A Game Theoretic Trust Model for On-Line Distributed Evolution of Cooperation in MANETs

Marcela Mejia[a,*], Néstor Peña[a], Jose L. Muñoz[b], Oscar Esparza[b], Marco A. Alzate[c]

[a] *Universidad de los Andes, Colombia*
[b] *Universitat Politècnica de Catalunya, Spain*
[c] *Universidad Distrital, Colombia*

Abstract

Cooperation among nodes is fundamental for the operation of mobile ad hoc networks (MANETs). In such networks, there could be selfish nodes that use resources from other nodes to send their packets but that do not offer their resources to forward packets for other nodes. Thus, a cooperation enforcement mechanism is necessary. Trust models have been proposed as mechanisms to incentive cooperation in MANETs and some of them are based on game theory concepts. Among game theoretic trust models, those that make nodes' strategies evolve genetically have shown promising results for cooperation improvement. However, current approaches propose a highly centralized genetic evolution which render them unfeasible for practical purposes in MANETs. In this article, we propose a trust model based on a non-cooperative game that uses a bacterial-like algorithm to let the nodes quickly learn the appropriate cooperation behavior. Our model is completely distributed, achieves optimal cooperation values in a small fraction of time compared with centralized algorithms, and adapts effectively to environmental changes.

Keywords: MANET, trust models, game theory, evolutionary algorithm

1. Introduction

Mobile Ad Hoc NETworks (MANETs) are infrastructureless networks formed by wireless mobile devices with limited resources. Source/destination pairs that are not within transmission range of each other must use intermediate nodes as relays (Perkins, 2001). The cooperation among nodes is fundamental for the operation of MANETs, since nodes that contribute with their own limited resources, such as battery, memory and processing capacity, should be able to use the resources contributed by other nodes. However, in this environment, there could be free-riders or selfish nodes, i.e., users that want to maximize their own welfare by using resources from the network to send their own packets without forwarding packets on behalf of others (Wrona and Mähönen, 2004). Thus, it is important to encourage nodes to participate in essential network functions such as packet routing and forwarding, because the higher the cooperation the better the network performance. In this sense, several trust mod els have been proposed as mechanisms to incentive node participation within the network (Mejia et al., 2009b). A trust

*Corresponding author. Tel +573005645704.
Email address: am.mejia75@uniandes.edu.co (Marcela Mejia)

model is a conceptual abstraction to build mechanisms for assigning, updating and using trust levels among the entities of a distributed system. In other words, the trust model allows the establishment of trust relationships among different entities with some degree of credibility, for a given action (Marti and Garcia-Molina, 2006).

Among proposals in the literature, trust models based on game theory are interesting because they properly model the dilemma that a node has: to cooperate and gain trust, or not to cooperate and save battery. To face this dilemma, several pure and mixed strategies has been studied in game theory, especially different forms of tit for tat (Osborne, 2004). However, in a MANET, game conditions can change over time, for which we focus on trust models that use genetic algorithms to evolve the strategies. These models are promising in MANET because they can dynamically adapt to the current network conditions.

In this article, we propose a trust model based on (1) a game, (2) a trust evaluation mechanism, (3) a way of coding the cooperation strategy and (4) a genetic algorithm that allows a node to evolve its cooperation strategy. The proposed evolution algorithm[1] introduces a low overhead because it is completely distributed and easy to compute. Indeed, it only requires local exchanges among neighbors. Our algorithm works much like plasmid migration in bacterial colonies (Dale and Park, 2004; Marshall and Roadknight, 2000) combined with a parallel cellular genetic algorithm (Alba and Dorronsoro, 2008 ; Alba and Troya, 1999; Cantupaz, 1998; Nowostawski and Poli,1999). Through this procedure, individual nodes select the strategies that locally maximizes their payoff in terms of both packet delivery and resource saving. The general idea is that, within the network, there are some nodes that are willing to cooperate if deemed worthy, called normal nodes. There are also some nodes that are expecting to use the resources of normal nodes without contributing to the network, called selfish nodes. The network environment is characterized by the fraction of selfish nodes within the total population of nodes in the network. For a given environment, we measure the cooperation as the fraction of packets originated by normal nodes that effectively reach their destinations. We measure the resource savings in terms of the detection and isolation of selfish nodes, since serving selfish nodes constitutes a waste of resources. Although these are global variables, the local payoff maximization is such that the whole network increases the cooperation and, consequently, the throughput, without wasting scarce energy resources on selfish nodes. Indeed, our trust model efficiently achieves both high adaptability to environment changes and quick convergence to almost optimal cooperation and energy saving values, as we show by simulation.

It is important to mention that we are concerned only with selfish nodes, not malicious nodes. Selfish nodes are free-riders that want to consume other nodes resources without supporting the network with their own resources, but they are not interested in undermining network security or causing any damage.

The rest of the article is organized as follows. Section 2 briefly reviews models for cooperation and genetic algorithms to put our work in context. Section 3 describes our game theoretic trust model. Section 4 presents the simulation scenarios and their parameters. Section 5 shows the performance evaluation for our trust model and compares it with the previously published results of a centralized evolution model. Section 6 concludes the article.

[1] A preliminary version of the algorithm was presented in a conference paper (Mejia et al.,2009a).

2. Background

In this section we briefly review cooperation models and genetic algorithms, to put in context our algorithm which uses a non-cooperative game trust model and an hybrid cellular/bacterial evolution mechanism.

2.1. Models for Cooperation

Models for cooperation enforcement in MANETs can be broadly divided in two categories according to the techniques they use to enforce cooperation (Marias et al., 2006): credit-based models and trust models. The former category is based on economic incentives, whereas the later is based on building reputation to enforce cooperation.

In credit-based models, the network tasks are treated as services that can be valued and charged. These models incorporate a form of virtual currency to regulate the dealings among nodes. The most widely-cited proposal of this type was introduced by Buttyan and Hubaux (Buttyán and Hubaux, 2000). These authors introduce a currency called nuglets. The exchange of nuglets relies on a tamper-resistant security subsystem being present in every node. Another credit-based proposal called Sprite (Yale and Zhong, 2002) makes use of a public key infrastructure to deal with the problem of selfishness. Nodes upload receipts to a Credit Clearance Service (CCS), a central authority which is available when the nodes are connected to the Internet. Express (Janzadeh et al., 2009) is a work based on Sprite. Express tries to minimize the cost of digital signatures by using hash chains. Express also uses an external trust entity called Reliable Clearance Center (RCC). In general, the main drawback of credit-based models is that they require the existence of either tamper-resistant hardware or a virtual bank, heavily restricting their usability for MANETs. An hybrid scheme called OCEAN (Bansal and Baker, 2003) uses both reputation to detect and punish selfish behavior, and a micro-payment component to encourage cooperation. The credit is earned for each immediate neighbor and it cannot be used to send packets in a different route. A reputation scheme optimized for video streaming inspired in OCEAN has been presented in (Mu et al., 2010). Another model called ad hoc-VCG (Anderegg and Eidenbenz, 2003) is also a credit-based model which introduces a second-best sealed type of auction. To this respect, a pricing question arises concerning the amount of the payment a node should ask to forward packets. Intermediate nodes declare their respective prices honestly. Honest behavior is assured by VCG mechanism since ad hoc-VCG is robust when only one cheating node exists. However, it might fail in the presence of collusions of nodes trying to maximize their payments. An additional issue is the excessive overhead because ad hoc-VCG requires complete knowledge of the network topology during the route discovery phase.

On the other hand, models in which trust is the base for cooperation are envisioned as the most promising solutions for MANETs because these models do not have the restrictions of credit-based models. Trust models can frustrate the intentions of selfish nodes by coping with observable misbehaviors. If a node does not behave cooperatively, the affected nodes, reciprocally, may deny cooperation. Generally speaking, in a trust model, an entity called the Subject S commends the execution of an action a to another entity called the Agent A, in which case we say that $T\{S : A, a\}$ is the trust level that S has on A with respect to the execution of action a (Sun et al., 2006). This trust level varies as the entities interact with each other; i.e., if the Agent A responds satisfactorily to the Subject S, S can increase the trust level $T\{S : A; a\}$. On the other hand, if the subject S is disappointed by the agent A, the corresponding trust level could be decreased by some amount. In this sense, a trust model helps the subject of a distributed system to select the most reliable agent among several agents offering a service (Marti and Garcia-Molina,

2006). To make this selection, the trust model should provide the mechanisms needed for each entity to measure, assign, update and use trust values. Several trust models have been proposed in the literature for improving the performance of MANETs. In (Mejia et al., 2009b), there is classification of trust-based systems based on the theoretical mechanisms used for trust scoring and trust ranking. Following this classification, we can further divide the trust-based proposals in approaches based on social networks, information theory, graph theory and game theory.

In proposals based on social networks, nodes build their view of the trust or reputation not only taking into account their own observation but also considering the recommendations from others. One of the first examples of a trust system based on social networks is CONFIDANT (Buchegger and Le Boudec, 2002). CONFIDANT works as an extension of a reactive source-routing protocol for MANET. Nodes monitor the next node on the route by either listening to the transmission of the next node or by observing route protocol behavior. Any misbehaving action generates ALARMs. Each node maintains received ALARMs from friend nodes and ALARMs produced by the node itself. In the improved CONFIDANT (Buchegger and Boudec, 2004), authors provided a modified Bayesian approach for reputation representation, updates, and view integration. When updating the reputation according to recommendations, only information that is compatible with the current reputation rating is accepted. This approach is objective and robust, but it still leaves an opportunity for elaborate attackers to launch false accusation attacks (Li and Wu, 2010). In (Michiardi and Molva, 2002), authors propose CORE. CORE relies on observations and recommendations which are combined by a specialized function. The CORE scheme is immune to some attacks because no negative ratings are spread, and, thus, it is impossible for a node to maliciously decrease another nodes reputation. However, two or more nodes may collude (i.e., send positive rating messages) in order to increase their reputation. To prevent such phenomena, the CORE implicitly provides some protection, since subjective reputation has more impact (i.e., weight) than the indirect. Finally, SORI (Bansal and Baker, 2003) is another trust model that also uses reputation spreading. In addition to the previous ones, there are some trust models that are based on social networks and that they also use cluster-heads. The cluster-head is a node who is elected to play a special role regarding the management of recommendations. An example can be found in (Safa et al., 2010). In this proposal, the protocol organizes the network into one-hop disjoint clusters then elects the most qualified and trustworthy nodes to play the role of cluster-heads. The proposed mechanism continuously ensures the trustworthiness of cluster-heads by replacing them as soon as they become malicious. As a concluding remark for trust models based on social networks, we would like to notice that the calculation and measurement of trust in unsupervised ad-hoc networks involves a very complex aspect like rating the honesty of recommendations provided by other nodes. Although there are efforts like (Luo et al., 2009), (Li and Wu, 2010) and (Zouridaki et al., 2009) that try to alleviate this problem, it is still a hard problem for systems that use recommendations. Furthermore, social trust models that also use clusters add another problem, they require a dealer, which must be involved in the working of other nodes, and this is hard to achieve in practical ad hoc networks.

Regarding the proposals based on information theory, one of these is (Sun et al., 2006), in which the authors proposed a trust model to obtain a quantitative measurement of trust and its propagation through the MANET. However, the proposal is theoretical and it does not include an implementation specification. (Sherwood et al., 2006) describes a trust inference algorithm in terms of a directed and weighted Trust Graph, T, whose vertices correspond to the users in the system and for which an edge from vertex i to vertex j represents the trust that node i has in node j. However, covering the whole graph is still a high complexity computational problem.

Finally, there are several proposals that use game theory. These proposals can be further

divided into cooperative and non-cooperative games (Osborne, 2004). In cooperative games, users form coalitions so that a group of players can adopt a certain strategy to obtain a higher gain than the one it may be obtained making decisions individually. In cooperative games, the nodes need to communicate with each other and discuss the strategies before they play the game. (Baras and Jiang, 2005) and (Saad et al., 2009) are representative proposals of cooperative games for MANET. Nevertheless, this type of games have the disadvantage of generating network overhead due to the complex processes of coalition management. On the other hand, non-cooperative games are especially suitable in scenarios in which players might have conflicting interests, and each of them wants to maximize her own profit taking individual decisions (Felegyhazi et al., 2006). Our proposal and many others that can be found in the literature (Seredynski and Bouvry, 2009),(Seredynski et al., 2007),(Milan et al., 2006),(Komathy and Narayanasamy, 2008), (Wang et al., 2010),(Ji et al., 2010), (Jaramillo and Srikant, 2010) are based on a variation of the classical non-cooperative game of the iterated prisoner's dilemma game, in which nodes can use different strategies. A further discussion of these proposals and a comparison with ours is provided in Section 5.3. It is worth to mention that, in general, all the mentioned models improve cooperation results, but they still have some difficulties to be applied in MANET.

In this article, we focus on trust models based on game theory, since they capture very well each node dilemma about deciding whether to cooperate and obtain the trust of peer nodes, or not to cooperate and save scarce energy resources. In particular, we focus on non-cooperative games since nodes rely only on private histories and thus, the costly coalition overhead and possible conflicting interests can be avoided. More specifically, trust models that use genetic algorithms for the evolution of strategies show promising results in MANET because they can adapt dynamically the behavior of nodes to the current conditions of the network. In particular, we take as reference model the centralized one presented in (Seredynski et al., 2007). This model is interesting because the evolution algorithm achieves promising results regarding cooperation and energy saving. However, we are still concerned about the highly centralized nature and the slow convergence of its evolution algorithm. Optimal strategies are obtained by using a centralized entity that runs a conventional genetic algorithm in an off-line way after a large number of interactions between nodes. For these reasons, in this article we propose a non-cooperative game theoretic trust model in which strategies evolve on-line according to a distributed genetic algorithm without requiring too much data exchange among nodes. Our model exploits the distributed nature of a MANET by using a local genetic information exchange. Our algorithm works much like plasmid migration in bacterial colonies (Dale and Park, 2004; Marshall and Roadknight, 2000) combined with a parallel cellular genetic algorithm (Alba and Dorronsoro, 2008; Alba and Troya, 1999; Cantupaz, 1998; Nowostawski and Poli, 1999). Through this procedure, individual nodes select the strategies that locally maximizes their payoff in terms of both packet delivery and resource saving. A brief summary about Genetic algorithms is introduced in the rest of this section to better understand the proposal.

2.2. Genetic algorithms

In many complex optimization problems, an exhaustive search of the solution space is unfeasible. Genetic algorithms (GAs) are heuristic approaches that are based on the genetic hereditary processes of biological organisms. A canonical GA works with a population of individuals, where each individual represents a possible solution to a given problem. A fitness score for each individual is assigned according to how it fits as a solution to the problem. The higher the fitness an individual has, the higher the opportunity it has to be selected for reproduction. This reproduction is done by crossing two individuals of the population over, generating new individuals

as offspring that contain the best characteristics of the previous generation. As generations come and go, these best characteristics are spread throughout the entire population. By favoring the mating between the fittest individuals, it is possible to explore the most promising regions of the solution space.(Holland, 1975; Whitley, 1993)

Generally, a GA performs four steps to obtain a new population: initialization, evaluation, selection and reproduction. The process is recursively repeated from the second step during a given number of generations or until the solution converges.(Holland, 1975)

I. *Initialization*. In a GA, a genetic code, or chromosome, represents any solution to the problem. The first population of chromosomes is typically randomly generated.
II. *Evaluation.* The mechanism used to measure the individual fitness to solve the given problem is called fitness function.
III. *Selection.* The selection algorithm picks individuals out among the current population to participate in the next generation. The most common approach is a roulette-wheel selection, where a selection probability is assigned to each individual chromosome, proportionally to its fitness.
IV. *Reproduction.* This is the process by which two parent chromosomes are recombined, normally through the genetic operators of crossover and mutation. Crossover is a sexual reproduction between two parents, where some portions of the parents chromosomes are swapped to form a child. Then, the mutation operator alters each child gene with a small probability in order to ensure that no point in the search space has a zero probability of being examined.

2.2.1. Parallel Genetic Algorithms

Parallel genetic algorithms (PGA) are not only an extension of canonic GA, but also a different efficient way to search the space of solutions for a given problem (Nowostawski and Poli, 1999). In a PGA, the population is divided into subpopulations and an independent GA is performed on each of these subpopulations. The local selection and reproduction rules allow the species to evolve locally, and diversity is enhanced by migration, i.e., by transferring genetic information among subpopulations. The processes of genetic evolution and migration are repeated until the solution converges.

The many parallel genetic algorithm models that can be found in the literature can be classified in island or cellular depending on their parallelism grade (Alba and Troya, 1999). In an island PGA (iPGA), the whole population is divided into a few separated subpopulations, or islands, with many individuals on it, and an independent genetic algorithm is performed in each island. After a certain number of generations, there is a migration process by which the fittest individuals migrate among islands in order to obtain genetic diversity (Nowostawski and Poli, 1999). In a cellular PGA (cPGA), the population is divided into a large number of subpopulations with a few individuals on it, and the genetic information exchange among subpopulations is performed by overlapping subpopulations (Alba and Troya, 1999). The population in a cPGA has a spatial structure that limits the interactions among individuals to just some small neighborhoods. However, by neighborhood overlapping, optimal local solutions can spread across the entire population (Nowostawski and Poli, 1999).

2.2.2. Plasmid Migration and Bacterial Genetic Algorithms

There are several genetic algorithms based on observed bacterial behavior. Different authors present them as microbial genetic algorithms (Harvey, 1996), bacterial algorithms (Cabrita et al.,

2003), pseudobacterial genetic algorithms (Nawa et al., 1999), lateral gene transfer (Ochman et al., 2000) and plasmid migration (Marshall and Roadknight, 2000) among others. In general, these algorithms avoid the sexual reproduction, so they are not classified as parallel genetic algorithms, although they are extremely distributed. Here we describe two of these algorithms, those on which we based part of our evolution process.

Plasmid Migration (PM) is based on the behavior of some bacteria (Marshall and Roadknight, 2000). Plasmids are self-replicating extrachromosomal DNA molecules that are not essential for the survival of the bacterium but encode a wide variety of genetic strains that permit a better survival in adverse environments. Plasmid has the ability to be transferred among bacteria within the same generation by allowing healthy individuals to deposit plasmids in the medium, so that less healthy individuals can take these plasmids from the medium. This ability gives bacteria a great adaptability to sudden environmental changes (Dale and Park, 2004). This analogy can be used in genetic algorithms to spread high quality strands from fitted individuals to the rest of the population through the gene transfer operation.

On the other hand, in a bacterial algorithm, a chromosome is divided in p parts, and each individual produces $m-1$ clones of itself. The randomly chosen i_{th} part of the $m-1$ clones is mutated and the best fitted part is replicated in the m individuals. After this mutation-evaluation-selection-replacement process is repeated for all the p parts, the fittest individual goes to the next population and the other $m-1$ individuals die (Nawa et al., 1999; Cabrita et al., 2003).

Both plasmid migration and bacterial algorithms are greedy algorithms that, at each step, make apparent good decisions without regarding for future consequences and, as such, can lead only to locally optimal solutions. In contrast, these solutions can be obtained very quickly, enhancing adaptability at the cost of optimality (Weiss, 1998). However, in many occasions, in a well designed plasmid migration algorithm, the mobility of the individuals allows good plasmid to spread all over the population, so that better solutions can be obtained through a more exhaustive search of the solution space. In this article, we propose an enhanced cPGA algorithm that includes some greedy bacterial heuristics to achieve fast convergence, optimality and adaptability.

3. A Model for On-Line Distributed Evolution of Cooperation

In this section we describe our proposal, where both the trust evaluation mechanism and the strategy evolution algorithm are designed to enhance the convergence speed and adaptability. Furthermore, both trust evaluation and strategy evolution are carried out in a distributed way among the nodes of the network. In our trust model, the interactions among nodes are based on the iterated prisoner's dilemma under the random pairing game (Ishibuchi and Namikawa, 2005). Each intermediate node utilizes a strategy that defines whether it should retransmit or discard a packet that comes from a certain source node. The strategy depends on two aspects: the past behavior of the network when the intermediate node acted as a source, and the trust level that the intermediate node has in the source node. The model is comprised of: (a) a trust evaluation mechanism; (b) a game based network model; (c) a strategy; and (d) a local genetic algorithm based on plasmid migration to evolve the strategy in a highly distributed way.

We take as reference the centralized model of (Seredynski et al., 2007). Essentially, the similarities between the centralized model and ours are in the game model and in the strategy encoding. These similarities are kept to make the performance comparison easier (see evaluation results in section 5). However, both the trust evaluation mechanism (for better adaptability) and the genetic evolution (for easy distributed computation) are totally different. On the other

hand, the more close related work is the BNS proposal (Komathy and Narayanasamy, 2008), which also uses a non-cooperative game model with a distributed evolution. In BNS, a node decides the strategy to follow by changing to other player's strategy if it seems to be doing better. Despite the distributed nature of this evolution strategy, the payoff structure strictly encourages cooperating strategies without taking into account the energy savings of discarding strategies. A more detailed comparison with BNS and with other proposals that also use non-cooperative game models is provided in Section 5.3.

3.1. Trust evaluation mechanism

Each node maintains a trust table based on the observed behavior of its neighbors. For example, if node B is observing node A, which is within its transmission range, it can know the number of packets that has been sent to A to be forwarded, n, and the number of packets that A has actually forwarded n_A. So B can compute the forwarding rate of A, as show in Eq.(1)

$$f_r(B, A; n) = \frac{n_A}{n} \tag{1}$$

We can use a simple cumulative average to compute the forwarding rate. For instance, B can update the forwarding rate of A by observing whether the n^{th} packet has been forwarded by A or not. This is done by applying Eq.(2):

$$f_r(B, A; n) = \frac{1}{n}\sum_{i=1}^{n} d_i = \frac{(n-1)\, f_r(B, A; n-1) + d_n}{n} \tag{2}$$

where $d_i \in \{0, 1\}$ is the i^{th} observed decision.

Since the strategies are modified continuously by the nodes to adapt to environmental changes, it would be unfair to have a long record of observed decisions if they were taken under a previous strategy, different to the current one. Correspondingly, we decided to take into account only the most recent m observed decisions, so we compute the forwarding rate as the moving average of the decision sequence, i.e., the fraction of retransmitted packets among the previous m packets received for forwarding.

The value of the memory depth, m, obeys a trade-off: we would like m to be large enough to obtain a fair evaluation of the forwarding rate, but we would also like m to be small enough to ensure that the forwarding rate actually corresponds to the current strategies (tuning of m is discussed in Section 4). Computing the forwarding rate with a finite memory of depth m, requires both the previous forwarding rate and the last m decisions as state variables, as we show in Eq.(3):

$$\hat{f}_r(B, A; n) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} d_i = \frac{(n-1)\,\hat{f}_r(B,A;n-1)+d_n}{n} & n \le m \\ \\ \frac{1}{m}\sum_{i=0}^{m-1} d_{n-i} = \frac{m\,\hat{f}_r(B,A;n-1)+d_n-d_{n-m}}{m} & n > m \end{cases} \tag{3}$$

With the current rate $\hat{f}_r(B, A; n)$, B can determine the trust level it should have in A, $T\{B : A; n\}$, as shown in Table 1. For comparison purposes, we keep the same ranges on the fraction of forwarded packets and the same corresponding trust values as in the centralized model.

Finally, it is worth to mention that using a moving average that only takes into account the previous m observed decisions may seem a subtle detail, but we introduce it because it is critical for the on-line adaptability of the strategies, as we will show in Section 5.

Table 1: Relation between delivery rate and trust level

$\hat{f}_r(B, A; n)$	$T\{B : A; n\}$
0.9 – 1	3
0.6 – 0.9	2
0.3 – 0.6	1
0 – 0.3	0

3.2. Game-based network model

Each intermediate node that receives a packet should decide whether to forward it or to discard it, according to its strategy. Each game starts with the transmission of a new packet from a source node and ends either when the packet is delivered to its destination, or when an intermediate node decides to discard the packet. Once the game has finished, each participant receives a payoff according to the decision it took and its trust level on the source node. Since there are two types of nodes, source nodes and intermediate nodes, two types of payoff tables are maintained, as shown in Table 2 (Seredynski et al., 2007). These payoffs have been directly taken from the centralized model for comparison purposes and also because they have two good properties: a successful transmission is the most rewarding event, and there is symmetry between the discarding and forwarding payoffs with respect to the trust value, indicating that saving energy is as important for each n ode as obtaining the trust of its neighbors.

Table 2: Tables of payoff

Source Node Payoffs

Transmission Status	
Successful	5
Failed	0

Intermediate Node Payoffs

	Trust Level of the Source Node			
	$T = 3$	$T = 2$	$T = 1$	$T = 0$
Cooperate	3	2	1	0.5
Discard	0.5	1	2	3

The forwarding or discarding decision of an intermediate node is observed by all nodes preceding it in the path. In addition, a node that receives a packet can consider that all its preceding nodes have cooperated.

3.3. The strategy

We use a strategy codification similar to the one used in the centralized model. The strategy that a node follows when it is acting as intermediate node is encoded by a string of bits, in which each bit represents the decision of discarding (D) (bit = 0) or cooperating (C) (bit = 1). The strategy depends on the following parameters:

- The trust level that the node has in the source node.
- The transmission status of the two previous games that the node has played as source, which could be success (S) or failure (F).

The resulting strategy has 16 bits, as shown in the example strategy of Table 3. For instance, according to the strategy of Table 3, an intermediate node will forward a packet if it has a trust level of 1 in the packet's source and if its two previous packet transmissions as source were successful (8^{th} bit of the strategy, from left to right).

The strategy is evolved by means of a genetic algorithm. A possibility is to randomly choose an initial strategy and then start evolving it. However, the convergence speed to optimal cooperation can be improved if some bits of the initial strategy are set to particular values. In addition, nodes cooperate if they have not played two times as source yet (we discuss these issues in Section 3.4).

Table 3: Strategy coding, example strategy 0001 0011 0101 0111

Source Trust Level	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
Transmission Status -2	*F*	*F*	*S*	*S*	*F*	*F*	*S*	*S*	*F*	*F*	*S*	*S*	*F*	*F*	*S*	*S*
Transmission Status -1	*F*	*S*	*F*	*S*	*F*	*S*	*F*	*S*	*F*	*S*	*F*	*S*	*F*	*S*	*F*	*S*
Current Decision	*D*	*D*	*D*	*C*	*D*	*D*	*C*	*C*	*D*	*C*	*D*	*C*	*D*	*C*	*C*	*C*

3.4. Distributed Bacterial-like Evolution Algorithm

A genetic algorithm is used to maximize the fitness or mean payoff of each node. The design of the evolution algorithm took us several iteration steps where we probed different design options to balance the different issues of the learning process. As a result, next we present our evolution algorithm and discuss its design. Furthermore, in Section 5, it is shown that our proposal clearly outperforms the centralized algorithm in terms of cooperation, convergence speed, energy saving and adaptability.

In our distributed bacterial-like algorithm, we evolve the strategies on-line during the life of the network. A successive sequence of games takes place during this network life. A game is a successful or failed packet transmission. In a game, a source node selects the most trusted h-hop route among r possible routes and sends its packet through it. The game is successful if the transmitted packet is received at the destination. When a node has played R times as the packet source, we said that it has completed a Plasmid Migration Period (PMP). At this moment, an evolution step must take place, i.e., the node exchanges genetic information with its neighbors and evolves its strategy.

We combine the plasmid migration concept with a classical cellular genetic algorithm by allowing each node to receive the genetic information from all its one-hop neighbors in order to start a reproductive mechanism to construct a new strategy. For reproduction we use the classical one-point crossover and mutation processes. The bacterial plasmid migration is introduced by allowing each node to keep its best previous strategy so that, if during the current plasmid migration period the new strategy did not increase the fitness, the old strategy can be restored. More specifically our model works as shown in Algorithm 1.

At time 0, node i starts with an initial random strategy, $s_i(0)$, whose fitness, $f_i(0)$, is evaluated during the first PMP, i.e., during the transmission of its own first R packets (5^{th} line of Algorithm 1). Like plasmid genes, node i keeps a record of the best proven strategy so far, i.e., if at the j^{th} PMP the current strategy $s_i(j)$ is worst than the previous one, $s_i(j-1)$, node i restores its previous strategy and fitness(7^{th} and 8^{th} lines of Algorithm 1). Node i exchanges its strategy $s_i(j)$ and its corresponding fitness $f_i(j)$ with its one-hop neighbors, $\mathcal{K}$, and selects a pair of potential parents,

Algorithm 1: Bacterial-like evolution pseudo-code

```
   // Node i goes on from the 0th PMP
   inputs: At each PMP, the set of i's current neighbors, K
   outputs: The sequence of strategies s_i(j) and fitnesses f_i(j) at the j^th PMP, for j = 0, 1, 2, ...
 1 begin
 2     s_i(0) ← GenerateStrategy() ; // Generate the initial strategy
 3     j ← 0 ; // Time index, expressed in PMPs
 4     while true do
 5         f_i(j) ← GetFitness(s_i(j)); // Evaluate this strategy (play some games)
 6         if (j > 0) && (f_i(j) < f_i(j − 1)) then // The previous strategy was better
 7             s_i(j) ← s_i(j − 1) ; // Restore the previous strategy
 8             f_i(j) ← f_i(j − 1) ; // and the previous fitness
 9         end
10         K ← GetNeighborhood(i) ;
11         p_1 ← RouletteWheel(K ∪ {i}) ; // Get first parent
12         p_2 ← RouletteWheel(K ∪ {i}) ; // Get second parent
13         while p_1 == p_2 do
14             p_2 ← RouletteWheel(K ∪ {i}) ; // We need two different parents
15         end
16         if f_i(j) ≥ Mean(f_p1(j), f_p2(j)) then
17             s_i(j + 1) ← s_i(j) ; // Changing the strategy is not worthy
18         else
19             s_i(j + 1) ← Crossover(s_p1(j),s_p2(j)) ;
20             s_i(j + 1) ← Mutation(s_i(j + 1)) ;
21         end
22         j++;
23     end
24 end
```

p_1 and p_2, one of which can be node i itself (lines 10^{th} to 15^{th} of Algorithm 1). This selection is performed through a roulette wheel process. In a roulette wheel, a selection probability is assigned to each individual strategy as $p_k = f_k / \sum_{n \in \mathcal{K}} f_n, k \in \mathcal{K}$, and each parent is randomly selected according to this distribution. If the mean fitness of the selected potential parents is worse than the fitness of the current strategy (16^{th} and 17^{th} lines of Algorithm 1), node i keeps its current strategy for the next PMP. Otherwise, the selected parent strategies are combined to construct a child strategy $s_i(j+1)$. In this last case, the function performs a one point crossover where half of the genes are taken from one parent and the other half are taken from the other parent. Finally, the child strategy goes through a mutation process, where each bit is flipped with a given probability. The resultant strategy is the one that will be used in the next PMP. This process is repeated during the life time of the network.

Regarding the initial random strategy, corresponding to the function `GenerateStrategy()` in the 2^{nd} line of Algorithm 1, we make some a priori decisions about some bits in order to speed up the convergence of the algorithm, as shown in Algorithm 2. In particular, we fix six bits of that initial strategy: an intermediate node cooperates when the source node has the highest trust value and the network has delivered at least one of its previous two packets (i.e., according to Table 3, bits 14, 15 and 16 are set to Cooperate -7th line of Algorithm 2). Similarly, an intermediate node discards a packet when the source node has the lowest trust value and the network has failed in

delivering at least one of its two previous packets (i.e., bits 1, 2 and 3 are set to Discard -8th line of Algorithm 2). We have validated these asumptions by simulation since this pattern was found in th e final strategies after convergence under very different simulation scenarios. Finally, the ten remaining bits of the initial strategy are randomly chosen as 0 or 1 with probability 1/2 (lines 3 to 6 of Algorithm 2).

Algorithm 2: Pseudo-code for randomly generating a new strategy

```
1  GenerateStrategy {
2  s ← null ; // strategy to be returned
3  for (bit = 4; bit < 14; bit + +) do
4  |   r ← Random() ; // 0 or 1, each with probability 1/2
5  |   SetStrategyBit(s, bit, r) ; // Set the given bit of s randomly
6  end
7  s ← OR(s, 0000000000000111) ; // conditions for initial collaboration
8  s ← AND(s, 0001111111111111) ; // conditions for initial discarding
9  return s ;
10 }
```

An important issue to address is how a node should behave at the beginning of network operation, when it has tried to send less than two packets as source node, so no information is available about network behavior. In this case, we decided that the node should try to immediately start building a good reputation among neighbors, as it is suggested in (Axelrod and Dion, 1988). Therefore, when the node has transmitted less than two packets, the decision made as intermediate node is always cooperate, regardless of the trust level it has on the current source node. The last issue we have to consider is how to manage the reputation of unknown nodes. In this case, the initial reputation level assigned is *One*.

The steps followed to design our algorithm are briefly summarized next. First, we tried a greedy plasmid migration algorithm in which each node exchanges its strategy with the first neighbor it founds and, if the neighbor's fitness is higher, replaces its own strategy with the acquired one. By simulation, we found that this simple plasmid migration algorithm quickly improved cooperation but the solution achieved was not very close to optimal cooperation. Next, we tried a chromosomal cellular algorithm, in which a standard genetic algorithm was run among neighbors. The genetic information was exchanged among one-hop neighbors and each node performed a roulette-wheel selection, a crossover and a mutation process with the received strategies. This approach yielded strategies that reached higher cooperation ratios than greedy plasmid migration. However, there were two problems: firstly, convergence to the maximum cooperation was slow, and secondly, sometimes cooperation suddenly decayed because nodes were quite disposed to change their strategies (i.e. they did not keep good strategies for too much time).

As a conclusion of the previous discussion, we devise an hybrid algorithm that mixes the previous ones. Our algorithm uses chromosomal crossover and mutation, and also allows for plasmid attributes such as the restoration of previous strategies or the possibility for a child to refuse the genetic material of its parents. This design of the evolution algorithm allowed us to find a good trade-off between the bias error (where the strategies are not close enough to optimal) and the variance error (where the strategies vary even to the point of forgetting good solutions). This process was particularly challenging because, in our problem, we have a double criterion: we need to get both a good cooperation level among normal nodes (to maximize the throughput) and

an effective isolation of selfish nodes (to minimize energy consumption). Indeed, many design options led to truly optimal cooperation values, although the selfish nodes were not completely isolated; other design options we tested were very good in saving energy resources from selfish nodes, but at the cost of lower cooperation among normal nodes. Algorithm 1 is the result of such design procedure, where we weighted those criteria with the additional requirements of fast convergence and good adaptability. Furthermore, the extended exploration of the solution space is due to both mobility and neighborhood overlap. This ensures the convergence to almost optimal cooperation values.

As a result, the foremost characteristics of our evolution algorithm are its distributed implementation and its convergence speed. These features make our algorithm readily implementable in a MANET since we exploit the natural structure of the network in two ways:

- The evolution is carried out in a distributed way through local neighborhoods.
- The genetic information is quickly disseminated all over the network after a few PMPs since these neighborhoods highly overlap through mobility.

4. Simulation

One of the main goals of the work presented in this article is to devise a distributed algorithm to evolve the cooperation strategy of nodes. The requirements for this algorithm are fast convergence to maximum cooperation and high adaptation to changing cooperation conditions. To test and develop our design, we could have used one of several powerful simulators such as ns-2 (NS2, 2009), Opnet (OPNET, 2009) or Qualnet (QUALNET, 2009), among others. All of them are known for having appropriate libraries for wireless ad hoc networks. However, we decided to develop a custom-made simulator in Java with a simplified network model (about mobility, routing etc.) because we wanted to measure, in an isolated way, the cooperation achieved by our proposal with respect to the maximum theorical cooperation. Also, a controlled design of the network allows us to observe and analyze the effects of our design choices isolated from the interactions of physical, multi-access, routing and transpo rt protocols. In particular, in our simulator, the mobility model and routing protocol are represented by the random selection of paths and neighborhoods. Finally, it is worth to mention that not only our algorithm has been implemented but also the centralized one for comparison purposes. The simulator and its main parameters are descrived next.

4.1. Simulator

In our simulation platform the network is composed of a population of mobile nodes divided into two groups: the normal nodes, which are willing to cooperate if deemed worthy, and the selfish nodes, which are free-riders that discard every packet in transit. The environment, which is characterized by the fraction of selfish nodes among the whole population, can be programed to change dynamically during the simulation by the conversion of some nodes from normal to selfish and from selfish to normal. The network simulation progresses through the execution of games, or packet transmissions, between randomly selected pairs of source/destination nodes. For each source/destination pair, several randomly selected paths are established. In a single game, once a path has been chosen, the source node transmits its packet and each intermediate node in the path decides to forward or discard the packet according to its own strategy. As mentioned in section 3.2, the forwarding or discarding decision of an intermediate node is observed

by all nodes in the path up to the destination (if the packet arrived there) or to the discarding node, if the transmission failed. The packet transmission (or game) succeeds if the packet arrives to the destination, and it fails if some intermediate node discards the packet. After a specified number of games, the genetic information exchange takes place among neighbor nodes to run an evolution step. This way, we can observe the evolution of strategies according to the programmed change of environment conditions.

The mobility is considered through two mechanisms: a random selection of a set of neighbors for each node at each plasmid migration period, and a random selection of the path at each single game. In the first mechanism, at the moment of a plasmid migration period, a set of six neighbors is randomly selected for each node, and the evolution process goes on within these neighborhoods. The neighborhood size is chosen in order to model typical cellular hexagonal geometry with nodes in the vertices and the centers of each hexagon. In the second mechanism, when a packet is to be transmitted between a source and a destination, a path length is randomly chosen according to a probability mass function P_h. Given the path length h, the number of routes, r, is randomly chosen according to a conditional probability mass function $P_{r|h}$. For performance evaluation and comparison purposes, these two distributions are taken from (Seredynski et al., 2007) (see Table 4). Among the discovered routes, the source node chooses the most trusted path, which is the one with the highest product of the forwarding rates of the participating intermediate nodes, according to the observations made by the source node.

Table 4: Probability distribution of path hops (P_h) and number of discovered paths given the length ($P_{r|h}$).

	Number of hops, *h*						
	2	**3**	**4**	**5**	**6**	**7**	**8**
P_h	0.4	0.3	0.1	0.05	0.05	0.05	0.05
$P_{1\|h}$	0.5	0.5	0.6	0.6	0.6	0.8	0.8
$P_{2\|h}$	0.3	0.3	0.25	0.25	0.25	0.15	0.15
$P_{3\|h}$	0.2	0.2	0.15	0.15	0.15	0.05	0.05

For the evolution algorithm, the cross point is established at the 8^{th} bit, which marks the difference between the strategies for trusted and untrusted source nodes. If the crossover takes place, each bit of the children strategy mutates with probability 0.001. All the simulation results are presented as the average of 60 independent replicas of the given experiment.

Finally, our algorithm requires appropriate values for the number of observed decisions that a node will keep in memory, m, and for the number of games that a node will play as source in each plasmid migration period, R. In the next section, we use some simulation experiments in order to set these parameters.

4.2. Tuning the Parameters of our Algorithm

Our proposal includes the capacity of on-line adaptation. To this effect, it is important that the trust values among nodes reflect, as much as possible, the behavior of their current strategies. Therefore, taking into account all the past events (having an infinite memory) is not appropriate for our algorithm, so we use a finite memory of depth m. This memory allows us to store the previous m decisions as state variables for updating the fraction of observed forwarded packets. Regarding this memory, notice that there is a trade-off: we would like a long memory to better

evaluate the trust placed in each node, but we would like also a short memory in order not to judge the nodes based on outdated behaviors. After some pilot tests, we concluded that using the last three observed decisions leads to optimal cooperation ratios. Furthermore, the value of $m = 3$ gives four possible forwarding rates (0, 1/3, 2/3 and 1), with which we can select the four corresponding trust values of Table 1.

The second important parameter is the number of times a node plays as source in a plasmid migration period, R. Again, a long value of R would allow a better evaluation of the fitness of each strategy, but at the cost of increased convergence time and reduced adaptability. So, for the purpose of deciding an appropriate value of R, we computed the cooperation evolution for R = 10, 25, 100 and 300 packets, under different number of selfish nodes within the population of 100 nodes. Recall that the cooperation is measured as the fraction of packets originated in normal nodes that effectively reach their destinations.

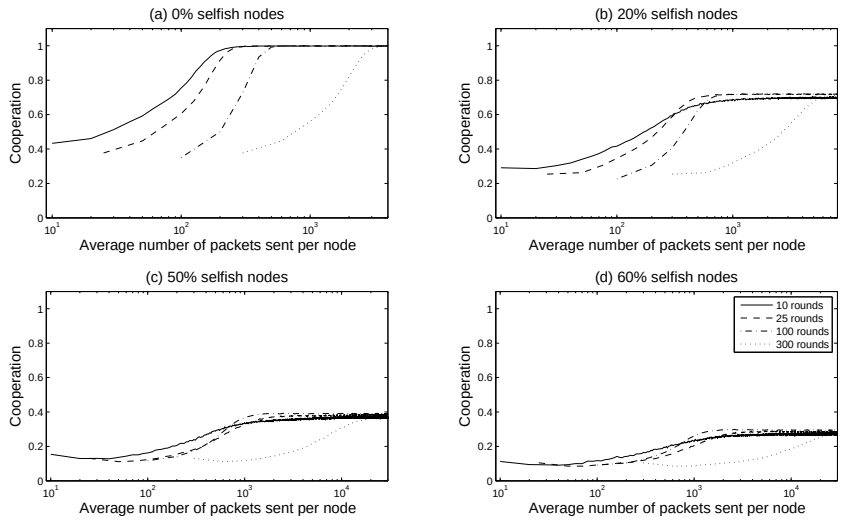

Figure 1: Evolution of cooperation for different number of selfish nodes and different number of packets per node in a plasmid migration period.

This cooperation is shown in Figure 1, where the cooperation among normal nodes is plotted against the average number of packets transmitted per node, in a logarithmic scale. Given a transmission rate, this average number of packets transmitted per node becomes a measure of time, so we are plotting both the convergence speed and the maximum achieved cooperation value. In all cases, the cooperation converges to the same steady values, but the speed of convergence varies both with R and with the fraction of selfish nodes. With no selfish nodes (Figure 1.a), a value of R as small as 10 produces fast convergence times (less than 200 packets per node) to the optimal cooperation value of 1, while this time increases to 250 packets, 600 packets and 4000 packets for R = 25, 100 and 300, respectively. With 20% of selfish nodes (Figure 1.b), R = 10 seems too small to obtain a good evaluation of the strategies. If we compare this ca se to that obtained by greater values of R, we get both a longer convergence time and a reduced maximum achieved cooperation. In this case, a value of R of 25 packets per node seems to be the best choice but, with greater fractions of selfish nodes, 100 packets per node in a plasmid migration period not

only show better convergence times but also slightly higher achieved cooperation values (Figures 1.c and 1.d). In the four environment cases, the convergence times were unacceptable when R = 300 packets. So, according to previous analysis, and taking into account that small values of R also generate greater transmission overhead due to frequent plasmid migration processes, we decided to use a PMP of $R = 100$ packets per node.

4.3. Centralized Algorithm Implementation

As we previously mentioned, we have also implemented the centralized evolution algorithm presented in (Seredynski et al., 2007). For comparison purposes and to better understand both proposals, in this section we introduce a more detailed description of this approach.

In the centralized model, nodes start with a randomly generated strategy. Then, series of tournaments are played to calculate the payoff that nodes will receive for their actions. The fitness of each player's strategy is evaluated as the average payoff per event. According to this fitness, a centralized entity, which knows all current strategies and fitnesses, selects $2N_N$ strategies through a roulette wheel mechanism, where N_N is the number of participating normal nodes, i.e., nodes that are willing to cooperate if they consider it advantageous. Applying a standard one point crossover and a standard uniform bit flip mutation over these $2N_N$ strategies, the centralized entity generates a set of N_N new strategies in order to distribute them among the normal nodes, and the whole process is repeated during a certain number of generations. The performance results of the centralized approach show that the strategies evolve according to different (but static) environments , where an environment is characterized by a given fraction of selfish nodes, i.e., nodes that always decide to discard every packet in transit. Finally, to compare both approaches, we need to consider the same network traffic. Unfortunately, the centralized model is presented in terms of rounds and tournaments instead of transmitted packets so we need to calculate the average number of packets per node per generation.

More specifically, a tournament in the centralized model is played among 50 nodes, randomly selected from a total population of 100 nodes. Each tournament is composed of 300 rounds. A round is composed of 50 (successful or failed) packet transmissions or games, one per participating node. Each of the nodes of the population must participate as source in at least two tournaments per generation. Since the probability that a given node is selected k times in n tournaments is $q(k,n) = \binom{n}{k} 2^{-n}$, the probability that a given node is selected more than once in n tournaments is $1 - q(0,n) - q(1,n) = 1 - (n+1)\,2^{-n}, n > 1$. We are interested in the probability $F(n)$ that, by the n^{th} tournament, the node that has participated in less tournaments has already been selected more than once. This is equal to the probability that any of the nodes of the population has been selected to participate in two or more of the n tournaments, i.e., $F(n) = [1 - (n+1)\,2^{-n}]^{100}$, assuming independence among nodes. Correspondingly, the probability of requiring exactly n tournaments in order for all the nodes to participate in at least two tournaments is $P(n) = F(n) - F(n-1), n > 1$. According to this distribution, the mean number of tournaments is 11.56[2]. In conclusion, in the centralized model, there are $11.56\ tournaments/generation \times 300\ rounds/tournament \times 50\ packets/round \div 100\ nodes =$ 1734 packets per generation per node, in average.

4.4. Maximum Cooperation

As we previously mention, one of the main reasons that lead us to develop a custom-made simulator with a simplified network model (about mobility, routing etc.) is to be able to compute

[2]This result we obtain theoretically has also been verified during the simulations.

the maximum theoretical cooperation, C_{max}. Next, we develop a theoretical expression for the maximum cooperation achievable under an optimal strategy with perfectly computed trust values. For this purpose, we assume that the selfish nodes are completely identified and that, with this information, the normal nodes cooperate among them and discard the packets of selfish nodes. This is the ideal condition that any trust model would like to achieve, where packets will be forwarded if they find at least one path composed exclusively of non-selfish nodes. In what follows, we will use the following notation

- N is the total number of nodes in the network
- N_N is the total number of normal nodes among the N nodes of the population
- P_h is the probability that a path has h hops (see Table 4).
- $P_{r|h}$ is the probability of finding r routes given that the path length is h hops (see Table 4).
- $B(h)$ is the event that there are no selfish nodes among h randomly selected nodes.
- A is the event that a packet finds at least one route made out exclusively of normal nodes.

According to the discussion above, a packet will reach the destination through an h-hop path if at least the first h nodes of the path are normal nodes (the destination can be a selfish node). Since the nodes of the path are randomly selected, the probability of finding a consecutive sequence of h normal nodes is given by Eq.(4)

$$Pr[B(h)] = \prod_{i=0}^{h-1} \frac{N_N - i}{N - i} \tag{4}$$

Correspondingly, the probability of having at least one selfish node in an h-hop path is $1 - Pr[Bh]$. So, the probability that each one of r routes of h hops has at least one selfish node is $(1 - Pr[Bh])^r$ and, consequently, the probability that at least one of the r routes is composed exclusively of normal nodes is $1 - (1 - Pr[Bh])^r$. Since the probability of finding r routes of h hops is $P_h \cdot P_{r|h}$, the probability that a packet finds at least one path composed exclusively of non-selfish nodes is given by Eq.(5).

$$C_{max} = P_r[A] = \sum_h \sum_r P_h P_{r|h} (1 - (1 - P_r[B(h)])^r) \tag{5}$$

As we said before, this is the ideal fraction of packets originated in normal nodes that reach their destination, i.e., this is the maximum cooperation, C_{max}.

5. Performance Evaluation

Figure 2 compares the maximum cooperation given in Eq.(5) with the maximum cooperation values obtained with the centralized and distributed algorithms as a function of the fraction of selfish nodes in a population of 100 nodes. Recall that we measure the cooperation as the fraction of packets originated by normal nodes that effectively reach their destinations. Figure 2 shows that not only the values achieved by our distributed bacterial-like algorithm are higher than values obtained by the centralized model, but also that our algorithm attains cooperation values very close to the optimal ones, especially under a low fraction of selfish nodes.

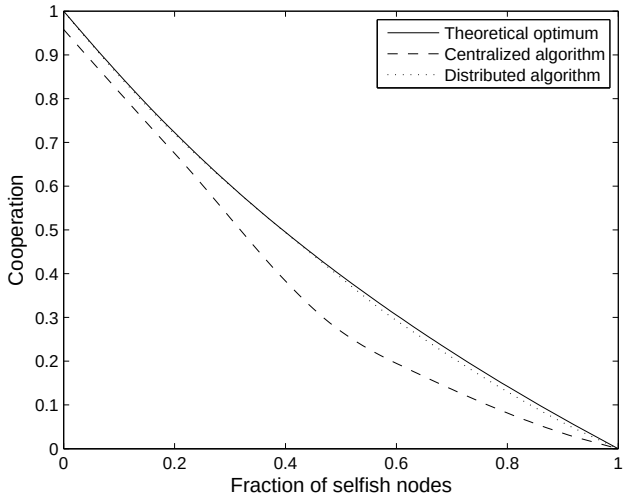

Figure 2: Maximum achievable cooperation values.

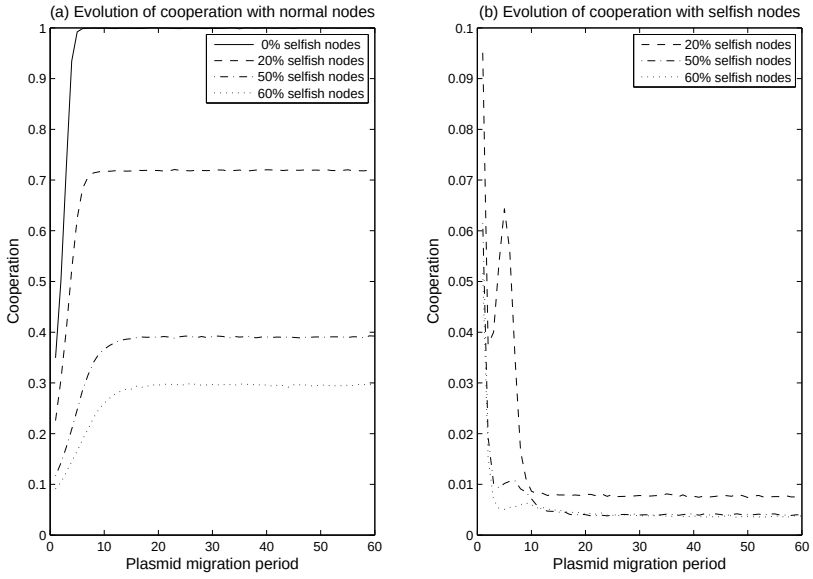

Figure 3: Evolution of Cooperation under four different environments.

Figure 3 shows the fraction of delivered packets for both normal and selfish nodes as a function of the plasmid migration periods. Figures are obtained under different percentages of selfish nodes, using the selected PMP length of $R = 100$ packets. It can be observed that our evolved strategies not only achieve a high cooperation among normal nodes but also effectively isolate

the selfish nodes by reducing their delivered packet ratio to a negligible value. These results imply the achievement of a high throughput, without a waste of energy in forwarding packets of selfish nodes (i.e. nodes save battery).

5.1. Comparison centralized vs distributed

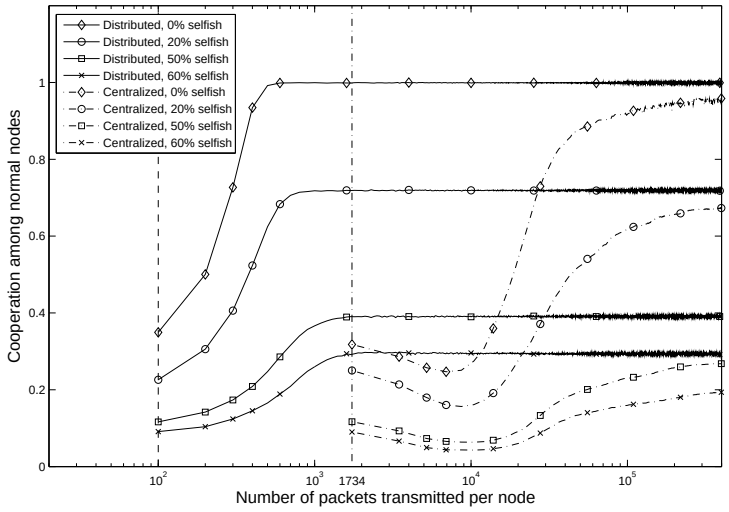

Figure 4: Comparison of centralized and distributed models.

Figure 4 shows the cooperation among normal nodes for our model and for the centralized one as a function of the average number of packets transmitted per node and the percentage of selfish nodes. Again, a PMP corresponds to $R = 100$ packets sent per node. On the other hand, a generation in the centralized algorithm corresponds to 1734 packets sent per node, as previously discussed in Section 4.3. The first observation is that our distributed algorithm converges much faster than the centralized algorithm: while the former requires between 500 and 2000 packets (from 5 to 20 PMP), depending on the environment, the later requires hundreds of thousands of packets to converge, corresponding to hundreds of generations. This is an expected result, since the distributed nature of our algorithm allows a more frequent execution of the evolution process. The second observation is that the final cooperation values achieved by our distributed algorit hm are also significantly higher than those of the centralized one. This is not surprising if we consider that each neighborhood can evolve to different solutions, so that the overlapped and mobile neighborhoods allow a more complete exploration and exploitation of the search space.

5.2. Dynamic environmental changes

Finally, we verify whether or not our distributed algorithm adapts properly to dynamic environmental changes. To do so, we change the number of selfish nodes on the fly during a single simulation by making some normal nodes to become selfish ones and vice versa.

Figure 5 shows the fraction of normal nodes and the cooperation values obtained by normal and selfish nodes as a function of time (in PMPs). In the first part of the Figure, each 50 PMPs a

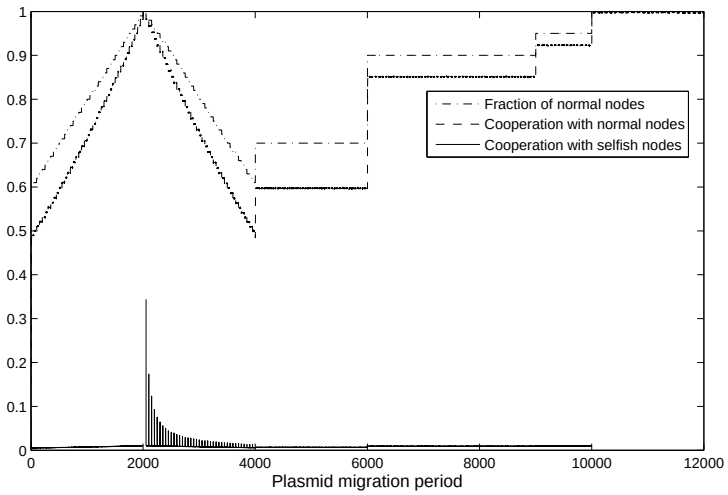

Figure 5: Adaptation of the cooperation to environmental changes.

node switches its behavior from selfish to normal. After 2000 PMPs, nodes start becoming selfish again with the same frequency (one node changes its behavior each 50 PMPs). This happens until 4000 PMPs. Then, sudden changes take place at random times. In the Figure, it can be observed that the cooperation evolved in such a way that it adapted to each new environment in just a few PMPs, keeping the network at optimal cooperation values most of the time. Notice also that nodes that became selfish were not able to keep their reputation. This means that our distributed genetic algorithm is very effective in detecting and isolating selfish nodes and, as a conclusion, selfish nodes do not cause a relevant waste of energy of cooperative nodes within the network.

In order to analyze in more detail how the adaptation takes place, we plot in Figure 6 a more detailed simulation, in which the environment changes between 60% and 100% normal nodes every 20 PMPs. At the beginning, with the initial random strategies, both normal and selfish nodes get very low cooperation from the network but, as the network runs, there is a slow adaptation towards optimal cooperation among normal nodes and almost total isolation of selfish nodes (this takes around 12 PMPs). When the selfish nodes switch their behavior to the normal one, nodes quickly learn the new cooperation conditions and quickly start cooperating. In this last case, convergence takes only 5 PMPs (compared with the 12 PMPs of the previous case). Finally, notice that when a normal node becomes selfish again, the network takes approximately 3 PMPs to detect it and isolate it.

5.3. Comparison with related work

In this section, we compare our proposal with other related works for cooperation enforcement in MANETS which are also based on non-cooperative games. To this respect, in (Seredynski and Bouvry, 2009), the authors present and analyze a centralized system similar to (Seredynski et al., 2007)[3]. In this new work, authors keep most of the system structure of (Seredynski

[3]Remember that (Seredynski et al., 2007) has been used in the article as reference model for comparison purposes.

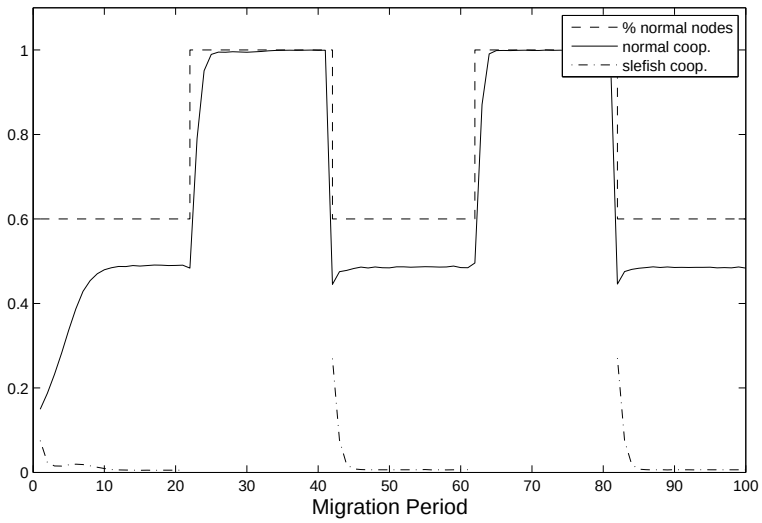

Figure 6: Evolution of cooperation under "step" environment transitions.

et al., 2007) but they change the encoding of the strategies of intermediate nodes. In particular, in the new proposal, the cooperation behavior during three consecutive time frames of different length are included in the codification of the strategy. It is worth to mention that this new work only studies static scenarios (environmental changes are not considered) and that authors assume a high interaction between nodes, increasing the number of rounds with respect to their previous work. In addition, the analysis presented in (Seredynski and Bouvry, 2009) is not focused in measuring the level of cooperation achieved, but it is focused in analyzing which are the most popular strategies after several generations. The main conclusion of their work is that the popular strategies can be seen as variations of tit-for-tat but that none of them resulted evolutionarily stable in their system. In this sense, another variation of tit-for-tat called generous tit-for-tat (GTFT) has also been proposed to be used in MANET (Milan et al., 2006). This strategy partly alleviates the impact of environmental changes by assuming that the nodes may be generous to contribute more to the network than to benefit from it. However, as pointed in (Ji et al., 2010), it is difficult to extend GTFT to a multi-player game scenario like end-to-end transmissions in multi-hop multi-node MANETs. In our proposal, the overall behavior of the network is optimized by considering the previous forwarding actions of the source node and the network within the strategy code. Then, these strategies evolve genetically to achieve an optimal behaviour.

On the other hand, a very close related work is presented in (Komathy and Narayanasamy, 2008). This work uses non-cooperative game theory with a distributed evolution based on BNS (Best Neighbor Strategy)(Komathy and Narayanasamy, 2007), in which a node decides the strategy to follow by changing to other player's strategy if deemed worthy. The BNS proposal distributes the evolution process among the nodes, but the payoff structure strictly encourages cooperating strategies without taking into account the energy savings of discarding strategies. Despite the distributed nature of this evolution strategy, our proposal is different from (Komathy and Narayanasamy, 2008) in three aspects. First, in our proposal the payoffs not only reward

cooperation but also reward the energy savings of discarding strategies. This payoff structure models better the natural dilemma a node faces, and also adds significance to the emerging cooperation, because discarding strategies can also give a good payoff. Second, in our proposal each node's strategy depends on the trust it has on the source of the packet to be forwarded, which, in turn, depends on the observed behavior history, not only on the outcomes of the random pairing games. And third, an important strategy criterion for a node is how useful has been the network in relying its own packets to the intended destinations, which better models the motivation for cooperation.

In (Wang et al., 2010), the authors propose a non-cooperative game based on a repeated forwarding game. They also define a strategy space with five possible strategies: defection, cooperation, tit-for-tat, anti-tit-for-tat and random. They divide the time in series of discrete slots, and they propose a system with a single memory position in which the strategy can be changed at each slot according to the fitness of the previous slot. They also propose a punishment mechanism, in which cooperating nodes change their forwarding strategy to a more restrictive one when they notice that their neighbors are doing the same thing. They argue that this kind of punishment mechanism can make other nodes around the selfish node to decrease their forwarding strategies and in this way achieve a global punishment. Compared with our results, they only show results for a network with a single selfish node that suddenly reduces its forwarding behavior and they do not consider the impact of selfishness in the performance of cooperative nodes.

In (Ji et al., 2010) the authors present an interesting non-cooperative game theoretic framework. The main contribution of this work is that it considers noisy and imperfect observation. To tackle this problem, the authors propose to use a belief system using the Bayes' rule to assign and update a belief (trust) in other nodes. In particular, this belief measure is related to the probability that a certain node will follow a certain cooperation strategy. The authors develop a model for the interaction of two nodes and then they try to generalize this model to a multi-node scenario. As the authors state, a direct design of their system for the multi-node case is difficult. For this reason, the two-player scenario is used as baseline and a strategy called BMPF (Belief-based multi-hop packet forwarding) is used to try to improve the end-to-end performance. To properly apply the BMPF strategy, the sender needs an updated belief value for each node on the possible route, which means that a lot of interaction between nodes are needed to take advantage of the BMPF strategy. On the other hand, their belief value accumulates all the previous history. Despite they use a discount factor, the accumulation of a long history makes quick adaptability difficult. To this respect, notice that the behavior of a device can switch to a selfish behavior because, for example, the node is running out of battery. When there are changes of this type, it is very important to consider the convergence time required for adaptation to the new conditions. In the mentioned work, there are not results in this direction, but their results are presented in average. It is true that their model can partially alleviate changes of behavior by considering them a kind of imperfect observation but their convergence speed is going to be low in comparison with ours since they keep long records of interaction history. In our proposal, we have explicitly analyzed this type of environmental changes and we have shown that in our system the nodes quickly learn the appropriate cooperation behavior (see our results in Section 5.2).

Finally, in (Jaramillo and Srikant, 2010), the authors present a system called Darwin which is also based on a non-cooperative game. The strategy proposed in Darwin considers a retaliation situation after a node has been falsely perceived as selfish. The system assumes that nodes share their perceived dropping probability with each other. This assumption is made in order to facilitate the theoretical analysis by isolating a pair of nodes, but in a real implementation, a mechanism is required to guarantee that even if a node lies, the reputation scheme still works.

Authors state that the study of this problem in the context of wireless networks is not solved, and that they just propose a simple mechanism which relies on other cooperative nodes to tell the actual perceived dropping probability of a node to minimize the impact of liars. On the contrary, in our proposal, nodes do not share trust values. Nodes just exchange their strategies and their associated fitness. The strategies do not contain specific data of particular nodes and a strategy is used several times only if it provides a good fitness, which can be directly measured by each node. Regarding performance results, the authors of Darwin do not show adaptability measures, but they mention that their system needs to interact for a long time in order to estimate the discarding probability of neighbor nodes. Simulations of Darwin show that the nodes that implement it effectively obtain more cooperation than those that not implement Darwin. Authors define non-Darwin nodes as those that apply random discard with some probability. Although their results are not directly comparable with ours, it is interesting to notice that unlike our system, even when the non-Darwin nodes discard with probability one (which is the definition of selfish nodes in our work), they still receive a considerable cooperation from the network (around 70%).

6. Conclusions

In this article we have shown that it is possible to use distributed algorithms for the genetic evolution of strategies in a game theoretic trust model for MANETs. We have proposed a trust model in which genetic information is exchanged among neighbor nodes, much like plasmid migration in bacterial colonies. This way we introduce low communication overhead, while disseminate good strategies throughout the network by means of neighborhood overlapping and mobility. Our proposal does not need a central entity and does not require unrealistically large number of node interactions to evaluate the fitness. Instead of this, each node adapts its strategy to the dynamical characteristics of the network, trying to maximize its payoff in terms of packet delivery and resource saving. As an emerging feature of this local optimization procedure, the whole network was able to maximize the cooperation and save energy by offering the forwarding service only to those nodes that were willing to cooperate with the network, and isolating those free-rider nodes that wanted to utilize the resources of the network without cooperating with it. Our distributed model not only achieved these objectives hundreds of times faster than the centralized model, but also obtained better cooperation values, very close to those predicted by a theoretical upper bound. Thanks to these features of fast convergence to optimal cooperation values and selfish node isolation, the algorithm also demonstrated a remarkable adaptation capacity to dynamic environmental changes.

7. References

Alba, E., Dorronsoro, B., 6 2008. Cellular Genetic Algorithms (Operations Research/Computer Science Interfaces Series), 1st Edition. Springer.

Alba, E., Troya, J. M., 1999. A survey of parallel distributed genetic algorithms. Complexity 4 (4), 31–52.

Anderegg, L., Eidenbenz, S., 2003. Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In: MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking. ACM, New York, NY, USA, pp. 245–259.

Axelrod, R., Dion, D., December 1988. The further evolution of cooperation. Science 242 (4884), 1385–1390.

Bansal, S., Baker, M., 2003. Observation-based cooperation enforcement in ad hoc networks. Tech. rep. URL `citeseer.ist.psu.edu/article/bansal03observationbased.html`

Baras, J., Jiang, T., 2005. Cooperation, Trust and Games in Wireless Networks. Springerlink, Ch. 4, pp. 183–202.

Buchegger, S., Boudec, J.-Y. L., 2004. A robust reputation system for p2p and mobile ad-hoc networks. In: In Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems.

Buchegger, S., Le Boudec, J.-Y., 2002. Performance analysis of the confidant protocol. In: MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing. ACM Press, New York, NY, USA, pp. 226–236.

Buttyán, L., Hubaux, J.-P., 2000. Enforcing service availability in mobile ad-hoc wans. In: MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing. IEEE Press, Piscataway, NJ, USA, pp. 87–96.

Cabrita, C., Botzheim, J., Ruano, A., Kczy, L., 2003. Genetic programming and bacterial algorithm for neural networks and fuzzy systems design. IFAC International Conference on Intelligent control Systems and Signal Processing (ICONS).

Cantupaz, E., 1998. A survey of parallel genetic algorithms. Calculateurs Paralleles, Réseaux et Systémes Répartis 10 (2), 141–171.

Dale, J., Park, S., 3 2004. Molecular Genetics of Bacteria, 4th Edition. Wiley.

Felegyhazi, M., Hubaux, J.-P., Buttyan, L., May 2006. Nash equilibria of packet forwarding strategies in wireless ad hoc networks. IEEE Transactions on Mobile Computing 5 (5), 463–476.

Harvey, I., 1996. The microbial genetic algorithm.
URL `http://www.cogs.susx.ac.uk/users/inmanh/Microbial.pdf`

Holland, J. H., 1975. Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor.

Ishibuchi, H., Namikawa, N., Sept. 2005. Evolution of cooperative behavior in the iterated prisoner's dilemma under random pairing in game playing. IEEE Congress on Evolutionary Computation 3, 2637–2644.

Janzadeh, H., Fayazbakhsh, K., Dehghan, M., Fallah, M. S., 2009. A secure credit-based cooperation stimulating mechanism for manets using hash chains. Future Gener. Comput. Syst. 25 (8), 926–934.

Jaramillo, J. J., Srikant, R., 2010. A game theory based reputation mechanism to incentivize cooperation in wireless ad hoc networks. Ad Hoc Networks 8 (4), 416–429.

Ji, Z., Yu, W., Liu, K. R., 2010. A belief evaluation framework in autonomous manets under noisy and imperfect observation: Vulnerability analysis and cooperation enforcement. IEEE Transactions on Mobile Computing 9, 1242–1254.

Komathy, K., Narayanasamy, P., 2007. Best neighbor strategy to enforce cooperation among selfish nodes in wireless ad hoc network. Computer Communications 30 (18), 3721–3735.

Komathy, K., Narayanasamy, P., 2008. Trust-based evolutionary game model assisting aodv routing against selfishness. Journal of Network and Computer Applications 31 (4), 446–471.

Li, F., Wu, J., 2010. Uncertainty modeling and reduction in manets. IEEE Transactions on Mobile Computing 9 (7), 1035–1048.

Luo, J., Liu, X., Fan, M., 2009. A trust model based on fuzzy recommendation for mobile ad-hoc networks. Computer Networks 53 (14), 2396–2407.

Marias, G. F., Georgiadis, P., Flitzanis, D., Mandalas, K., 2006. Cooperation enforcement schemes for manets: a survey: Research articles. Wireless Communications and Mobile Computing 6 (3), 319–332.

Marshall, I. W., Roadknight, C., 2000. Adaptive management of an active service network. BT Technology Journal 18 (4), 78–84.

Marti, S., Garcia-Molina, H., 2006. Taxonomy of trust: categorizing p2p reputation systems. Computer Networks 50 (4), 472–484.

Mejia, M., Alzate, M., Muoz, J. L., Pea, N., Esparza, O., 09 2009a. Distributed evolution of strategies in a game theoretic trust model for mobile ad hoc networks. In: VII Jornadas de Ingeniería Telemática - JITEL 2009.

Mejia, M., Peña, N., Muñoz, J. L., Esparza, O., 2009b. A review of trust modeling in ad hoc networks. Internet Research volume 19 (1), 88–104.

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

Milan, F., Jaramillo, J. J., Srikant, R., 2006. Achieving cooperation in multihop wireless networks of selfish nodes. In: GameNets '06: Proceeding from the 2006 workshop on Game theory for communications and networks. ACM, New York, NY, USA, p. 3.

Mu noz, J. L., Esparza, O., Aguilar, M., Carrascal, V., Forné, J., 2010. Rdsr-v. reliable dynamic source routing for video-streaming over mobile ad hoc networks. Computers Networks 54 (1), 79–96.

Nawa, N. E., Furuhashi, T., Hashiyama, T., Uchikawa, Y., December 1999. A study on the discovery of relevant fuzzy rules using pseudo-bacterial genetic algorithm. IEEE Transactions on Industrial Electronics 46 (6), 1080 – 1089.

Nowostawski, M., Poli, R., 1999. Parallel genetic algorithm taxonomy. In: Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. pp. 88–92.

NS2, 2009. Network simulator. http://www.isi.edu/nsnam/ns/.

Ochman, H., Lawrence, J. G., Groisman, E. A., May 2000. Lateral gene transfer and the nature of bacterial innovation. Nature 405 (6784), 299–304.
OPNET, 2009. Opnet technologies inc. http://www.opnet.com.
Osborne, M., 2004. an Introduction to Game Theory. OXFORD University Press.
Perkins, C. E., 1 2001. Ad Hoc Networking. Addison-Wesley Professional.
QUALNET, 2009. Scalable network technologies. http://www.scalable-networks.com/products/developer.php.
Saad, W., Han, Z., Debbah, M., Hjørungnes, A., 2009. A distributed coalition formation framework for fair user cooperation in wireless networks. Transactions on Wireless Communications 8 (9), 4580–4593.
Safa, H., Artail, H., Tabet, D., 2010. A cluster-based trust-aware routing protocol for mobile ad hoc networks. Wirel. Netw. 16 (4), 969–984.
Seredynski, M., Bouvry, P., 2009. Evolutionary game theoretical analysis of reputation-based packet forwarding in civilian mobile ad hoc networks. In: IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing. IEEE Computer Society, Washington, DC, USA, pp. 1–8.
Seredynski, M., Bouvry, P., Klopotek, M., April 2007. Modelling the evolution of cooperative behavior in ad hoc networks using a game based model. IEEE Symposium on Computational Intelligence and Games CIG., 96–103.
Sherwood, R., Lee, S., Bhattacharjee, B., 2006. Cooperative peer groups in nice. Computer Networks 50 (4), 523–544.
Sun, Y. L., Yu, W., Han, Z., Liu, K., Feb. 2006. Information theoretic framework of trust modeling and evaluation for ad hoc networks. IEEE Journal on Selected Areas in Communications 24 (2), 305–317.
Wang, K., Wu, M., Ding, C., Lu, W., 2010. Game-based modeling of node cooperation in ad hoc networks. In: 19th Annual Wireless and Optical Communications Conference (WOCC).
Weiss, M. A., 1998. Data Structures and Algorithm Analysis in Java. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
Whitley, D., 1993. A genetic algorithm tutorial. Tech. rep., Colorado State University.
Wrona, K., Mähönen, P., 2004. Analytical model of cooperation in ad hoc networks. Telecommunication Systems Volume 27, 347 – 369.
Yale, S. Z., Zhong, S., 2002. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In: in Proceedings of IEEE INFOCOM. pp. 1987–1997.
Zouridaki, C., Mark, B. L., Hejmo, M., Thomas, R. K., 2009. E-hermes: A robust cooperative trust establishment scheme for mobile ad hoc networks. Ad Hoc Netw. 7 (6), 1156–1168.