HAL
archives-ouvertes.fr

Enhancing nodes cooperation in ad hoc networks

Michaël Hauspie, Isabelle Simplot-Ryl

▶ **To cite this version:**

Michaël Hauspie, Isabelle Simplot-Ryl. Enhancing nodes cooperation in ad hoc networks. WONS, Jan 2007, Obergurgl, Austria. 2007. <inria-00117251>

HAL Id: inria-00117251
https://hal.inria.fr/inria-00117251

Submitted on 3 Jun 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Enhancing nodes cooperation in ad hoc networks[1]

Michaël Hauspie
INRIA Futurs/L.I.F.L. CNRS UMR 8022
Université de Lille I, Cité Scientifique
F-59655 Villeneuve d'Ascq Cedex, France
Email: hauspie@lifl.fr

Isabelle Simplot-Ryl
L.I.F.L. CNRS UMR 8022
Université de Lille I, Cité Scientifique
F-59655 Villeneuve d'Ascq Cedex, France
Email: ryl@lifl.fr

***Abstract*— Ad hoc networks are distributed, self-organized wireless networks. By their nature, it is easy for a malicious user to enter this kind of networks with the intention of disturbing the way they are behaving by not participating to the network. This kind of behavior is a form of selfishness where nodes want to save their energy by not routing packets. Many solutions based on virtual currency mechanisms or on reputation mechanisms have been shown to increase the networks reliability for this kind of problems. We advocate in this paper that this issue can be treated with local algorithms that have minor drawbacks compared to sophisticated solutions developed in other works. We conduct an evaluation of our solution, which shows satisfying enough results to be used in civilian spontaneous networks.**

I. Introduction

An ad hoc network is a collection of wireless mobile hosts forming a spontaneous network without the aid of any fixed infrastructure. They have potential application in civilian and military environments such as disaster relief, conference, wireless office, and battlefields. Ad hoc sensor networks for monitoring environment are also being deployed.

Most protocols developed for ad hoc networks usually consider that nodes are cooperative. Indeed, in military or emergency applications such collaboration can be assumed. However, ad hoc networks also have a great potential in civilian applications (for extending wireless Internet connectivity for example) where many nodes typically do not belong to a common authority. In such context, malicious nodes could try to cheat in their interaction with peers on network to save their own energy. For instance, cheating may happen by not forwarding routed packets. This kind of cheating has been shown to greatly impact on network performance. Indeed, in [1], Marti et al. show that 10 to 40% of misbehaving nodes cause 16 to 32% of degradation of the average throughput of the network. In this context, a significant amount of works have developed solutions to enforce node cooperation. Most of these solutions achieve significant improvement of network throughput but at considerable costs. The goal of this paper is to develop a local algorithm that improves the throughput of the network and force nodes to cooperate with minimal overhead.

The rest of the paper is organized as follows. Section II presents the context of our work and summarizes and discusses the related works. Section III presents our local algorithm that increases the throughput of cooperating nodes and decreases the throughput of non-cooperating nodes in an ad hoc network. Section IV presents the evaluation of the proposed algorithm. Section V discusses the results of our experiments and propose an improvement of the evaluation. Section VI makes concluding remarks.

[1]This work was partially supported by a grant from CPER Nord-Pas-de-Calais/FEDER TAC MOSAÏQUE and CNRS National platform RECAP

II. Context

There are several algorithms suggested in the literature to prevent node selfishness, mostly based on reputation mechanisms and virtual currency mechanisms. Usually, the underlying idea is either to reward cooperating nodes or to punish misbehaving nodes.

In this section, we present an overview of some of the well-known proposals in this area.

A. Effects of misbehaving nodes on network throughput

The first work to point out network service degradation caused by misbehaving nodes is [1]. This paper proposes a solution that combines two algorithms to counter the presence of malicious nodes. The watchdog algorithm detects locally malicious nodes and propagates the information. The Pathrater algorithm (an extension of DSR) uses this information to find routes that avoid malicious nodes. The watchdog algorithm is mainly a monitoring algorithm in which each node monitors its neighbors:

- each node has a fault counter for each of its neighbors,
- when a node forwards a packet, it starts a timer,
- if the node does not overhear the re-transmission of the packet after the timeout, the fault counter of the neighbor that was supposed to forward the packet is increased,
- when the fault counter of a neighbor is greater than a threshold, the neighbor is flagged as "malicious" and the source of the packet is notified accordingly.

As the watchdog monitors its neighbors, it needs to know the path of a packet: this fits well protocols using DSR. Then, the pathrater algorithm is used to find the "best" route based on the behavior of intermediate nodes: each node runs the pathrater algorithm and maintains a rating for every other node it knows (depending on their behavior in the past). Then, when several routes are available to the same destination, the source node is able to choose the one with the best rate.

This approach is general enough and addresses several possible malicious behaviors of nodes, however the results reported in the paper only cover malicious nodes which goal is to disturb the network. *Malicious nodes* are nodes that participate in the route discovery but drop all data packets. Results show a throughput (fraction of generated data packets that are received), which is maintained at a value around 80% even with 40% of misbehaving nodes (the simulations consider 50 nodes). The watchdog approach increases the throughput of the network up to 27% with less than 10% of additional network overhead (6% for 40% of misbehaving nodes and a random waypoint model movement scenario with 60 second pause time). The problem with this approach is that its primary goal is to increase network throughput, not to punish malicious nodes. As a result, nodes that do not cooperate are not demanded to forward packets and thus can use the network as every normal user. Instead of encouraging node to cooperate, this encourages them to be selfish and thus, more than just disturbing the network, they also save energy and are able to use network services.

Based on this initial work, several new algorithms have been developed to enforce node cooperation. Most of them are based on social or economics models, following the idea that malicious behavior is not a technical node failure but rather a consequence of user decision. Readers can refer to [2] for an overview of ad hoc networking security. In the rest of this section, we point out some works of the two main categories: economics models and reputation mechanisms.

B. Economics models

The idea of economics models is to reward cooperating nodes with virtual currency. Nodes have to pay to send a packet and forwarding nodes are rewarded since they consume resources to resend other nodes' packets.

The initial model was introduced by Levente Buttyán and Jean-Pierre Hubaux in [3]. This model uses a virtual currency called nuglets and establishes a virtual trade market around packet forwarding: nodes have to pay to send/receive packets, and nodes that forward other nodes' packets earn some nuglets.

Nodes are thus forced to cooperate in the packet forwarding process to be able to send their own packets. In the model called packet purse model (PPM), a node that wants to send a packet has to load it with nuglets to pay its transport. Then, each forwarding node uses some nuglets of the packet to pay itself before forwarding it. If a packet has not enough nuglets to reach the destination, it is discarded.

This model encourages nodes to cooperate to earn nuglets. Simulation of this model shows that it enforces nodes cooperation: Their simulations show that nodes that cooperate reduce their probability of running out of nuglets and thus increase their probability to send their own packets. Each node that is asked to forward a packet compute a payoff depending on its current battery level and nuglets level and uses it to decide if it forwards or not. Simulations show that it is possible to choose a payoff function that encourages nodes to cooperate and maintains the throughput of the network until nodes have very little battery power remaining.

The drawback of this solution is the necessity to secure the nuglets management: nodes are encouraged to cooperate because they need nuglets and thus, the model must ensure that they cannot forge or steal nuglets. To address this issue, [3] proposes the use of a tamper resistant hardware module called "security module". This module is in charge of the nuglets management: an encrypted header – the Packet Purse Header – is added to the packets between the MAC Layer header and the Network Layer header. Only security modules are allowed to manipulate this new header ensuring safe nuglets management.

Although this model is efficient in encouraging nodes to cooperate, it is limited by the need to deploy a new hardware module using cryptographic functions in all the nodes, which is costly and can be cumbersome.

In [4], a new model named Sprite, that avoids the use of a hardware tamper-proof module, is introduced. The main difference between Sprite and the nugget module is the payment control: to avoid the use of a specific module, Sprite uses a central controller, the Credit Clearance Service (CCS) which determines the credit and charge of each node of the system. When a node forwards a message, it keeps a receipt of the message and reports it to the CCS when a fast wire connection is available. Although the Sprite system allows enforcing node cooperation without any specific hardware module, it is limited in its use of the CCS. A CCS induces centralized architecture and assumes that each nodes has from time to time access to a fast connection to communicate with the CCS. Moreover, each node is supposed to have a certificate issued by a scalable authority. To avoid packet tampering, each packet is signed by the sender (using a public/private key scheme) and each forwarding node has to verify the signature, leading to extra overhead. The receipt-submission scheme is shown to be cheat-proof using game theory. An important issue that is not discussed in the paper is the fact that the sender pays *a posteriori*. It means that the sender could send a message even if it has not enough credits. As the Sprite model relies on the use of the CCS and of public key infrastructure, one can imagine that the CCS broadcasts the identity of nodes that are out of credits and of nodes that regain credits. This will lead to an extra overhead and to some latency in the information transmission, as nodes are not always connected to the CCS. As a result, some nodes may cheat for a long time before being detected while others may not be able to send packets even if they bought or earned enough credits to do so. Moreover, as one receipt is generated per hop, a large number of extra messages are generated (even if their are supposed to be sent on fast connections). Thus, once again although this proposal shows some effectiveness the cost incurred is quite heavy.

C. Reputation mechanisms

The other widely used approach in dealing with the problem of misbehaving or selfish nodes is to use *reputation mechanisms*. The general goal of reputation mechanisms is to detect malicious nodes and to propagate their "bad boys" reputation

to other nodes. Then, cooperative nodes can avoid forwarding packets for selfish nodes. The idea is to "punish" selfish nodes by preventing them to use network services. Some of these mechanisms offer the possibility to re-socialize nodes when they start to cooperate again (for example a node can refuse to forward packets when it lacks battery power and restarts its participation when it is plugged to a power supply). These kinds of mechanisms are close to the watchdog approach since they are based on monitoring.

One of the most well-known reputation mechanisms for ad hoc networks is the CONFIDANT protocol [5]. It uses first hand and second hand reputation information to detect malicious nodes and isolate them. This is the closest work to ours. Nodes deploys monitors that observe their neighbors and detect any possible misbehavior. Like in the watchdog approach, a path manager helps nodes to choose paths depending on the reputation of nodes in these paths. Moreover, two other components, the trust manager and the reputation system are added to manage reputation of nodes. The first one is in charge of the alarm messages that are used by a node to signal misbehaving nodes to its "friends", to decide the validity of such messages and to manage the list of friends. The former is used to rate nodes depending on their behavior. In fact this protocol starts with the watchdog approach and enriches it with reputation information to avoid misbehaving nodes in network functionalities. Simulations show a throughput of good nodes equivalent to the one of a a well-behaving DSR network even in the presence of one third of malicious nodes in the population. The protocol assumes that nodes can be identified (*i.e.* has a unique identity) and that it is possible to manage groups of "friends" in an ad hoc network.

The difficulty when using these kinds of solutions is the propagation of reputation [6] and the impact of false reputation. In [7], authors propose to use Bayesian statistics for exclusion of liars to improve the CONFIDANT protocol.

In the approach presented in CORE [8], it is not possible to spread negative reputation between the nodes, thus, the protocol avoids false reputation propagation classical problem. It uses a sophisticated reputation function that integrates observation of neighbors, positive reputation reported by other nodes and functional reputation (for specific tasks). An interesting point is that nodes can be reintegrated into the network if they start to cooperate again after they have been detected as misbehaving nodes.

All the approaches using reputation systems are interesting and are shown either by simulation or by formal approach like game theory to enforce node cooperation. Anyway they have a cost and suffer spoofing attacks and collusion attacks.

III. Local algorithm

The algorithm we propose in this paper uses the same basic idea as in the watchdog approach. The main weakness of this approach is that it was not designed to punish misbehaving nodes but only to avoid them. They are still able to use network services. Moreover, as a side effect of the pathrather algorithm, the malicious nodes save energy as they are no more asked to participate to the network functionnalities.

Most of the solutions that have been proposed to address this issue tend to increases the network performance but they are quite heavy. Either they require a special-purpose hardware module or a connection to a central authority in the case of virtual currency mechanisms or they require spreading some reputation over the whole network, creating more overhead. A major problem using reputation algorithms is that it is difficult to use second hand reputation information or to avoid false reputation to be propagated (see for example [7], [9]).

In [10], authors demonstrate that CORE is not efficient and argue that reputation mechanisms cannot be efficient because they require information about neighborhood that are not available in mobile environments, and that selfishness is not a relevant problem. We discuss these arguments in section V. In our opinion, the selfish nodes problem is a problem of scroungers or network availability. If it is treated as a security issue, most of the solutions are not sufficient since they are not accurate (except solutions with hardware tamper-proof modules): they are most of the time not robust enough to counter spoofing attacks or collusion attacks. On the other hand, if it is only considered as an unpleasant drawback when users need the help of other users to communicate, proposed solutions are probably too heavy. Thus, we propose in this section a lightweight local algorithm that helps nodes avoid waiting for help from non-cooperating nodes or helping non-cooperating nodes. This solution is not an exact solution but it is shown to increase the network efficiency (in the next section) with negligible overhead.

A. Local solution

Solutions that do require neither hardware adaptations nor central authority are reputation mechanisms. These mechanisms build reputation management on top of monitoring mechanisms. Thus, we propose to evaluate a local algorithm that bases itself on of neighbors monitoring information to make local decision of either forwarding a packet or not. The main idea is to extend the initial watchdog mechanism that increases the throughput of "good nodes" by adding some features that decrease the bad node's throughput.

B. Algorithm

Our proposed algorithm consists of two phases

1) The monitoring phase: is responsible for monitoring neighbors and for deciding whether they are malicious or not.

Each node monitors its immediate neighbors only. For each neighbor n, a node maintains a fault counter $\mathcal{F}_n$. Each time a node has to forward a data packet along a route using the DSR routing algorithm, it starts a timer after having forwarded the packet:

- if it overhears the neighbor, which must forward the packet before its timer expires (as we are using the DSR algorithm, we know which neighbor should forward the packet) then nothing is done except canceling the timer,

- else, if the timer expires before the packet has been forwarded by its neighbor, the counter $\mathcal{F}_n$, associated with this neighbor, is incremented. If this counter reaches a given threshold, the node blacklists the neighbor.

In the situation depicted in Figure 1, node 1 has an initial route to node 4 which uses node 2 and 3. In this scenario, node 3 is malicious and therefore, it has forwarded the route request and route reply packet but will drop any data packet it should forward. After forwarding a data packet from 1, 2 starts a timer and listen for 3 to forward it (keep in mind that as we are using DSR, 2 knows that 3 is the next node in the path). As 3 drops the packet, the timer will expire and, thus, 2 puts 3 in its blacklist.

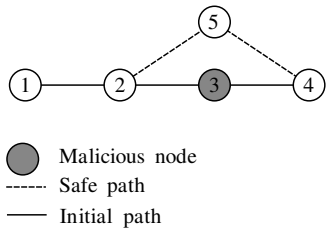

Fig. 1. Illustration of malicious nodes avoiding

2) The punishment phase: is responsible for avoiding malicious nodes and for reducing their throughput. This phase is active when a node n has to forward a packet. There are two cases here:

1) The packet is a route discovering packet (so either a route request or a route reply). In that case, the node will check all the nodes involved in this packet against its blacklist. If any of the node included in the packet (as source or part of the constructed route) has been blacklisted earlier, n drops the packet. This action will avoid constructing routes for malicious nodes and it will prevent any route using a malicious node from being constructed, and thus preventing malicious nodes from being routed. Indeed, as malicious nodes declare themselves in routes and drop data packets, we must avoid creating route using them.
2) The packet is a data packet. Here, the node checks the source of the packet against its blacklist. If the source has been blacklisted before, the node discards the packet in order to reduce the throughput of a malicious node.

In the above scenario, node 2 has blacklisted node 3. As node 3 has dropped the data packet, the DSR algorithm will trigger a route failure and thus, node 1 will try to find a new path to node 4. When DSR route replies return to node 1, the one forwarded by node 3 will be discarded by node 2 which will forward only the one forwarded by node 5. Thus, a new safe path from node 1 to node 4 has been constructed.

C. Blacklist management

The main challenge raised by our proposed algorithm is establishing a blacklist. We propose, in this paper, three versions of our algorithm to evaluate our understanding of the notion of blacklist. The three versions are one with a global blacklist, one with a pure local blacklist and one with a 1-hop blacklist.

In the global blacklist version, when a node blacklists another one, the entire network is aware of it. That is, when a node checks if another one has been blacklisted, we assume that it can know the decision of all the other nodes of the network. Thus, if a node a has blacklisted a node b, a node c which can be located anywhere in the network will know that b is a blacklisted node. This algorithm is of course not usable in a real ad hoc network context and is used here only for evaluation purpose. Indeed, the global blacklist algorithm gives the best results that our algorithm can achieve, as well as any algorithm using local monitoring and reputation propagation.

In the local blacklist version, a node knows only its own decision. It does not cooperate with others and thus, will not be aware that a node a has blacklisted a node b, even if they are its neighbors. This algorithm is the most efficient one in term of overhead but, as we will see in the evaluation section, is the less efficient because of the very small knowledge of the nodes.

In the 1-hop blacklist version, a node will know its own decision *and* the decision of its neighbors. That is, when a node a blacklist a node b, it informs its neighbors of its decision. This algorithm will have a small overhead because information is kept extremely local. Indeed, using 1-hop information has been used many times in other kind of algorithms (routing algorithms, optimized flooding algorithms...) and it has always been considered (and evaluated) as acceptable in term of overhead. As we will see in the evaluation section, the results of this algorithm are very good, even if nodes are moving.

Another issue has been left out in the description of the algorithm. If the above algorithm is run *as is*, the nodes will blacklist each other's very fast. Indeed, if a node drops a packet because it has decided that this packet was involving a blacklisted node, it will certainly be blacklisted as well by the preceding node. To avoid this, we introduce an announce packet. A node, which wants to drop a packet, must send another packet of the same size. By sending a packet with the same size, the node shows that it has not dropped the packet to save power, but it fills the packet with noise to punish the malicious node. When a node receives this kind of packet, claiming that it has just been blacklisted by one of its neighbor, it blacklists the corresponding neighbor as well. Using this simple rule, if a node claims falsely that others are malicious, it will push itself out of the network.

IV. Evaluation

A. General settings

In the experiments, we consider a network using standard DSR algorithm without defense as reference, and the three versions of our algorithm (global, local and 1-hop) presented in the previous section. The networks are generated randomly

and we introduce malicious nodes selected randomly. Like in most of the related works [1], [3], [7], a malicious node is a node that participates to the route discovery but always drops data packets (Nodes that do not participate to the route discovery decrease the network density but do not increase the number of data packets dropped). In the simulations, we vary the percentage of malicious nodes from 0% to 40%, in order to compare our results to [1] (even if 40% of malicious nodes may seem unrealistic). For each percentage of malicious nodes, we run 5 experiments on different network configurations. We use the same seed to generate the networks for each percentage to avoid side effects in the comparisons of results at the different percentages.

The density of networks is set to 16 (*i.e.* the nodes have an average of 15 neighbors), thus, network have a high probability to be connected to avoid side effects of isolated nodes and to sometimes provide alternate paths around malicious nodes.

The simulation consists of several rounds. At each round, 10 pairs of connected nodes are selected in the network and one node in a pair tries to send 40 data packets of 512 bytes to the other. We measure the throughput of the non-malicious sending nodes (referred to as *goodput*) and of the malicious sending nodes (referred to as *badput*). The throughput of a node is the ratio of packets that have reached the destination over how many were sent. Notice that the detection of malicious nodes may introduce route defaults. As the initial density is set in order to have a connected network, we consider any route default as a failure: for each pair of selected nodes, the sender has 40 packets to send. If it is not able to find a route to the destination, the 40 packets are considered as lost. If a route failure occurs after having send a fraction of the 40 packets, another route is looked for, and if no new route is found, the remaining packets are considered to be lost. Note that this is a major difference with the pathrater algorithm of [1], which tries to use the best possible route even if it contains some malicious nodes.

The communication range is set to 100 meters and the size of the area is set according to the desired density (*i.e.* average number of neighbors, 15 here). The maximal transmission rate is 11 Mbits/s and each node is supposed to process the packets during a constant time of 5ms (since the packet size is constant). We run our algorithm for two different types of network: static, and with mobility. The mobility model used is the random waypoint model. In this model, each node selects a random point and then moves towards it at a random constant speed. When it reaches it, the node selects a random stop time and after this stop time is over, it restart the process of selecting a waypoint and so on. We chose two mobility senarii:

- mean mobility: 5m/s and pause time randomly chosen between 0s and 20s,
- high mobility: 20m/s and no pause time.

The second one is probably not realistic for spontaneous networks created by people carrying portable objects, but it may correspond to car speed and it is used to stress the protocol and evaluate its limits.

In this set of experiments, we consider a perfect underlying network and nodes that are either always misbehaving either always cooperative thus, after a certain time, the malicious nodes are all detected and the network does not evolve anymore. For this reason, we only consider the initial phase. In the next section, we present experiments in which nodes can change their behavior.

B. Experiments on a 50 nodes static networks

We first use a 50 nodes which is the most common size used in the evaluations of others models ([1], [5] for example). The parameters are fixed as shown in Table I.

Parameter	Level
Number of nodes	50
Density	16
Rounds	200
Range	100m
Area	313m $\times$ 313m
Pairs of nodes	10
Number of packets per connection	40
Packet size	512 bytes
Transmission rate	11Mbits/s
Process time per node	5ms

TABLE I
PARAMETERS FOR 50 NODES EXPERIMENTS.

In Figures 2 and 3, we observe that the throughput of the nodes without using any defense algorithm is similar to the one given by the other papers of the field ([1] for example). This shows that the context of our experiments can be compared to theirs.

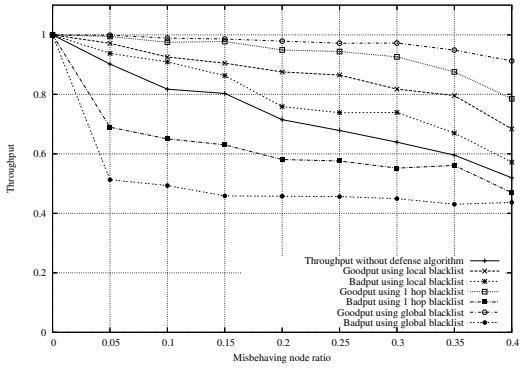

Fig. 2. 50 nodes, static network.

We first consider a static network. As stated in section III, the results given by the algorithm using a global blacklist are the best possible results we can have using our method: as the blacklist is considered global and the propagation is considered instantaneous, we will never be able to have more accurate results using a totally local decision. This algorithm was of course only evaluated to serve as reference.

As we can see Figure 2, without defense algorithm, the throughput of the nodes of the network falls from about 100% to about 52% when 40% of the nodes are malicious.

This shows the impact of malicious nodes over the network performance. If we look at the global blacklist algorithm, we can see that it achieves to keep the goodput very high, even when a lot of nodes are malicious. This tends to show that even a local decision can give very good results if everyone is aware of this decision, which is the basic goal of reputation systems. We must remark that these satisfying results are possible because of the connectivity of the network and the probability of the existence of several routes between two nodes, the throughput falls with 40% of selfish nodes mostly because of the lost of connectivity.

On the opposite, the local algorithm impact is not satisfying: the average difference of throughput between malicious and cooperative nodes is 14%, it cannot be considered as an incentive for node cooperation.

The best algorithm in terms of results and overhead is the 1-hop algorithm. With 40% of selfish nodes, the throughput of cooperative nodes falls to 79%, which is 13% less than the referent global solution but still 27% more than the defenseless solution. On the other side, the throughput of selfish nodes falls bellow 60% which encourages nodes to cooperate. One can notice that this badput is reached with only 20% of selfish nodes whereas the throughput of cooperative nodes is 95%. Thus, with 10 to 35% of selfish nodes, the average difference between the throughput of cooperative and selfish nodes is around 35%, and the throughput of cooperative nodes is maintained over 85%. It appears from this experiments that we reach a compromise with the 1-hop solution in terms of efficiency and overhead.

C. Experiments on a 50 nodes mobile networks

Figure 3 shows the same experiments on networks where nodes are mobile with both senarii: mean mobility and high mobility. The throughput of the defenseless network is better in the mobility case than in the static case. This can be explained by the fact that nodes neighborhood is changing and thus no node is enclosed in a malicious neighborhood for the experiment duration. Anyway these experiments confirm the static results in the following sense: in both cases, the average difference between goodput and badput is around 47% which can be considered as a good motivation for nodes to cooperate.

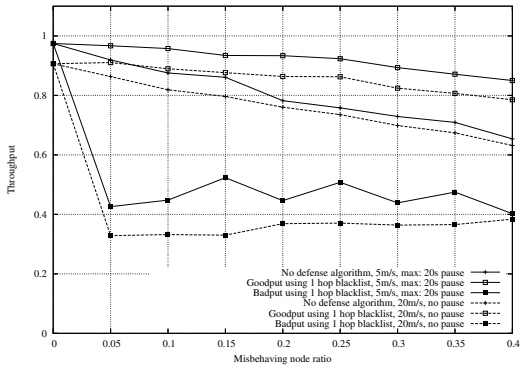

Fig. 3. 50 nodes with mobility.

As conclusion of this subsection, we give a summary of the benefits of the protocol in Figure 4. Depending on the mobility senario, the gain in terms of goodput and the loss in terms of badput vary, but the difference between gain of goodput and loss of badput in a given senario is mostly equivalent, which is the stability necessary to show the efficiency of our protocol to enforce node cooperation.

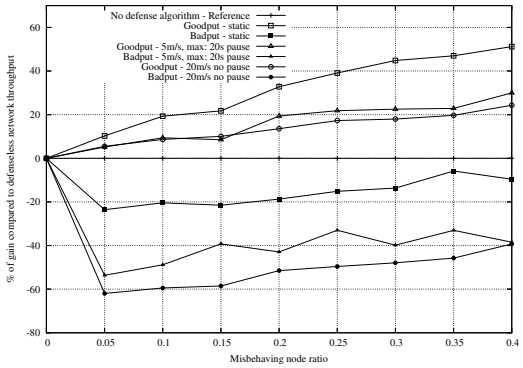

Fig. 4. Percentage of throughput gain of 1-hop algorithm compared to defenseless solution (50 nodes).

V. Discussion and experiment improvement

A. Discussion

We could naively conclude from the previous section that the 1-hop algorithm is very efficient and is sufficient to obtain a satisfying difference between the throughput of cooperative and non cooperative nodes to encourage node to cooperate. Nevertheless, when we consider 50 nodes uniformly distributed on a 313×313m square (to have a density of 16), the mean route size in experiments is 2.69 hops (note that the theoretical mean distance between two points randomly chosen is approximated by 162,76m [11], but as we use distances in hops, the average route size is a little bit longer). Using the 1-hop algorithm, after a short initial time, each node has the complete information for the route of size lower or equal to 2, so the impact of the use of a local protocol is not completely demonstrated. We have chosen a 50 nodes network because it is the network size used in most of the references we mentioned, and it corresponds to a medium size meeting of people, in a classroom or a workshop for example. To improve our results and discuss the robustness of the protocol, we now have to conduct experiments on larger networks.

B. Larger experiments

We now conduct the simulations in the same context as previously except that we now simulate a 250 nodes network (of density 16 like previously). Parameters are depicted in Table II.

We first conduct experiments on a static network. The results are depicted in Figure 5 and must be compared to Figure 2. The first thing that we can notice is the reference throughput of the defenseless network. Marti et al. announced

Parameter	Level
Number of nodes	250
Density	16
Rounds	200
Range	100m
Area	700m $\times$ 700m
Pairs of nodes	10
Number of packets per connection	40
Packet size	512 bytes
Transmission rate	11Mbits/s
Process time per node	5ms

TABLE II

PARAMETERS FOR 250 NODES EXPERIMENTS.

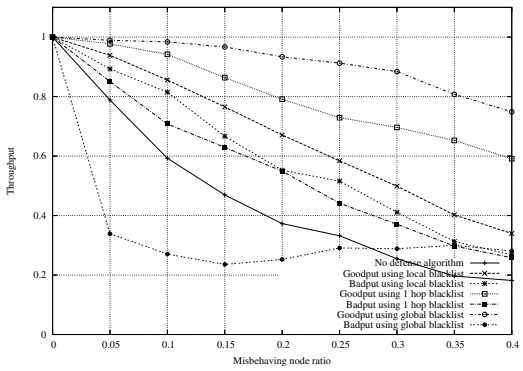

Fig. 5. 250 nodes, static network.

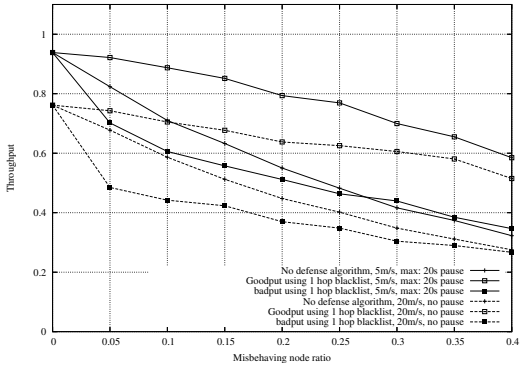

Fig. 6. 250 nodes with mobility.

in [1] that if 10% to 40% of the nodes misbehave, then the average throughput degrades by 16%-32%. They used a 50 nodes network and we obtain similar results. We can see on Figure 5 that the throughput degrades by 33%-83%. As we use a 250 nodes network, the mean route size is now 4,5 hops. Table III shows a comparison of routes in the 50 nodes network and in the 250 nodes network: the probability to find a selfish node in a route of the mean size (denoted by PSNR) is very high in general and higher in the 250 nodes case. Most important, the probability to use a 1-hop route is 30% in the 50 nodes network and only 6% in the 250 nodes network. This factor is very important since malicious nodes have absolutely no impact of the traffic on 1-hop routes. This means that 30% of the traffic can be considered as safe and distorts the results.

	50 nodes	250 nodes
Mean route size	2.69	4.5
PSNR with 20% of misbehaving nodes	0.45	0.63
PSNR with 30% of misbehaving nodes	0.61	0.8
PSNR with 40% of misbehaving nodes	0.75	0.9
1-hop routes	30%	6%

TABLE III

ROUTES COMPARISON FOR 50 AND 250 NODES NETWORKS.

Nevertheless, the experiments show once again the necessity to use the 1-hop algorithm, which maintains the goodput over 70% until 30% of malicious nodes. When more malicious nodes are introduced in the network, the goodput severely degrades but the difference between goodput and badput reaches 33% with 40% of malicious nodes. Moreover, the use of the 1-hop algorithm increases the goodput by 39% in average compared to the reference throughput of a defenseless network. Thus, we can conclude that nodes are encouraged to cooperate by the 1-hop algorithm even in worse conditions.

The impact of the mobility shown in Figure 6, consists on a general deterioration of the goodput. Anyway, the 1-hop algorithm maintains an average difference of about 27% between goopdput and badput.

We conclude this subsection by Figure 7, which summarizes the gain of throughput we obtain compared to the reference defenseless network.

C. Relevance of our model

We discuss in this subsection the relevance of our model. We have shown on experiments that it creates a significant difference between cooperative nodes throughput and malicious nodes throughput that may enforce nodes to cooperate. We could strengthen the algorithm effect by for example dropping the packets addressed to malicious nodes. This improvement of the protocol could be easily implemented but would have complicated the analysis of the results as even non-malicious nodes would have seen their throughput decreased if they wanted to send data to a misbehaving node.

The main issue with our algorithm is that it is based, such as reputation mechanisms, on neighborhood monitoring. In [10], authors argue that the monitors (like WatchDog) require information about the neighborhood that they cannot know (except in non-mobile environments), and that the rate of false negative can easily become significant enough to obliterate the network throughput. Moreover, they argue that reputation systems only work under the assumption that nodes identity cannot be falsified. Our algorithm suffers the same limits: it is not resistant to spoofing attacks. We only argue that we can obtain interesting results with less overhead and less drawbacks that traditional reputation systems that are conceptually sound but quite heavy compared to a simple local solution. We can also detect false negative but, as we do not

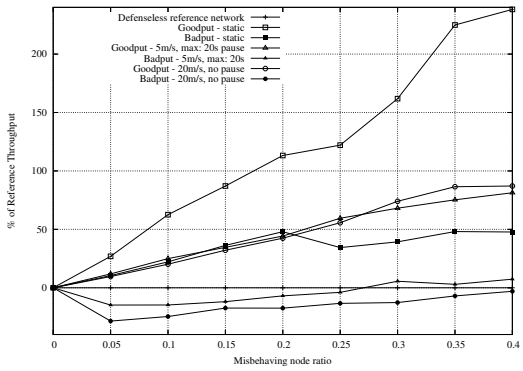

Fig. 7. Percentage of reference throughput 250 nodes.

propagate reputation, the error is local to a node.

To illustrate how our protocol would react when the behavior of the nodes can evolve, we conducted another experiment. We run the above scenario in a situation where nodes can be potentially malicious. A potentially misbehaving node is a node that cycle between the malicious and non-malicious states over time. This kind of scenario gives results on the behavior of our proposed algorithm if we want nodes to be able to be kind again, for instance if they notice that their malicious behavior has been spotted.

Although the parameters and the scenario of the experiments remains the same as those depicted in table II, two main changes are done this new experiment. First, the malicious nodes behave periodically as malicious and non-malicious. They stay in malicious mode for 12 seconds and switch to non-malicious for 48 seconds. A node's full cycle is thus 60 seconds. The initial state and time in this cycle is selected randomly so that not all the nodes are in the same state at the same time. Second, to handle the changing behavior of the nodes, a node that do not misbehave for 5 seconds is removed from the blacklist.

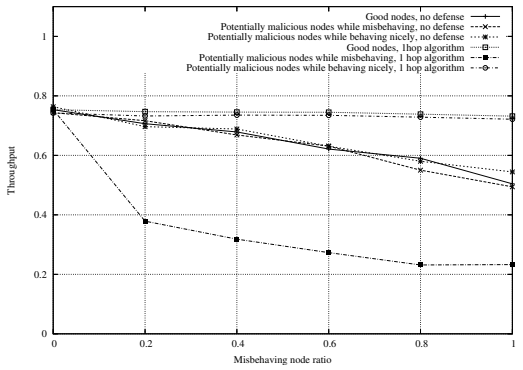

Fig. 8. 250 nodes, 20m/s, 0 pause, social.

The results of this experiment is given Figure 8. In this experiment, we measure the throughput of good nodes and potentially malicious nodes while in their two possible state. As we can see, the throughput of the malicious nodes when they are behaving nicely is almost as good as the good nodes' one while their throughput when they are misbehaving has been reduced by the algorithm. This shows that our algorithm is able to allow a node to change its mind if it sees that its malicious behavior has been spotted.

VI. Conclusion

In this paper, we propose a local algorithm that prevent malicious user to behave in a selfish manner by pretending that the node will participate and not doing so. We advocate that the use of a local algorithm gives sufficient results to solve this issue in civilian networks: it increases the network throughput with minimum drawbacks. The algorithm has some weaknesses, for example, it cannot detect nor punish malicious nodes that change identity to flood the network. Anyway, the other protocols suffer the same problem: they must rely on a public key infrastructure or other mechanisms. In future work, we plan to continue the evaluation of the protocol on larger networks and to introduce some re-socialization mechanisms to allow nodes that have been blacklisted be the neighbors to start again to cooperate.

Acknowledgment

The authors would like to thank Issa Traoré for his careful review and his pertinent remarks.

References

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

[2] P. Michiardi and R. Molva, *Mobile Ad Hoc Networking*. Wiley-IEEE Press, 2004, ch. Ad Hoc Network Security, pp. 329–354.

[3] L. Buttyán and J.-P. Hubaux, "Enforcing service availability in mobile ad-hoc WANs," in *Proc. of the 1st ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2000)*. Boston, Massachusetts: ACM, 2000, pp. 87–96.

[4] S. Zhong, J. Chen, and Y. R. Yang, "Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks," in *Proc. of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*. San Francisco, CA, USA: IEEE, 2003.

[5] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc neTworks," in *Proc of MobiHoc*. ACM, 2002, pp. 226–236.

[6] Y. Liu and Y. R. Yang, "Reputation propagation and agreement in mobile ad-hoc networks," in *Proc. Wireless Communications and Networking (WCNC 2003)*, vol. 3. IEEE, 2003, pp. 1510–1515.

[7] S. Buchegger and J.-Y. L. Boudec, "The effect of rumor spreading in reputation systems for mobile ad-hoc networks," in *Proc. of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'03)*, 2003.

[8] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proc. of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V., 2002, pp. 107–121.

[9] J. Liu and V. Issarny, "Enhanced reputation mechanism for mobile ad hoc networks," in *Proc. of Trust Management, Second International Conference (iTrust 2004)*, ser. LNCS, C. D. Jensen, S. Poslad, and T. Dimitrakos, Eds., vol. 2995. SPRINGER, 2004, pp. 48–62.

[10] R. Carruthers and I. Nikolaidis, "Certain limitations of reputation based schemes in mobile environments," in *Proc. of the 8th ACM International Symposium on Modeling, analysis and Simulation of Wireless and Mobile systems (MSWiM '05)*. Montréal, Canada: ACM Press, 2005, pp. 2–11.

[11] "http://mathworld.wolfram.com/squarelinepicking.html."