On the Use of Cooperation as an Energy-Saving Incentive in Ad Hoc Wireless Networks

Maurizio D'Arienzo[1], Sabato Manfredi[2], Francesco Oliviero[2],
and Simon Pietro Romano[2]

[1] Dipartimento di Studi Europei e Mediterranei, Seconda Università di Napoli, Italy
maudarie@unina.it
[2] University of Napoli Federico II, Napoli, Italy
{sabato.manfredi,folivier,spromano}@unina.it

Abstract. In this paper we show how cooperation can improve the overall energy efficiency of an ad hoc network. By exploiting a behavior-tracking algorithm inspired by the results of game theory and allowing traffic to be forwarded only towards cooperative nodes, we show how we can dramatically reduce power wastage at the same time maximizing goodput. Under this perspective, cooperation can definitely be seen as an incentive for all nodes, since it allows to optimize one of the most crucial parameters impacting the performance of ad hoc networks.

1 Introduction

Ad hoc networks are composed of several nodes with wireless connection capability. Differently from wired networks, in an ad hoc environment each node is an end system and a router at the same time. A transmission between a sender and a receiver happens with the help of one or more intermediate nodes that are requested to relay packets according to routing protocols designed for this kind of networks. A blind trust agreement among nodes makes it possible the right message forwarding. Actually, a generic node of the network should be able to decide whether or not to trust the other nodes. This obviously calls for a capability of each single node to somehow interpret (or, even better, predict) the behavior of the other nodes, since they represent fundamental *allies* in the data transmission process. The situations in which a decision of a part depends on the predicted behavior of another part have been elegantly studied in game theory. Game theory has been already applied [3] [4] [5] to ad hoc networks with interesting results. The basic assumption is that all the players follow a rational behavior and try to maximize their payoff. The simplest games see the involvement of only two players who have to decide whether to cooperate or defect with the others. The best solution may not maximize the payoff, but can reach an *equilibrium* as proposed by Nash. One of the versions of this game is known as *prisoner's dilemma* and has an equilibrium in case both users decide to defect. This is true for the game played only one time while in its iterated version the situation is more complex and even cooperation can be convenient. In case of

Joel J.P.C. Rodrigues et al.: (Eds.): GreeNets 2011, LNICST 51, pp. 148–160, 2012.

ad an hoc network, the player is a node that needs to cooperate with the others to send its traffic. However some nodes can decide to defect for a number of unspecified reasons and, as a first need, the other nodes should be informed of their behavior in order to react in the most appropriate way.

In this paper, we show how cooperation can be perceived by nodes as an incentive, thanks to the fact that it helps save the overall amount of energy needed for data transmissions. Differently from recent works proposed in the literature, which aim at making the routing process become *natively* aware of the energy-related parameters, we herein propose a different approach, by leveraging cooperation in order to improve the overall energy efficiency of an ad hoc network without modifying the existing routing protocol. We present in the paper an algorithm to identify and isolate defecting nodes. The algorithm takes inspiration from the results of game theory and keeps a local trace of the behavior of the other nodes. In case defecting nodes are identified, different countermeasures (i.e. not relaying packets coming from defecting nodes) can be adopted.

The paper is organized in six Sections. Section 2 deals with both background information and related work. Section 3 presents the algorithm we designed to infer behavioral information about the network nodes, whose implementation is described in Section 4. Results of the experimental simulations we carried out are presented in Section 5, while Section 6 provides concluding remarks and proposes some directions of future work.

2 Background and Related Work

In this section we try to shed light on the context of our contribution, by properly defining the scope of our research. We focus on the most important aspect of our contribution, namely cooperation. Indeed, as we already pointed out, cooperation is a fundamental subject of our recent research and is herein studied under one of its most challenging facets, i.e. its use as an incentive for all the nodes of the network, thanks to the significant performance improvements that it entails in terms of energy savings associated with data transmissions.

Cooperation of nodes involved in an ad hoc network is usually induced because the efforts related to the offered services are compensated with the possibility to request a service from the other nodes. However current ad hoc network protocols do not provide users with guarantees about the correct behavior of other nodes that can potentially decide to act as parasites. Several works have identified the problem of stimulating cooperation and motivating nodes towards a common benefit. The main solutions rely on a virtual currency or on a reputation system, and more recently on game theory.

Virtual currency systems [6] [7] give well behaving users a reward every time they regularly relay a packet. They can then reuse the reward for their transmissions as long as they have a credit. The first issue of such systems is related to the need of a centralized server to store all the transactions among the users. Reputation systems repeatedly monitor and build a map of trustworthy nodes on the basis of their behavior [1] [2] [5] [8]. These systems distinguish between the *reputation*, which rates how well a node behaved, and *trust*, which represents how honest a node is. Most of these systems consider the reputation value as a metric of trust . A node is refrained from relaying a packet coming from untrusted nodes, which are then excluded from the network operations. Several issues are related to the use of these systems. First, each node needs to maintain a global view of the reputation values, with considerable caching. Some proposals keep local information, others disseminate reputations to other nodes, with an increased overhead due to the exchange of such messages. Reputation values can be modified, forged or lost during operations, and they can differ from node to node, which can bring to inconsistencies.

To overcome some of these issues, it has been proposed to model the nodes taking part to an ad hoc network with game theory, a branch of applied mathematics which witnessed a great success thanks to the application of its results to a wide selection of fields, including social sciences, biology, engineering and economics. Game theory covers different situations of conflicts regarding, in a first attempt, two agents (or *players*), and in the generalized version, a population of players. Each of these players expects to receive a reward, usually named *payoff*, at the end of the game. The basic assumption is that all the players are self interested and rational: given a utility function with the complete vector of payoffs associated with all possible combinations, a rational player is always able to place these values in order of preference even in case they are not numerically comparable (e.g. an amount of money and an air ticket). This not necessarily means that the best value will be selected, since the final reward of each player is strongly dependent on the decision of the other players. Each player is then pushed to plan a *strategy*, that is a set of actions aiming at a total *payoff* maximization, provided that he is aware that the other players will try to do the same. Games are now classified according to various properties. Here we are mainly interested in the difference between *cooperative* and *non-cooperative* games as well as the difference between *strategic games* (played once) and *extensive games* (played many times).

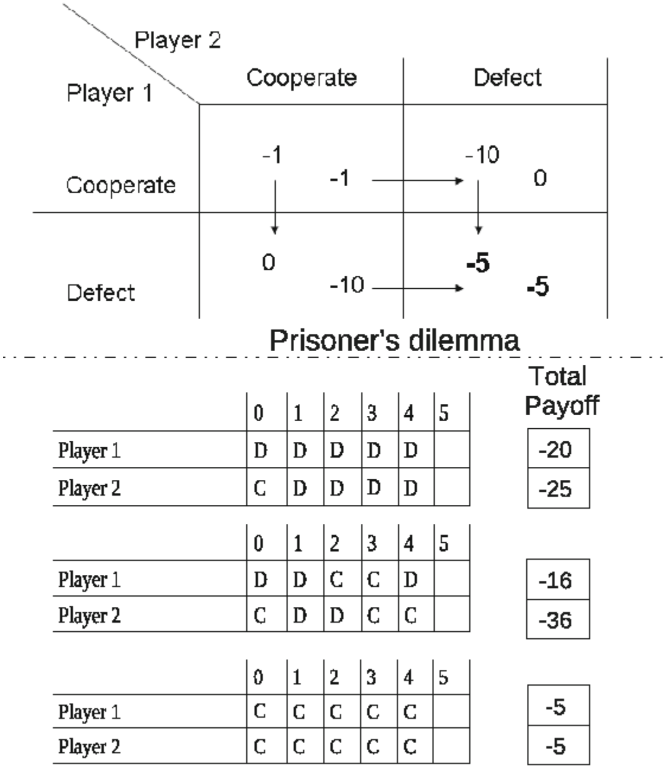
One of the fundamental problems of game theory is known as *prisoner's dilemma*, which can be represented in the matrix format of Fig. 1: two suspects of a crime are arrested and jailed in different cells with no chance to communicate between each other. They are questioned by the police and receive the same deal: if one confesses (*defect*) and the other stays silent (*cooperate*), the first is released, the second is convicted and goes to prison with a sentence of 10 years, the worst; if both stay silent (*cooperate*), they go to prison for only 1 year; if both testify against the other (*defect*) they go to prison with a sentence of 5 years. The situation in which they both stay silent (*cooperate*) is the more convenient to both of them; however, it was demonstrated that a rational behavior is to confess (*defect*) and receive the sentence of 5 years, and this situation represents the only equilibrium, as first introduced by Nash [14] [13]. Hence, the prisoner's dilemma falls in the field of strategic non-cooperative games. In its basic form the prisoner's dilemma is played only once and has been applied to many real life situations of conflict, even comprising thorny issues of state diplomacy.

Another version of the prisoner's dilemma is played repeatedly rather than just once and is known as iterated prisoner's dilemma (ITD), which turned out to be a cooperative game under certain circumstances [9][10]. The goal of both players still is the maximization of their payoff, as the cumulated payoff earned at each stage. If the number of rounds is finite and known in advance, the strategy of always defecting is still the only situation of equilibrium and the game is still non-cooperative. However, in case the number of repetitions is infinite, it was demonstrated that the choice to always defect is not the only equilibrium as even the choice of cooperating may be an equilibrium. In this case, one of the strategies that let players maximize their payoff is the so-called *Tit for Tat* game, in which each player repeats the past behavior of the other player: a player is keen to cooperate if the other node behaved correctly the last time, otherwise it defects. If we consider the first five tournaments of a two players game, a player who defects (D) against a cooperative (C) player adopting the tit for tat strategy would play (D,D,D,D,D) and earn $(0, -5, -5, -5, -5) = -20$. If the first player decides to cooperate two times out of five (D,D,C,C,D), he would earn $(0, -5, -10, -1, 0) = -16$. In case he always cooperates, his payoff would be $(-1, -1, -1, -1, -1) = -5$, which is the best he can achieve. So, continued cooperation for the iterated prisoner's dilemma also yields the best payoff. Despite this benefit, the main result of the tit for tat strategy is that it stimulates the cooperation. We base our algorithm to mitigate the node selfishness on the results of this version of the game.

Game theory has already been applied to the study of ad hoc networks. One of the first proofs of the improvements produced by cooperation in such networks is presented in [3]. The authors first introduce a normalized acceptance rate (NAR) as the ratio between the successful relays provided to the others and the relay requests made by the node. Then they propose two models, namely GTFT (Generous Tit for Tat) and m-GTFT for the case of multiple players, to give the (rational) nodes the chance to make a decision concerning the possibility to cooperate or defect with other nodes, and they analytically demonstrate that these models represent a Nash equilibrium. In such a situation, a node does not improve its NAR to the detriment of the others. Also, at the opposite of reputation schemes, each node can maintain per session rather than per packet information, thus leading to a scalable solution.

In [15] the authors prove the selfishness property of the nodes in a MANET by using the Nash equilibrium theorem [13]. They define a generic model for node behavior which takes into account also energy consumption due to the transmission process. By adopting a punishment based technique they prove that it is possible to escape from the theoretically unique equilibrium point of non-cooperation and to enforce a cooperation strategy under specific conditions.

In [16] the authors also focus on forwarding mechanisms. They provide a model for node behavior based on game theory in order to determine under which conditions cooperation with no incentives exists. They prove that network topology and communication patterns might significantly help enforce cooperation among nodes.



	0	1	2	3	4	5	Total Payoff
Player 1	D	D	D	D	D		-20
Player 2	C	D	D	D	D		-25

	0	1	2	3	4	5	
Player 1	D	D	C	C	D		-16
Player 2	C	D	D	C	C		-36

	0	1	2	3	4	5	
Player 1	C	C	C	C	C		-5
Player 2	C	C	C	C	C		-5

Fig. 1. The tit-for-tat strategy in action

Game theory has been also used to improve routing algorithms in wireless networks. An actual implementation of a game theory model in the AODV routing protocol with two distinct approaches has been proposed in [4]. The first plays a deterministic tit for tat game and the second a randomized version of the same game deployed with a genetic algorithm. In both cases, they achieve better performance in terms of experienced delay and packet delivery ratio in case of cooperation of nodes. The models are tested in a simulated environment and rely on static distribution of nodes' behavior profiles while not supporting a mechanism for a dynamic adaptation to changed situations.

3 A Novel Behavior-Tracking Approach

In an ad hoc network, the number of nodes and links can change over time, so we consider the number of nodes $N(t)$ as a function of time t. We also define a dynamic array $C(t)$ of $N(t)$ elements for each node of the network. The generic element $c_i(t)$ of $C(t)$ assumes the values (UNKNOWN, COOPERATE, DEFECT) meaning that the behavior of node i at time t is respectively unknown, cooperative or non cooperative. At time $t = 0$ all the values are set to UNKNOWN, since at the beginning each node is not aware of the behavior of the other nodes. We herein propose to introduce a routing control strategy at each node based on the game theory framework described earlier. Basically, the control algorithm firstly identifies the cooperative nodes (Detection Phase) and then reacts in the most appropriate way in order to give priority to packets generated by cooperative nodes. Specifically the control algorithm is composed of a *Detection Phase*, followed by a *Reaction Phase*. The former works as follows. Suppose the generic node s of the network needs to send some traffic to the destination d. The first task is to discover an available path, if it exists, to reach the destination. To this purpose, we consider a source based routing protocol capable of discovering a list $A(t)_{(s,d)i}$, $\forall i : 0 < i < P$, of P multiple paths. All the nodes in the list $A(t)_{(s,d)i}$ are considered under observation and marked as probably defecting in the array $C(t)$ unless a positive feedback is received before a timeout expires. The sender s starts sending his traffic along all the discovered paths. If the destination node generates D acknowledgement messages containing the list of all the nodes $L_{(s,d)i}$ (with $0 < i < D$) traversed, as it happens in some source based routing protocols, the sender s is informed about the behavior of intermediate nodes. For each acknowledgement message received, the sender s can make a final update of the array $C(t)$ by setting the matching elements $c_i(t)$ contained in the list $L_{(s,d)i}$ as cooperative. Notice that the last update overwrites the previous stored values and represents the most recent information concerning the behavior of a node.

Once done with the detection phase, each node is aware of the behavior of other nodes and can enter the reaction phase in the most appropriate way. For example, a node can refuse to relay packets of defecting nodes, or operate a selective operation like queuing their packets and serving them only if idle and not busy with the service requested by cooperative nodes. In this first proposal, we rely on the harsh policy of packet discarding, and this brings to the isolation of defecting nodes. However, a defecting node can even gain trust of other nodes if it starts to cooperate. The array $C(t)$ is not static over time and its values are continuously updated. In fact, due to the dynamic situation of ad hoc networks, the search of available paths is frequently repeated, and the list $A_{(s,d)}$ consequently updated. Hence, if a defecting node decides to cooperate, its identification address will be included in one of the acknowledgement messages $L_{(s,d)i}$ sent to the sender s and its aim to cooperate will be stored in the array $C(t)$.

The situation described here for the pair (s, d) is replicated for all possible pairs of nodes that try to interact, but each node stores only one array $C(t)$ that is updated upon reception of any acknowledgement message, wherever it comes from. Furthermore, not all the packets relayed are checked in order to verify the nodes' behaviors, but only a sample of them, thus keeping the total overhead under control.

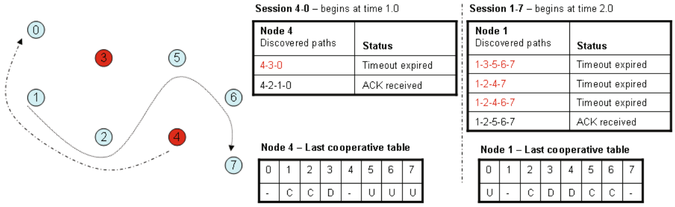
4 Algorithm Implementation

The algorithm introduced in the previous section has been implemented in AH-CPN (Ad Hoc Cognitive Packet Network) [11], an existing source based routing protocol for ad hoc networks. AH-CPN is the wireless version of CPN (Cognitive Packet Network) [12], a proposal for a self aware network architecture with native support for QoS.

There are four different kinds of packets in AH-CPN: Smart Packets (SP), Smart Acknowledgements (SA), Dumb Packets (DP), and Dumb Acknowledgements (DA). SPs are lightweight packets sent by a sender towards a destination to discover new paths according to specific QoS goals, e.g. discovering paths that minimize the delay or maximize the throughput. Once at the destination, a SA is generated and sent backwards along the reverse path received in the SP. Finally, the actual data can be sent across the network in a DP, which is prepared with the whole path copied in the DP header. Once the DP reaches its destination, a DA is sent along the reverse path. Notice that differently from IP networks, in CPN the acknowledgements are generated upon reception of each single packet, whatever the transport protocol is. This feature is helpful in the deployment of our algorithm to identify defecting nodes, as we will soon explain. We first modified this protocol to support the search of multiple paths, and then included the new algorithm for the identification of non cooperative nodes.

The basic AH-CPN version looks for one available path, the best in terms of the requested QoS goal. We modified this protocol to search for multiple paths. To this purpose, SPs are initially sent via flooding to collect all the available paths. To prevent loops, SPs are marked with an identification number ID, and those with the same ID touching a node for the second time are discarded. SPs reaching the same destinations with different contents for what concerns the routing map are considered valid, and SAs are sent backward to inform the sender. The sender collects the different SAs and updates its routing table. DPs are sent on a round robin basis. Once the available paths are discovered, the transmission of SPs is not terminated; it is rather repeated periodically for path maintenance, to check if the topology has changed, and in our case also to verify if there is a different configuration concerning the behavior of nodes.

We then provided the multipath source based routing protocol with the support for identification and isolation of defecting nodes. The array $C(t)$ is added and stored in each node and its dimension can change according to the number of nodes active in the ad hoc area. When node a needs to send traffic to node b, SPs are immediately sent in flooding. We make the assumption that non cooperative nodes try to cheat by forwarding inexpensive SPs, that do not carry



Session 4-0 – begins at time 1.0

Node 4 Discovered paths	**Status**
4-3-0	Timeout expired
4-2-1-0	ACK received

Node 4 – Last cooperative table

0	1	2	3	4	5	6	7
-	C	C	D	-	U	U	U

Session 1-7 – begins at time 2.0

Node 1 Discovered paths	**Status**
1-3-5-6-7	Timeout expired
1-2-4-7	Timeout expired
1-2-4-6-7	Timeout expired
1-2-5-6-7	ACK received

Node 1 – Last cooperative table

0	1	2	3	4	5	6	7
U	-	C	D	D	C	C	-

Fig. 2. The simulated testbed

any payload, while they do not relay DPs containing the real data. In case the non cooperative nodes decide to block the SPs forwarding, they are immediately discovered as non cooperative and have no chance to cheat. In this scenario, every time a SP traverses a node, its cognitive map is extended with the label of the visited node. Once at the destination, the complete cognitive map is copied into the DA and sent back to the sender along the reverse path. Obviously, this is repeated for all the discovered paths, so at the end of the process node a has a complete knowledge of all the available paths, also those comprising cheating nodes, and these are all stored in $A(t)_{(a,b)}$. At the time of the first transmission, the real data are packed in multiple DPs and sent along all the available paths on a round robin basis, but the interested cheating nodes will not rely ay them. Since in CPN a destination b must send an acknowledgement message DA whatever the transport protocol is, node a will receive only the DAs containing the successful paths, i.e. those without cheating nodes. This information, as described before, helps finalize the array $C(t)$ with the list of cooperative and defecting nodes, and the traffic is sent only along the path or the paths composed of cooperative nodes rather than towards all the available paths. When one of the cheating nodes requests the relaying of a message to node a, it is aware of his past behavior and can decide to drop all its packets, while it can regularly relay packets coming from cooperative users.

The situation concerning the cooperation and the selection of paths is not static and can change over time, so isolated nodes are not banned forever from the network. Although the traffic from a node is delivered only along paths composed of cooperative nodes, sending nodes continue to check periodically the paths containing the defecting nodes. Should a defecting node decide to change its behavior and begin to cooperate, the routing protocol soon detects this change and admits again the node to the transmission of flows. This way, a node reacts following a *Tit for Tat* strategy.

5 Experimental Results

The introduction of a system able to detect defecting nodes composing a wireless ad hoc network can lead to a better distribution of the energy consumed by each node. To show this effect, we tested the proposed system with the ns-2 simulator. The current implementation of the algorithm relies on a dedicated multiple path ad hoc routing protocol that supports the explicit acknowledgement of packets regularly received at the final destinations. To highlight the robustness of the algorithm, we designed a scenario associated with several working conditions, on a simple wireless testbed composed of 8 nodes (see Fig.2), labeled from 0 to 7. In such network we set up the following conditions: (i) node 3 defects all the time; (ii) the behavior of node 4 dynamically changes over time; (iii) all the other nodes are cooperative. The duration of the experiments is set to 12 minutes. The defection of a node means that the relay of traffic to serve other nodes is totally stopped, so the percentage of node 3's cooperation is 0% (of the total time). As far as node 4 is concerned, five situations are considered, most of them offering the other nodes the chance to reply with a *tit for tat* strategy: (i) node 4 never cooperates, hence requests of relay are never forwarded and the percentage of cooperation is 0%; (ii) node 4 follows a switching behavior: each 3 minutes interval, it defects for the first 2 minutes and then cooperates for the remaining minute, for a total percentage of cooperation of 33%; (iii) node 4 still switches its behavior: each 2 minutes interval, it defects and cooperates in equal parts, arriving at a cooperation percentage of 50%; (iv) node 4 switches its behavior in a way that is opposite to the one described in the second item of this list: each 3 minutes interval, node 4 defects for the first minute and cooperates for the last 2 minutes, hence cooperating for 75% of the time; (v) node 4 always cooperates: all relay requests are served (for a final percentage of cooperation of 100%).

Two equal sessions of constant bit rate traffic are activated between node 4 and node 0 and node 1 and node 7, respectively at time 1.0 and at time 2.0. In the ideal situation of all cooperating nodes, the shortest paths would be $(4, 3, 0)$ and $(1, 2, 4, 7)$. However, node 3 is always defecting, so the path $(4, 3, 0)$ turns out to be unavailable and the traffic coming from node 4 is forced along the other available path $(4, 2, 1, 0)$. As long as node 1 does not generate traffic, it does not have the chance to track the behavior of node 4, so the relay requests coming from node 4 are regularly served. At time 2.0 node 1 begins the discovery of paths to reach node 7. Besides the other choices, the best path $(1, 2, 4, 7)$ is soon discovered and selected to immediately generate traffic. If node 4 follows a switching behavior, then node 1 has the chance to react in compliance with the *tit for tat* strategy. Notice that in case node 4 is in a defecting state, node 1 still can send traffic to the destination along the path $(1, 2, 5, 6, 7)$.

In the first graph we evaluate the goodput of node i as the ratio $G_i(t) = r_i(t)/s_i(t)$ at the end of the experiment ($t = 12min$) between the number of bytes correctly received at destination and the total number of bytes sent. The x axis represents the percentage of node 4's cooperation, the y axis is the final goodput $G_i(t)$. On the left hand side of Fig. 3, node 4 is fully defecting; the same applies to node 3. Traffic from node 4 towards node 0 is regularly sent between time 1.0 and time 2.0 because node 1 did not generate any request and did not yet check the behavior of the other nodes. At time 2.0, however, node 1 tries to send traffic to node 7 and hence has the chance to verify the behavior of the other nodes. Among the other discovered paths, it realizes that paths comprising nodes 4 and 3 are not working, so as soon as the timeout expires it marks nodes 3 and 4 as defecting and immediately stops relaying traffic coming from node 4. The final goodput G_1 of node 1 is closer to the ideal value because the alternative path $(1, 2, 5, 6, 7)$ is soon discovered and used for the entire duration of the experiment. Goodput G_4 is instead severely reduced. As node 4's percentage of cooperation increases up to 100%, goodput G_4 increases until it reaches a value close to goodput G_1 when there is full cooperation. Although node 3's defection makes the path $(1, 3, 0)$ unavailable, the routing protocol discovers the alternative path $(4, 2, 1, 0)$ composed of cooperative nodes, while the shortest $(1, 2, 4, 7)$ is regularly available in this case. This is the only situation in which node 4 maximizes its goodput. In the intermediate cases the trend is linear

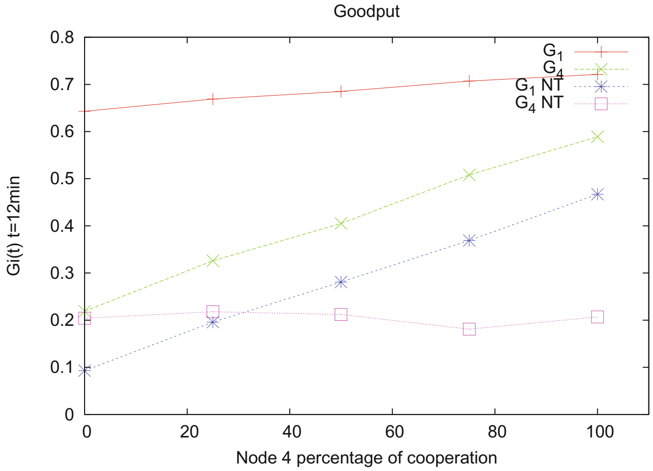


Fig. 3. Goodputs of Node 1 and Node 4

and clearly demonstrates the correct implementation of the *tit for tat* reaction mechanism, as node 1 cooperates only when node 4 does the same. Goodput G_1 remains more or less unaltered independently of node 4's behavior, thanks to the fact that node 1 has a chance to discover alternative cooperative paths. We compared these results with the situation in which the nodes are unable to detect the defecting behavior. We mark these sessions with NT in the same figure 3. The situation is now opposite to the previously analyzed case because goodput G_4 outperforms G_1 in the case of node 4's full defection. Node 1 is now unaware of node 4's defection; hence, while its traffic is not relayed, it regularly relays the incoming packets having node 4 as source. Anyway, both goodputs G_1 and G_4 are lower than in the previous case. This time the lack of tracing of nodes defection affects even node 4's performance, because such node tries to forward traffic not only along the path $(4, 2, 1, 0)$ but also along the uncooperative path $(4, 3, 0)$, which explains the halved final goodput.

We now introduce a new function to evaluate node energy consumption with respect to the various levels of cooperation. We define the parameter S_i computed at the end of the experiments as:

$$S_i = \frac{Ec_i}{(s_i + rl_i)} * \frac{s_i}{r_i}$$

being Ec_i the energy consumed by node i, s_i the total number of bytes sent to the destination, rl_i the number of bytes relayed from node i, and r_i the bytes correctly received at destination. S_i has a dimension of $[Joule/bytes]$ and represents the energy spent by a node to successfully deliver a byte to the destination. We show in figure 4 the value S_1 and S_4 calculated at the end of the experiments and for all the aforementioned combinations of cooperation. From the figure we can observe a significant difference between the energy consumed to deliver one

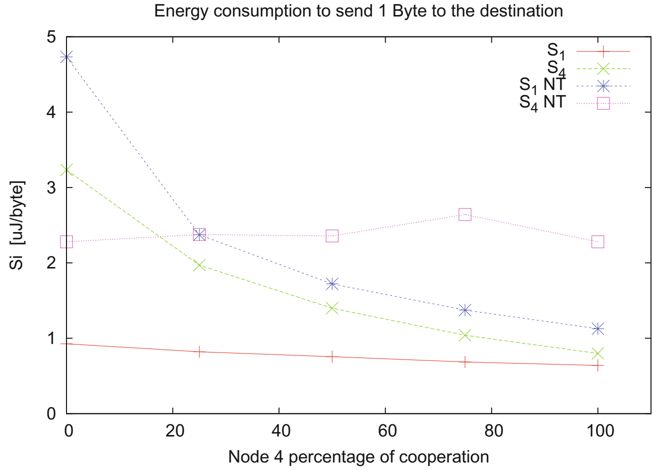


Fig. 4. Energy consumed to deliver a single byte

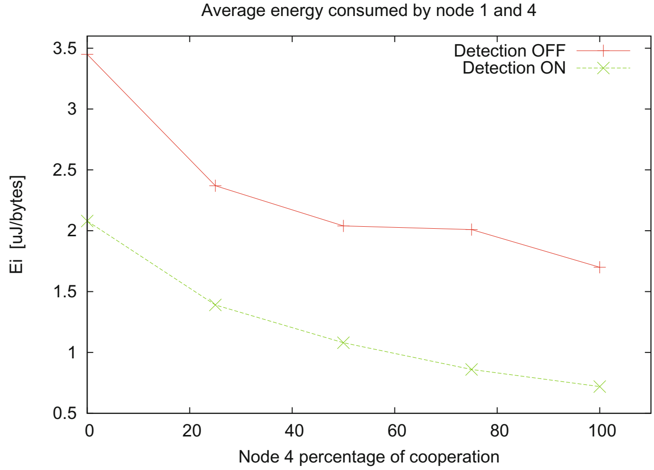


Fig. 5. Average consumed energy of node 1 and 4

byte to the destination in favor of node 1 when node 4 does not cooperate at all. This difference reduces as node 4's percentage of cooperation increases, and it becomes negligible when both cooperate. Even in this case, the trend along the intermediate situations seems linear.

We compared again these results with the analogue experiments executed without the detection of defecting nodes. Energy consumption at node 1 is severely affected, especially when node 4 does not cooperate at all. The convenience of node 4 limits to the single case of total defection at the expenses of the unaware node 1; however, starting from the next experiment and until full cooperation is reached, node 1 again saves more energy than node 4.

Finally, notice that if we consider the average energy consumed by both node 1 and node 4 in case the detection of defecting nodes is enabled, such value is always lower compared to the case when detection is disabled, as showed in Fig. 5. The energy consumed by node 1, the only node which always cooperates, is always lower than that consumed by node 4. Furthermore, node 4's consumption reaches its lowest value when the cooperation is full.

6 Conclusions

In this paper we demonstrated through simulations that cooperation actually acts as an incentive for ad hoc network nodes, since it allows for a lower average energy expenditure per byte transmitted. We also studied the positive impact of cooperation on goodput, which is considered a key performance indicator for any networked environment. Namely, we proved that the resulting network equilibrium achieved in the presence of cooperative nodes increases fairness in terms of energy consumed per unit of successfully delivered packets.

This work is clearly just a first attempt at studying the many facets of cooperation in ad hoc networks. Among the numerous improvements that we identified and which represent directions of our future work, we firstly mention a more detailed analysis of the dependence of the performance improvements deriving from cooperation on the specific network topology taken into account. Apart from this, we also intend to study how the specific location of a node in the ad hoc network topology affects its performance and consequently its willingness to cooperate. This requires that a thorough analysis of the tradeoff between relaying other nodes' packets and sending one's own data is conducted.

References

1. Olivero, F., Romano, S.P.: A reputation-based metric for secure routing in wireless mesh networks. In: IEEE GLOBECOM 2008, pp. 1–5 (December 2008)
2. Mandalas, K., Flitzanis, D., Marias, G.F., Georgiadis, P.: A survey of several cooperation enforcement schemes for MANETs. In: IEEE Int. Symp. on DOI, pp. 466–471 (2005)
3. Srinivasan, V., Nuggehalli, P., Chiasserini, C.F., Rao, R.R.: Cooperation in wireless ad hoc networks. In: INFOCOM 2003, vol. 2, pp. 808–817 (April 2003)
4. Komathy, K., Narayanasamy, P.: Trust-based evolutionary game model assisting aodv routing against selfishness. J. Netw. Comput. Appl. 31(4), 446–471 (2008)
5. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: ACM MobiCom 2000, New York, USA, pp. 255–265 (2000)
6. Zhong, S., Yang, Y., Chen, J.: Sprite: A simple, cheat-proof, credit-based system for mobile ad hoc networks. In: INFOCOM 2003, vol. 3, pp. 1987–1997 (2003)
7. Buttyán, L., Hubaux, J.P.: Stimulating cooperation in self-organizing mobile ad hoc networks. Mob. Netw. Appl. 8(5), 579–592 (2003)
8. Buchegger, S., Le Boudec, J.-Y.: Performance analysis of the confidant protocol. In: ACM MobiHoc 2002, New York, USA, pp. 226–236 (2002)
9. Axelrod, R.: The Evolution of Cooperation. Basic Books (1988)

10. Axelrod, R., Dion, D.: The further evolution of cooperation. Science 242(4884), 1385–1390 (1988)
11. Gelenbe, E., Lent, R.: Power-aware ad hoc cognitive packet networks. Ad Hoc Networks 2(3), 205–216 (2004)
12. Gelenbe, E., Lent, R., Xu, Z.: Design and performance of cognitive packet networks. Perform. Eval. 46(2-3), 155–176 (2001)
13. Nash, J.: Non-Cooperative Games. The Annals of Mathematics 54(2), 286–295 (1951)
14. Nash, J.F.: Equilibrium Points in n-Person Games. Proceedings of the National Academy of Sciences of the United States fo America 36(1), 48–49 (1950)
15. Urpi, A., Bonuccelli, M., Giordano, S.: Modelling Cooperation in Mobile Ad Hoc Networks: A Formal Description of Selfishness. In: Proceedigns of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (2003)
16. Félegyházi, M., Hubaux, J.P., Buttyán, L.: Nash Equilibria of Packet Forwarding Strategies in Wireless Ad Hoc Networks. IEEE Transaction on Mobile Computing 5(5), 463–476 (2006)