodes.

VII. Conclusion

In this paper, we proposed a reputation-based system as an extension to source routing protocols for detecting and punishing selfish nodes in mobile ad-hoc networks. We also evaluated our proposed system by implementing it on JiST/SWANS simulator. The simulation results show that our system can identify both selfish nodes and suspicious nodes and punish them accordingly. Although they could save their resources by not forwarding packets for others, their packets would not be delivered as well as we can see from the result in the application goodput. The result can also provide selfish nodes with an incentive to behave more cooperatively, as behaving selfishly does not benefit them as they expect. Instead, being cooperative would result in their increased benefit.

References

[1] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proceedings of MobiCom 2000*, Boston, MA, August 2000.

[2] P. Michiardi and R. Molva, "Core: A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks," *Communication and Multimedia Security Conference*, September 2002.

[3] S. Bansal and M. Baker, "Observation-Based Cooperation Enforcement in Ad hoc Networks," *Research Report cs.NI/0307012*, Stanford University, 2003.

[4] Q. He, D. Wu, and P. Khosla, "SORI: A Secure and Objective Reputation-based Incentive Scheme for Ad-hoc Networks," *Proceedings of WCNC 2004*, Atlanta, GA, March 2004.

[5] S. Buchegger and J.-Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks," *Proceedings of MobiHoc 2002*, June 2002.

[6] S. Buchegger and J.-Y. Le Boudec, "A Robust Reputation System for Peer-to-Peer and Mobile Ad Hoc Networks," *Proceedings of P2PEcon 2004*, Cambridge, MA, June 2004.

[7] D. B. Johnson, D. A. Maltz, and Y. C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," *Internet Draft, MANET Working Group, IETF*, July 2004.

[8] Y. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks," *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, Atlanta, GA, September 2002.

[9] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, E. Belding-Royer, "A Secure Routing Protocol for Ad Hoc Networks," *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP 2002)*, Paris, France, November 2002.

[10] Y. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, Calicoon, NY, June 2002.

[11] S. Zhong, J. Chen, and Y. Yang, "Sprite: A Simple, Cheat-Proof, Credit-based System for Mobile Ad-Hoc Networks," *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM 2003)*, San Francisco, CA, April 2003.

[12] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec, "Self-Organization in Mobile Ad-Hoc Networks: the Approach of Terminodes," *IEEE Communication Magazine*, Volume 39, Issue 6, pp. 166–174, June 2001.

[13] R. Barr, "JiST - Java in Simulation Time Users Guide," http://jist.ece.cornell.edu/docs/040319-jistuser.pdf

[14] R. Barr, "SWANS - Scalable Wireless Ad hoc Network Simulator Users Guide," http://jist.ece.cornell.edu/docs/040319-swans-user.pdf