Distributed Evolution of Strategies in a Game Theoretic Trust Model for Mobile Ad Hoc Networks

Marcela Mejia[1], Marco Alzate[2], Jose L. Muñoz[3], Néstor Peña[1] and Oscar Esparza[3]

[1]Universidad de los Andes
[2]Universidad Distrital
[3]Universidad Politécnica de Cataluña

am.mejia75@uniandes.edu.co, malzate@udistrital.edu.co,
jose.munoz@entel.upc.edu, npena@uniandes.edu.co, oscar.esparza@entel.upc.edu

***Abstract*—Cooperation among nodes is fundamental for the operation of mobile ad hoc networks (MANETs). However, in these networks there could be selfish nodes that use resources from other nodes to send their packets but do not offer their resources to forward packets for other nodes. Several trust models have been proposed as mechanisms to incentive cooperation in MANETs. 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 so they cannot properly adapt to fast changing conditions. In this paper, we propose a game theoretic trust model that uses a bacterial-like algorithm to let the nodes quickly learn the appropriate cooperation behavior. Our model is completely distributed and achieves good cooperation values in a small fraction of the time compared with centralized algorithms.**

***Keyword*—MANET, Trust models, Game theory, evolutionary algorithm**

I. 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 [1]. MANETs are particularly vulnerable to selfish behavior, as some nodes may prefer saving resources instead of forwarding packets on behalf of others [2]. Thus, it is important to encourage the nodes to participate in essential network functions such as packet routing and forwarding. In this sense, several trust models have been proposed as mechanisms to incentive node participation within the network [3]. A trust model is a conceptual abstraction to build mechanisms for assigning, updating and using trust levels between the entities in a distributed system [4]. The trust model becomes a tool for helping the subject of a distributed system to select the most reliable agent among several others offering a service [4]. Many different mechanisms have been used in the literature to design trust models for mobile ad hoc networks [3]. The problem of deciding whether to cooperate (and improve the trust) or not to cooperate (and save resources) can be seen as a game. For this reason, many of the proposals in the literature are based on game theory concepts [2], [5], [6]. Among the proposals in the literature, the trust model presented by Seredynski et. al. in [7] is interesting because it presents a way of dynamically adapting the collaboration strategy to the network conditions. The evolution is performed using a genetic algorithm that presents promising results regarding cooperation improvement. However, there are still serious concerns about the highly centralized nature of the approach and its slow convergence. Indeed, the optimal strategies are obtained by using a centralized entity that runs a conventional genetic algorithm after a large number of interactions between nodes to evolve the set of strategies. Taking into account the drawbacks of the previous proposal, in this paper we present a distributed bacterial-like evolution algorithm based on a few interactions among nodes. Our model does not assume a central entity nor requires an unrealistically large number of interactions among nodes to evolve the strategies. It is based on distributed parallel cellular genetic algorithms [8],[9],[10],[11],[12] where genetic information is interchanged among neighbor nodes (much like plasmid migration in bacterial colonies [13],[14]). This way, each individual node selects the strategy that locally maximizes its payoff in terms of packet delivery and resource saving. This local payoff maximization is such that globally the whole network increases the cooperation (and, consequently, the throughput) and reduces the resources wasted serving selfish nodes. The rest of the paper is organized as follows. Section II briefly describes the trust model of [7], because we base our proposal on it. Section III introduces our trust model and its evolutionary algorithm. Section IV shows some numerical results and compares them with the previously published results of [7] and with a theoretical upper bound on the maximum achievable cooperation. Section V concludes the paper.

II. Centralized Evolution Algorithm

In [7], the interactions among nodes are based on the iterated prisoner's dilemma under the random pairing game [15]. 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 source node and the trust level that the intermediate

node has in the source node. The model is comprised of a trust evaluation mechanism, a game based network model, a strategy and a genetic algorithm to evolve the strategy.

A. 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, pcs_A, and the number of packets that A has actually forwarded, pcf_A. So B can compute the forwarding rate of A $f_r(B, A) = pcf_A/pcs_A$. With this rate, B can determine the trust level it should have in A, $T\{B : A\}$, as shown in Table I.

Tabla I
RELATION BETWEEN DELIVERY RATE AND TRUST LEVEL

$f_r(B, A)$	$T\{B : A\}$
$1 - 0.9$	3
$0.9 - 0.6$	2
$0.6 - 0.3$	1
$0.3 - 0$	0

B. Game-based network model

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. In this model, two types of nodes are defined: source nodes and intermediate nodes. Therefore, two types of payoff tables are maintained, as shown in Table II.

Tabla II
PAYOFF TABLES

Source Node	
Transmission Status	Payoff
Successful	5
Failed	0

Intermediate Node				
	Trust Level			
Decision	$T = 3$	$T = 2$	$T = 1$	$T = 0$
Cooperate	3	2	1	0.5
Discard	0.5	1	2	3

C. The strategy

The strategy that a node has to follow when it is acting as intermediate node is represented by a string of bits. Each bit represents a decision (Discard (0)/ Cooperate (1)) taking into account a set of parameters. In [7], this set of parameters is formed by: (i) the transmission status of the two previous games that the node has played as source (which could be success (S) or failure (F)) and, (ii) the trust level that the node has in the source node. The resulting strategy has 18 bits and an example is shown in TableIII. Notice that two additional bits are used when the node has played less than two games as source node.

Tabla III
STRATEGY CODING, EXAMPLE STRATEGY 0000 0011 0101 1010 11

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

D. The genetic algorithm

The genetic algorithm aims to maximize the mean payoff of each node. Initially, nodes have a randomly generated strategy and, 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, $2N$ strategies are selected through a roulette wheel mechanism, where N is the number of participating normal nodes. Applying a standard one point crossover and a standard uniform bit flip mutation over these $2N$ strategies, a set of N new strategies is generated and the whole process is repeated during a certain number of generations. The results of the simulations in [7] show that the strategies evolve to adapt to different environments, where an environment is characterized by a given number of selfish and normal nodes. Finally, we clarify some definitions used in [7]. A "tournament" is played among 50 nodes, randomly selected from a total population of 100 nodes, where each tournament is composed of 300 "rounds". A round is composed of 50 (successful or failed) packet transmissions or "games". Since each of the 100 nodes must participate in at least two tournaments per generation, the average number of tournaments per generation is 11.56, for a total of $11.56 \times 300 \times 50 \div 100 = 1734$ packets per node per generation, in average. This number will be important to compare the efficiency of this centralized algorithm with our distributed algorithm.

III. DISTRIBUTED BACTERIAL-LIKE EVOLUTION ALGORITHM

The proposal in [7] fairly represents the dilemma of forwarding packets to gain trust or discard them to save energy in a MANET. However, it has an expensive fitness evaluation mechanism and a highly centralized evolution algorithm, since, after each tournament, a central entity should collect all the strategies and their firtness in order to compute and redistribute the new evolved strategies. This makes the model unfeasible for implementation in a real MANET. For this reason, we propose and evaluate a distributed model based in [7], in which we keep the trust evaluation mechanism, the game based network model and the strategy but change the genetic algorithm to evolve the strategy. Indeed, we omit both the central entity and the costly process of having multiple generations by allowing the nodes to exchange genetic material among their neighbors, like plasmid migration in bacterial colonies [13]. A plasmid is an extrachromosomal DNA molecule that bacteria can take up from the external environment, in order to obtain a gene that gives the cell a

selective advantage. When the cell replicates, it makes copies of the acquired plasmid [14]. This model can be used in evolutionary algorithms instead of the traditional Darwinist method used in [7]. The plasmid migration is a greedy algorithm that, at each step, makes 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. For us, the foremost characteristics of the algorithm are its distributed implementation and its convergence speed. These features make the algorithm readily implementable in a MANET environment.

In our algorithm, we evolve the strategies on-line during the life of the network instead of using a centralized entity to run the genetic algorithm for each generation, and different tournaments to evaluate the strategies at each generation. So, we keep the concepts of "game" and "round" of [7] but omit those of "tournament" and "generation". A game is a successful or failed packet transmission for which a source node selects the most trusted h-hop route among r possible routes. The number of hops, h, obeys a probability distribution p_h and, given h, and the number of routes, r, obeys a conditional probability distribution $p_{r|h}$. These distributions will be used below to compute the maximum achievable cooperation. A round is a set of 100 games in which each node plays once a game as source. The evolution goes through periods of R rounds, called *Plasmid Migration Periods* (PMP), after which every node interchanges genetic information with its neighbors. Figure1 shows how our distributed model works. Each node starts with a random strategy, whose fitness is evaluated during a PMP of R rounds. At the j^{th} PMP, node i interchanges its strategy $s_i(j)$ and its fitness $f_i(j)$ with its one-hop neighbors. Each node selects a potential parent strategy among the neighbor strategies through the roulette wheel process, and then:

- If the current fitness $f_i(j)$ of the node is better than the selected one $f_N(j)$, the node keeps its own strategy by skipping the crossover process.
- Otherwise, the selected strategy is combined with the current strategy of the node using a one point crossover.

Finally, the resultant strategy suffers a standard uniform bit flip mutation process before going to a new PMP for evaluation. The process is repeated during the life time of the network. If we choose the number of rounds in a PMP as R = 1734, a PMP in our distributed algorithm would be equivalent to a generation in the centralized algorithm of [7]. However, our algorithm converges much faster, which allows us to use lower values for R and, at the same time, have a convergence in a PMP similar to the convergence achieved in a generation in [7]. This means that our algorithm will be close to $1734/R$ times faster than the centralized one. In the next section we perform some experiments to determine R.

IV. Numerical Results

In this section we show some evaluation results to demonstrate the usability of our proposal for the MANET scenario. As in [7], all the simulations have been independently replicated 60 times. One of the first experiments was aimed at deciding R (i.e. the length of a PMP). For this purpose, we

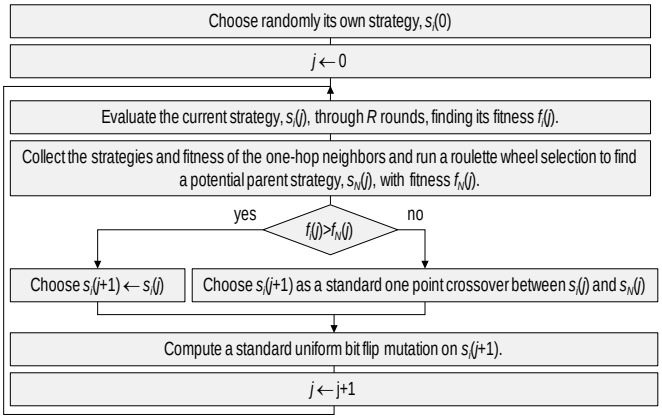

Fig. 1. On-Line distributed evolutionary algorithm

computed the cooperation evolution for R = 25, 75 and 300 rounds under different number of selfish nodes within the population of 100 nodes. The results obtained (not included here due to space constraints) show that in all the cases the cooperation converged to steady values after a few hundred plasmid migration periods. The cooperation achieved with 75 and 300 rounds per migration period was very similar in every scenario, although the convergence was faster with 75 rounds per PMP than with 300. With 25 rounds per PMP, not only the convergence took longer, but also converged to lower cooperation values. Because of these results, we decided to use a PMP of R = 75 rounds. Figure 2 plots the average cooperation as a function of the number of packets generated per node, comparing the evolution speed of our distributed algorithm and the centralized algorithm of [7] under different fraction of selfish nodes. We compare the first 250 generations of the centralized algorithm, corresponding to an average of 433500 packets generated per node (in grey color), and the first 2500 PMPs of our distributed algorithm, corresponding to 187500 packets generated per node (in black color). The continuous lines correspond to a scenario with no selfish nodes, the dashed lines correspond to a scenario where 20% of the nodes are selfish, the dashed-dotted line is for 50% of selfish nodes and the dotted line is for 60% of selfish nodes. The centralized algorithm starts at 1734 packets transmitted per node because that corresponds to the first generation. The distributed algorithm starts at 75 packets, corresponding to the first PMP.

The figure2 shows that our algorithm converges much faster than the centralized one. The improvement achieved over the convergence speed is above an order of magnitude, although the best cooperation achieved is lower for our distributed algorithm than for the centralized one when there are selfish nodes within the network. This behavior can be explained because (1) the parallel evolution only considers the genetic information of local individuals instead of the global information of the whole population, (2) the fitness is not as thoroughly evaluated with 75 transmitted packets as with 1734, and, fundamentally, (3) the nodes carry the reputation of the strategies they used before the previous PMP.

In order to compare the maximum achieved cooperation of the centralized and distributed algorithms, below we develop a theoretical expression of the maximum achievable cooperation

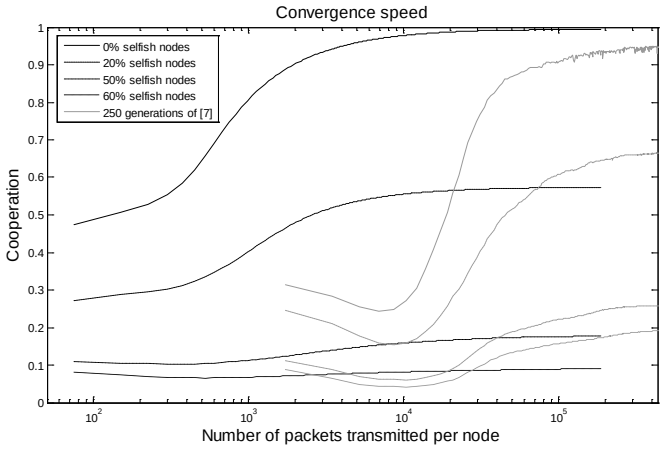

Fig. 2. Maximum achievable cooperation values

under an optimal strategy with perfectly computed trusts. 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. In this condition, a packet will get through whenever the path is composed exclusively of normal nodes, which occur with the probability shown in Equation (1), where P_h is the probability that the path length is h hops, $P_{r|h}$ is the probability of finding r routes given that the path length is h hops, A is the event *a packet finds at least one route made out exclusively of normal nodes* and B(h) is the event *there are no selfish nodes among h randomly selected nodes*. Clearly, the ideal condition implies that the best possible cooperation, C_{best}, is the probability of A. The probability of $B(h)$ is given in Equation (2), where N_N is the number of normal (not selfish) nodes and N is the total number of nodes.

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

$$P_r[B(h)] = \prod_{i=0}^{h-1} \frac{N_N - i}{N - i} \quad (2)$$

Figure3 compares the theoretically maximum achievable cooperation with the maximum values obtained through the centralized and distributed algorithms. Except when there are no selfish nodes, our best values are lower than those of the centralized algorithm and even farther from the optimal ones. Nevertheless, our values, which are obtained on-line in a distributed way, are close to those obtained in a centralized way.

There is a tradeoff between optimality and adaptability in terms of the length of the PMP, since it will determine the accuracy of the fitness evaluation. In [7], since each node generates an average of 1734 packets per generation before going through the genetic evolution algorithm, there is a highly accurate estimation of the fitness at the cost of a prohibitively large convergence time. Besides, in [7] the central entity knows the strategies of all the nodes and their fitness, so this central entity knows the entire space of feasible strategies to compute the next generation and distribute it over all the nodes. In our approach, we accept a higher variance in the evaluation of the fitness by reducing the PMP to only a few node interactions. This allows us both to replace the central entity by a distributed plasmid migration based on locally interchanged information over one-hop neighbors, and to run this plasmid migration often enough for the nodes to converge on line to good strategies. This makes our scheme implementable in a real MANET, because it exploits its distributed organization and takes advantage of the clustered nature of its topology.

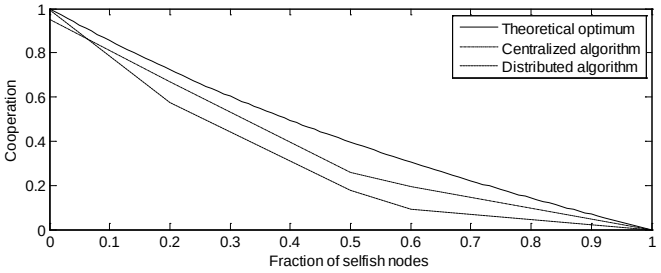

Fig. 3. Maximum achievable cooperation values

V. Conclusions

In this paper 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 nodes interchange genetic information among neighbor nodes, much like plasmid migration in bacterial colonies. This way, each node adapts its strategy to the dynamical characteristics of the network, maximizing its payoff in terms of packet delivery and resource saving. Our proposal does not need a central entity and does not require unrealistically large number of node interactions to evaluate the fitness. The numerical results show that our algorithm can quickly find good strategies, more than 20 times faster than the corresponding centralized algorithm. Further work will modify the trust computation to reflect the change of strategies at each PMP and will evaluate the adaptability to changing conditions.

Acknowledgements

This work was partly supported by the Colombian Institute for science and Technology development, Colciencias, Universidad Nueva Granada and Universidad de los Andes, in Colombia, as well as the Spanish Government through projects CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", TSI2007-65393-C02-02 "ITACA" and TSI2005-07293-C02-01 "SECONNET", and by the Government of Catalonia under grant 2005 SGR 01015 to consolidated research groups.

References

[1] Charles E. Perkins. *Ad Hoc Networking*. Addison-Wesley Professional, 1 2001.

[2] Konrad Wrona and Petri Mähönen. Analytical model of cooperation in ad hoc networks. *Telecommunication Systems*, Volume 27:347 – 369, 2004.

[3] Marcela Mejia, Néstor Peña, José. L. Muñoz, and Oscar. Esparza. A review of trust modeling in ad hoc networks. *Internet Research*, volume 19-1(1):88–104, 2009.

[4] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: categorizing p2p reputation systems. *Comput. Netw.*, 50(4):472–484, 2006.

[5] Juan José Jaramillo and R. Srikant. Darwin: distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 87–98, New York, NY, USA, 2007. ACM.

[6] Lu Yan and Stephen Hailes. Cooperative packet relaying model for wireless ad hoc networks. In *FOWANC '08: Proceeding of the 1st ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing*, pages 93–100, New York, NY, USA, 2008. ACM.

[7] M. Seredynski, P. Bouvry, and M.A. Klopotek. Modelling the evolution of cooperative behavior in ad hoc networks using a game based model. *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 96–103, April 2007.

[8] Enrique Alba and BernabÃ© Dorronsoro. *Cellular Genetic Algorithms (Operations Research/Computer Science Interfaces Series)*. Springer, 1 edition, 6 2008.

[9] Enrique Alba and José M. Troya. A survey of parallel distributed genetic algorithms. *Complex.*, 4(4):31–52, 1999.

[10] Erick Cantpaz. A survey of parallel genetic algorithms. *Calculateurs Paralleles*, 10, 1998.

[11] M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, pages 88–92, Dec 1999.

[12] Enrique Alba and Bernabé Dorronsoro. Auto-adaptación en algoritmos evolutivos celulares, un nuevo enfoque algoritmico. In *II Congreso Español sobre Metahurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB03)*, 2003.

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

[14] Jeremy Dale and Simon Park. *Molecular Genetics of Bacteria*. Wiley, 4 edition, 3 2004.

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