On Cheating in CSMA/CA Ad Hoc Networks*

Mario Čagalj
LCA-EPFL
CH-1015 Lausanne
Switzerland
mario.cagalj@epfl.ch

Saurabh Ganeriwal
NESL-UCLA EE
Los Angeles, CA
saurabh@ee.ucla.edu

Imad Aad
LCA-EPFL
CH-1015 Lausanne
Switzerland
imad.aad@epfl.ch

Jean-Pierre Hubaux
LCA-EPFL
CH-1015 Lausanne
Switzerland
jean-pierre.hubaux@epfl.ch

ABSTRACT

CSMA/CA protocols rely on the random deferment of packet transmissions. Like most other protocols, CSMA/CA was designed with the assumption that the nodes would play by the rules. This is important, since the nodes themselves control their random deferment. However, with the higher programmability of the network adapters, the temptation to tamper with the software or firmware is likely to grow; in this way, a user could obtain a much larger share of the available bandwidth at the expense of other users.

We use a game-theoretic approach to investigate the problem of the selfish behavior of nodes, specifically geared towards the most widely accepted protocol in this class of protocols, IEEE 802.11. We show that a selfish and non-cooperative behavior by a small population of (two or more) cheaters leads to a network collapse. We argue that this provides an incentive for cheaters to cooperate with each other. However, explicit cooperation among nodes is clearly impractical. By applying the model of dynamic games borrowed from game theory, we derive the conditions for the stable and optimal functioning of a population of cheaters. We use this insight to develop a simple, localized and distributed cheating protocol that needs no explicit cooperation between cheaters or *a priori* knowledge about the total number of nodes/cheaters in the system. We show that the scheme successfully guides multiple selfish nodes to a Pareto-optimal Nash equilibrium.

1. INTRODUCTION

Carrier sense multiple access with collision avoidance (CSMA/CA) protocols rely on the random deferment of packet transmissions for the efficient use of the shared wireless channel among many nodes in a network; in spite of its shortcomings, this class of MAC protocols is one of the most popular for ad hoc networks.

It is in general assumed that all nodes respect the rules of the protocol. However, we claim that this assumption is less and less legitimate, because the network adapters are becoming more and more *programmable*. As a result, a user can today very easily modify the behavior of its wireless interface.

In this paper, we study wireless ad hoc networks containing one or several *selfish* users. By "selfish" we designate the users who are ready to tamper with their wireless interface in order to increase their own share of the common transmission resource; we assume these users to be rational, and not malicious (hence they are ready to harm other users only if they can derive a benefit from this misbehavior).

More specifically, we consider that a cheater makes use of the easiest (and yet highly rewarding) cheating technique, namely it does deliberately not respect the random deferment of the transmission of its own packets (see Figure 1).

Although this cheating technique is straightforward, we show that studying its implications is far from trivial. In order to be able to corroborate our simulations with analytical results, we make use of game theory: each node[1] is a player, the throughput it enjoys is its payoff, and the size of its contention window represents its move.

By making use of this model and of extensive simulations, we study several problems in a systematic way. First,

*The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. (`http://www.terminodes.org`)

EPFL-IC Technical Report No. IC/2004/27

[1]In the rest of the paper, we do not distinguish between the user and his or her node.

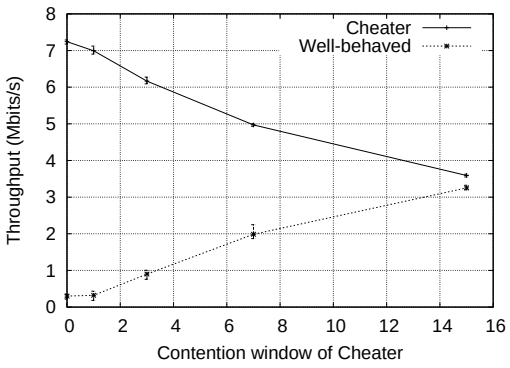


Figure 1: Using a configurable IEEE 802.11b wireless card, (such as Atheros used here), a cheater can reduce his contention window size to increase his share of throughput at the expense of the well-behaved node; error bars show max and min values of 5 real tests; two nodes (including the cheater) send full data rate UDP flows to a common destination; channel capacity is 11 Mbits/s.

we consider the simple case of a network with a single cheater. We then assume the presence of several cheaters, and show that, without appropriate precautions, the network collapses. We compute the Pareto-optimal point of operation of such a system, and study the equilibria of dynamic games. We introduce the notion of *cooperative players*, namely cheaters who try to continue operating at the Pareto-optimal point of operation. We also propose a detection and a punishment technique against those players who exhibit a non-cooperative behavior. Finally, we explain how the players can collectively search for the optimal point of operation, even if they are unaware of the number of nodes present in the network.

To the best of our knowledge, this paper is the first to provide a systematic analysis of rational cheating in CSMA/CA networks. To make it as concrete as possible, we refer to the most prominent incarnation of CSMA/CA, namely IEEE 802.11; however, the conclusions we derive are valid for any protocol of the CSMA/CA family.

The rest of the paper is organized as follows. The next section addresses the related work. Section 3 describes the system model considered in the paper. Section 4 studies the case of a static game, whereas Section 5 studies the case of a dynamic game. Section 6 shows the distributed implementation of the repeated game and its evaluation. Section 7 analyzes the distributed cheating protocol. Last, Section 8 concludes the paper.

2. RELATED WORK

The problem of non-cooperative nodes in ad hoc networks has been widely addressed on the network layer, whereas little work has been done on the MAC layer. MacKenzie and Wicker [1] study the problem of selfish users in Aloha from a game-theoretic point of view. They analyze the stability of the system (Nash equilibrium), and calculate the transmission probabilities that optimize each node's throughput. They assume however that all nodes have the same transmission rates and costs. Moreover, every node has an *a priori* knowledge about the total number of nodes in the system.

Altman et al. [2] reconsider the same Aloha "game" with partial information, where the transmission probability is adapted according to collision feedback only. They consider two frameworks: team work and non-cooperative game.

Jin and Kesidis [3] study non-cooperative equilibria of Aloha networks for heterogeneous users.

For IEEE 802.11, Kyasanur and Vaidya [4] propose that the receiver assigns the backoff value to be used by the sender, so the former can detect any misbehavior of the latter. If the sender deviates from the assigned value, it will be assigned high backoff values on the next round to compensate its deviation. As mentioned by the authors, this mechanism has several limitations such as the possible collusion between sender and receiver, and mainly the fundamental change to the protocol.

Konorski [5] proposes a misbehaviour resilient backoff algorithm that exhibits the same limitation of changing the current protocol.

There are similarities between non-cooperative games on the MAC layer and on the transport layer [6]. From a game-theoretic point of view, Akella et al. [6] analyze the stability of a network where each node is capable of changing its TCP congestion avoidance parameters to individually increase its throughput.

Game theory has been applied in the study of optimal routing [7, 8, 9], congestion control [10], power control [11, 12], as well as incentive engineering in wireless access networks [13].

3. SYSTEM MODEL

We consider N nodes, with radio communication capability, willing to transmit data to N designated receivers (one per sender). In this study, we assume all the nodes to be within the same communication range (i.e., each node can hear any other node). This is to avoid complications introduced by the *hidden terminal problem*. Nodes use a CSMA/CA based protocol to resolve contention at the MAC layer. In this paper, we will be dealing exclusively with IEEE 802.11 [14]; we note that the analysis carried out in this paper can also be extended to other CSMA/CA based protocols. We further assume each node to have authentic MAC layer identity. This can be achieved by means of MAC layer authentication. A possible way to relax this assumption is discussed in Section 7. Finally, we assume that the nodes are static and they always have packets (of the same size) to send.

We consider a scenario where out of N nodes, a subset of C nodes deliberately fail to follow the IEEE 802.11

protocol. We designate this subset of nodes as *cheaters*. There can be a number of ways in which a node can *cheat*. In this paper, we restrict ourselves to a model in which a cheater is in the full control of modifying his contention window size. In violation of the standard protocol, a cheater i initializes his window size to a lower value in order to obtain a higher throughput. We will call this lower value W_i throughout the paper. Moreover, a cheater does not respect the backoff principle and keeps his contention window size fixed after a collision, i.e. equal to W_i. This mode of cheating is the easiest for potential cheaters, since it does not require to perform changes in the operation of IEEE 802.11 protocol. As a proof of concept, we set the contention window of the Proxim Orinoco 802.11a/b/g Gold ComboCard [15] to arbitrary values. However, the main conclusions of this paper are applicable to any other cheating technique, and to the wide range of Atheros chip-based wireless cards (e.g. DLink, Linksys, Netgear and Proxim).

The relevance of these misbehaving techniques become even higher with the emerging standards for the Quality of Service support, such as IEEE 802.11e [16]. This last gives the users total control of the MAC parameters, therefore enabling them to easily cheat.

A cheater is well aware that his decisions can cause different outcomes, only few of which are preferable to him. Thus, a cheater has different preferences for different outcomes. We assume that the cheaters in our model are rational, i.e., they want to maximize their own benefit. In this particular context, the cheaters want to maximize the average throughput they receive r_i.

This problem can easily be modelled in a game theory framework. All the cheater nodes are the *players* in this game. The strategy of each (cheater) player i consists in setting the value of his contention window W_i such that player i's expected payoff (utility) U_i is maximized. In this work, we define a player i's utility to be equal to the enjoyed throughput r_i (i.e., $U_i = r_i$).

4. STUDY OF A STATIC GAME

In this section, we first analyze the problem of misbehaving from a perspective of a single cheater and then consider more complex scenarios of multiple cheaters in the system.

4.1 Variation of throughput with W_i

In [17], Bianchi presented a saturation throughput model for the IEEE 802.11 protocol. Since we assume that a cheater's objective is to maximize his throughput (and we assume he always has a packet to send), he will tend to use the full channel capacity (i.e., the system will operate at the saturation point). Therefore, we make use of the same model as [17]. To estimate the throughput of IEEE 802.11, in a network with no misbehaving nodes, Bianchi [17] used a two-dimensional Markov chain of m backoff stages in which each stage represents the backoff time counter of a node. A transition takes place upon collision and successful transmission, to a higher stage and to the first stage respectively.

Considering the stationary distribution of the chain, the *access probability* τ of a node is derived as a function of the number of levels m and the minimum contention window value W_{min}:

$$\tau = \frac{2}{1 + W_{min} + pW_{min}\sum_{i=0}^{m-1}(2p)^i} \quad (1)$$

where p is the conditional probability that a transmitted packet collides, that is:

$$p = 1 - (1-\tau)^{N-1} \quad (2)$$

where N is the number of the contending nodes. Equations (1) and (2) form a system of two nonlinear equations that has a unique solution [17].

The throughput enjoyed by a given node, which is the average information payload transmitted in a slot time over the average length of a slot time, can be computed using Bianchi's model as follows:

$$r_i = \frac{P_i^s \overline{L}}{P^s T^s + P^c T^c + P^i T^i} \quad (3)$$

where $P_i^s = \tau_i \prod_{j \neq i}(1-\tau_j)$ is the probability that an arbitrary station successfully transmits during a random time slot; $\overline{L}$ is the average size of a packet; $P^s = \sum_k P_k^s$; T^s is the average time needed to transmit a packet of size $\overline{L}$ (including the inter-frame spacing periods [17]); $P^i = \prod_k (1-\tau_k)$ is the probability of the channel being idle; T^i is the duration of the idle period (a single slot); $P^c = 1 - P^i - \sum_k P_k^s$ is the probability of collision; T^c is the average time spent in the collision. Note that for the expression (3) we must have the following expression satisfied: $P^s + P^c + P^i = 1$.

We extend Bianchi's model to describe a network with misbehaving nodes. Two separate Markov chains are considered. The first, with a single backoff stage, since cheaters are assumed to fix their contetion windows, is used to derive the cheaters' access probabilities τ_i^c. The second chain, similar to the one in [17], is used to derive the access probabilities τ_i^l of well-behaved nodes. The conditional collision probabilities are derived considering both well-behaved and cheating nodes access probabilities.

As mentioned in Section 3, cheater i sets his contention windows to some value $CW_{min} = CW_{max} = W_i$. Therefore, cheater i has one backoff stage ($m = 1$), and his accessing probability reduces to:

$$\tau_i^c = \frac{2}{W_i + 1} \quad (4)$$

The channel accessing probability for well behavedr-nodes, τ^l, is expressed by:

$$\tau^l = \frac{2}{1 + W_{min} + p^l W_{min}\sum_{i=0}^{m-1}(2p^l)^i} \quad (5)$$

where

$$p^l = 1 - (1-\tau^l)^{N-C-1}(1-\tau^c)^C \quad (6)$$

After some algebraic manipulation of equation (3), we obtain the following expression for the throughput of a cheater i:

$$r_i = \frac{\tau_i^c c_1^i}{\tau_i^c c_2^i + c_3^i} \quad (7)$$

where $c_1^i = p_{-i}\overline{L}$; $c_2^i = p_{-i}(T^s - T^i) - s_{-i}(T^s - T^c)$; $c_3^i = (1-p_{-i}-s_{-i})T^c + s_{-i}T^s + p_{-i}T^i$; with the following substitutions: $p_{-i} = \prod_{j\neq i}(1-\tau_j^c)\prod_k(1-\tau_k^l)$; $s_{-i} = \sum_{j\neq i}\tau_j^c \prod_{k,d\neq j,i}(1-\tau_k^c)(1-\tau_d^l)$.

It is important to notice here, that the only parameter that a node has a control over is its own W_i. By varying W_i, a node changes its own access probability τ_i, as well as the access probabilities of the other nodes. The exact dependency between these quantities can be deduced from equations (4)-(6).

Let us assume that r_i is a continuous function of W_i. Although the accessing probabilities of other cheaters (and thus expressions c_1^i, c_2^i and c_3^i) loosely depend on τ_i^c, we neglect this dependence for a first degree analysis. This approximation allows us to elaborate a closed form expression of the first derivative of equation (7):

$$\frac{\partial r_i}{\partial W_i} = \frac{\partial r_i}{\partial \tau_i^c}\frac{\partial \tau_i^c}{\partial W_i} \quad (8)$$

$$= \frac{c_1^i c_3^i}{(\tau_i^c c_2^i + c_3^i)^2}\frac{-2}{(W_i+1)^2} \quad (9)$$

which, for $T^s \geq T^c$ and $\tau_j^c < 1$, $j \neq i$, is always negative[2]. We conclude that the expected received throughput r_i is a strictly decreasing function of W_i (for $\tau_j^c < 1$, $j \neq i$). Thus, by unilaterally decreasing its own W_i, a node can increase its received throughput (except if $\tau_k^c = 1$, for some player k; we will treat this case later in the text). Note that this conclusion would remain the same even if we considered the dependence of c_1, c_2 and c_3 on τ_i^c. In fact, by using this approximation, we actually undermine the benefits of the cheater (the cheater gets more throughput in reality, as will be shown shortly).

We will now verify this claim by simulations performed in *ns-2* [18]. The simulation setup[3], shown in Table 1, consists of $N = 20$ nodes randomly spread over a 100×100m area, all within receive range of each others (no hidden nodes). Node X deliberately fails to adhere to the protocol and tries to misbehave following the cheating model presented in Section 3. Traffic sources are constant bit rate, sending 1050-byte frames every 5 ms. This is enough to saturate the 2 Mbits/s (1.6 Mbits/s effective data rate) channel, even when a single node is transmitting. The parameter values for the IEEE 802.11 protocol are chosen according to the IEEE 802.11b standard [14].

[2]According to IEEE 802.11 [14] we have the following: $Ts = PHYheader + MACheader + \overline{L} + SIFS + \sigma + ACK + DIFS + \sigma$, $Tc = PHYheader + MACheader + \overline{L} + DIFS + \sigma$, where σ is the propagation delay. From this we conclude that $T^s \geq T^c$ holds.

[3]In the future, we will only mention the changes that are done from this reference simulation setup.

Table 1: Simulation parameters

Parameter	Value
Propagation	Free space
MAC	802.11b
Scheme	Basic (No RTS/CTS)
Channel capacity	2 Mbits/s
Traffic sources	CBR / UDP 1050-byte frames each 5 ms

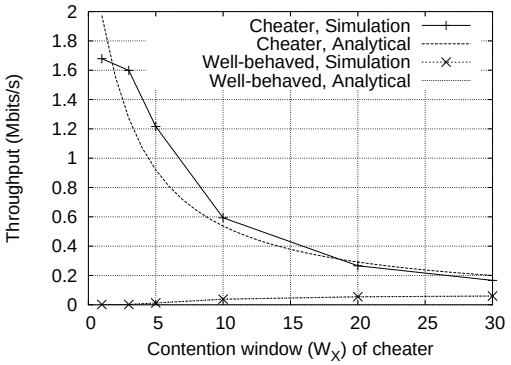


Figure 2: Throughputs for 20 nodes, out of which one is a cheater

Figure 2 plots the throughput obtained by cheater X, as well as by each well-behaved node for different values of W_X. Simulation results show a good match with the analytical results. The duration for each simulation run is 50 seconds and the results are averaged over 5 simulation runs. As can be observed from the Figure 2, the cheater can increase his expected payoff (received throughput) by choosing a small value of W_X. Furthermore, the throughput obtained by the cheater increases monotonically with the decrease in W_X.

A cheater gets the best throughput by setting his contention window size (W_X) to 1. Note that $W_X = 1$ corresponds to the case when the cheater accesses the channel right after DIFS, i.e., his access probability τ is equal to 1.

4.2 Nash equilibrium of a static game

Based on the cheaters' payoff function defined above, in this subsection, we will see that a network is always headed towards collapse when there is no explicit (or implicit) cooperation between cheaters. We will use a model of static games. A static game is one in which all players make decisions (or select a strategy) simultaneously, without knowledge of the strategies that are being chosen by other players. This model represents well the ad hoc setting where explicit cooperation between nodes is impractical. The solution concept we will be using to study the equilibrium of the static game is Nash equilib-

rium [19]. In this study we do not consider well-behaved nodes.

DEFINITION 1 (NASH EQUILIBRIUM). *A strategy profile $W = (W_1, \ldots, W_C)$, which is the set of contention window values used by players, is a Nash equilibrium if and only if, for every player $i = 1, \ldots, C$*

$$r_i(W_i, W_{-i}) \geq r_i(W_i^{'}, W_{-i})$$

for all $W_i^{'} \in S_i$, where S_i is a strategy set of player i and $W_{-i} = (W_1, \ldots, W_{i-1}, W_{i+1}, \ldots, W_C)$.

In other words, no player has an incentive to unilaterally change his strategy. Assuming that players are selfish (i.e., each player aims at maximizing his received throughput), the network is expected to operate at a Nash equilibrium point (the player's best response to other players' strategies). In this subsection, we first investigate whether a Nash equilibrium point exists for the system or not. We will study the existence of a Nash equilibrium point by making use of the concept of a player's *best-response function (correspondence)* [19].

DEFINITION 2. *We say that player i's best-response correspondence $b_i : S_{-i} \rightarrow S_i$, is the correspondence that assigns to each $W_{-i} \in S_{-i}$ the set $b_{W_{-i}} = \{W_i \in S_i : U_i(W_i, W_{-i}) \geq U_i(W_i^{'}, W_{-i})$ for all $W_i^{'} \in S_i\}$.*

Now, we can restate the definition of a Nash equilibrium as follows: The strategy profile $W = (W_1, \ldots, W_C)$ is a Nash equilibrium of our game if and only if $W_i \in b_{W_{-i}}$ for $i = 1, \ldots, C$.

PROPOSITION 1 (STATIC GAME). *For any strategy profile $W = (W_1, \ldots, W_C)$ that constitutes a Nash equilibrium, $\exists i$ s.t. $W_i = 1$.*

PROOF. We prove this proposition by making use of the player's best-response correspondence. Assume $W = (W_1, \ldots, W_C)$ is such that $W_j > 1$, $j = 1, \ldots, C$. Now, take one player, say i, and try to calculate his best-response correspondence. Thus, $b_{W_{-i}} = \{W_i \in S_i : U_i(W_i, W_{-i}) \geq U_i(W_i^{'}, W_{-i})$ for all $W_i^{'} \in S_i\} = \{W_i \in S_i : r_i(W_i, W_{-i}) \geq r_i(W_i^{'}, W_{-i})$ for all $W_i^{'} \in S_i\}$. Since r_i is a decreasing function of W_i (equation (9) and assumption $W_j > 1$ ($\tau_j^c < 1$; equation (4)), $j = 1, \ldots, C$), it readily follows that the only value for W_i that satisfies $r_i(W_i, W_{-i}) \geq r_i(W_i^{'}, W_{-i})$ for all $W_i^{'} \in S_i$, is unity (i.e., $W_i = 1$). Finally, assuming that $\tau_k^c = 1$ for some player k, the proposition follows trivially from equation (4). □

PROPOSITION 2. *There exists an infinite number of Nash equilibria.*

PROOF. Assume that for player i we have $W_i = 1$. Then his access probability $\tau_i^c = 1$ and consequently for all players $j \neq i$, $p_{-j} = (1 - \tau_i^c) \prod_{k \neq j,i} (1 - \tau_k^c) = 0$. From equation (7) it follows that $r_j = 0$ ($j \neq i$) for any value of W_j. Thus for any value of W_j we have $W_j \in b_{W_{-j}}$. This is clearly true for any number of players who have their contention window set to 1. □

Proposition 1 gives us an insight into the characteristics of Nash equilibria points. It is interesting to notice that the equilibria can be classified in the two following families: (i) In the first family, there is only one selfish player i who receives a non-null throughput $r_i > 0$ and for all other players j we have $r_j = 0$; (ii) In the second family, we have more than one selfish node, in which case $r_k = 0$, for $k = 1, \ldots, C$. Since we assume that each cheater strives to attain the most of the channel capacity, the most likely equilibrium is the one at which all cheaters have the size of the contention window W_i set to unity.

COROLLARY 1. *In the presence of more than one cheater, each player's payoff is zero ($r_i = 0$, $i = 1, \ldots, C$).*

This corollary implies that multiple cheaters in the system will eventually converge to the Nash equilibrium point of the static game where they will obtain no payoff. Thus, selfish behavior by multiple cheaters in the system results in disaster for every cheater in the system! This result is know as a *tragedy of commons* in game theory [19].

In this section, we concluded that when there is no explicit (or implicit) cooperation between cheaters, the system is headed towards collapse. In Section 5, we will show that more desirable outcomes are possible when cheaters take into consideration the moves of other cheaters. In this direction, in the following section, we first define what we mean by an *optimal* (and *efficient*) point of operation.

4.3 Pareto-optimal point of operation

According to the analysis of the earlier section, there exists a Nash equilibrium point at which all cheaters have their W_i equal to 1, i.e., every cheater simultaneously tries to access the channel all the time, which results in repeated collisions. This is clearly inefficient and leads us to question whether anything better can be achieved or not.

To investigate further, let us temporarily remove the assumption that every cheater acts independently of the presence of other cheaters. If a cheater is aware of the presence of another cheater in the system, operating at $W_i = 1$ is not the best strategy, as it results in a zero payoff for both cheaters. The situation is analogous to the problem of the Prisoner's Dilemma [19] in game theory. Let us first investigate whether there exists a value of W_i such that a better throughput can be achieved. We consider a hypothetical scenario where all the cheaters use the same value $W_i = W$ and modify it in synchronization with the other cheaters in the system.

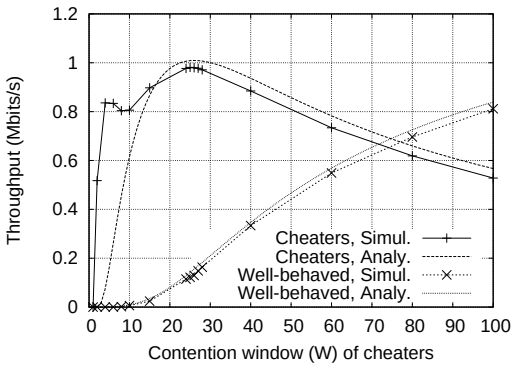


Figure 3: Throughput vs. contention window size of the cheaters (20 nodes, out of which 10 cheaters)

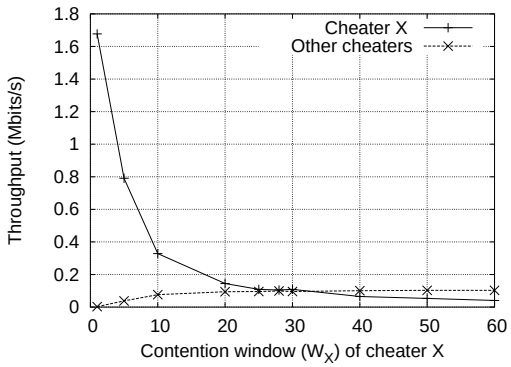


Figure 4: Unilateral deviation by a cheater from the optimal point of operation

The simulation set up consists of $N = 20$ nodes out of which $C = 10$ nodes. The simulation parameters are the same as in Section 4.1 (Table 1). Figure 3 plots the average throughput obtained by a cheater at different values of W, from simulations and from the analytical model described in Section 4.1. The results are averaged over 5 simulation runs. Figure 3 confirms that operating at $W = 1$ leads to network collapse. However, there exists an optimal point of operation ($W = 27$) at which the throughput is maximized for every cheater in the system. We refer to this optimal point of operation as W^*. For $W < W^*$, congestion dominates and results in lower throughput, whereas for $W > W^*$, the system is not used to its full potential. Thus, besides trying to maximize his own probability of accessing the channel, a cheater cooperates by giving sufficient opportunities to other cheaters to access the channel.

The results obtained by simulations show a good match with the analytical results, that were obtained using Bianchi's model (in Matlab). There is a slight discrepancy at low values of W, as Bianchi's model holds only for large values of N and W.

Stability analysis. We investigate the stability of this optimal point of operation from a game theory perspective. We want to verify whether W^* corresponds to the Nash equilibrium for the system or not. To do this, we now observe the payoff obtained by a *single cheater* when it unilaterally deviates from the optimal point of operation. We use the same simulation setup (N=20, C=10). We fix up a simulation seed and run a similar simulation as mentioned in the previous section to calculate the optimal value of W, $W^* = 27$. A cheater (designated as node X) is randomly chosen and is made to unilaterally deviate from this optimal W^* value of 27. Figure 4 plots the throughput obtained by cheater X at different values of X's contention window, W_X. All the other cheaters in the system keep their contention window size fixed, equal to $W^* = 27$. Note that unlike previous figures, Figure 4 plots the results obtained by a single simulation run. This is because a Nash equilibrium is defined for an instance of a static game. An average Nash equilibrium point does not have any physical significance.

As can be observed from Figure 4, if the cheater X deviates towards the right ($W_X > 27$), he suffers a decrease in throughput but if he deviates towards the left ($W_X < 27$) he enjoys an increase of his throughput. This means that a unilateral deviation from the optimal point of operation results in a better payoff for player X and hence it is not a Nash equilibrium point (which is in accordance with our findings in Section 4.2).

However, the deviating player X gets this payoff at the cost of other player(s) (other cheaters) and hence this point is Pareto optimal [20]. A Pareto optimal point means that it is impossible to move from that point in such a manner that the payoff enjoyed by other cheaters does not change. Moreover, the payoff of every cheater is maximized simultaneously. We further note that all points $W \leq W^*$ are Pareto optimal. However, as shown in Figure 3, the cheaters payoff is maximized for a single Pareto optimal point ($W = W^*$). We also note that $W = W^*$ is the only point for which a *max-min fair* allocation [21] of the system capacity is achieved. It is this point that we call the optimal and efficient point of operation, or shortly *Pareto optimal point*.

In our context, this is significant since: (i) W^* is not a Nash equilibrium point, but it is Pareto optimal; (ii) $W = 1$ is a Nash equilibrium point but is not Pareto optimal (efficient). This creates a dilemma for a system with multiple cheaters, where we look for a Nash equilibrim point of operation which is also Pareto optimal and efficient.

5. STUDY OF A DYNAMIC GAME

Having determined the Pareto-optimal point W^*, we now intend to devise a strategy allowing the cheaters to converge to this point. For this purpose, we make use of the theory of *dynamic and repeated games* [22, 19]. These kind of games capture the idea that a player is allowed to use strategy that depends on previous actions. Using the dynamic game model, we devise a simple distributed algorithm that adopted by the cheaters converges to a desired Nash equilibrium point. In this section, we assume, without any loss of generality, that the cheaters

can directly manipulate their accessing probabilities τ_i, rather than their respective contention windows W_i; indeed, accessing probabilities and contention windows are interchangeable by means of the Bianchi's model.

5.1 Game formulation

We extend the game theory model introduced in Section 4.1 to a dynamic game model in which the players are allowed to make their decisions based on previous actions and system states. We assume that our dynamic game is played infinitely long. We also assume all the nodes to be cheaters, i.e., $C = N$. Let $\mathcal{C}$ denote the set comprising all the cheaters. In the new game theoretic model the cheater's (player's) utility function J_i takes the following form:

$$J_i = U_i - P_i, \tag{10}$$

where P_i denotes a penalty function. Let us, for the moment, assume that P_i is defined to be:

$$P_i = k_i(\tau_i - \underline{\tau}), \quad k_i \geq 0, \quad \underline{\tau} \in (0,1), \tag{11}$$

with k_i and $\underline{\tau}$ are constants imposed on player i. Note that we assume P_i can be negative; we will show in Section 5.3 that with an appropriate selection of k_i we always have $P_i \geq 0$ (i.e., $\tau_i \geq \underline{\tau}$). In this section, the definition of P_i, (as well as k_i and $\underline{\tau}$), is used as a mathematical tool to derive optimality conditions for the players; later in Section 5.3, we discuss practical interpretations of values k_i and $\underline{\tau}$.

Combining equations (10), (11) and (7) (definition of U_i) we can define the following non-cooperative game:

$$\max_{0\leq\tau_i\leq 1} J_i = \frac{\tau_i c_1^i}{\tau_i c_2^i + c_3^i} - k_i(\tau_i - \underline{\tau}), \quad \forall i \in \mathcal{C}. \tag{12}$$

In order to solve the maximization problem (12), we define the Lagrangian function $L(\tau_i, \lambda_i)$ for each player i as

$$L(\tau_i, \lambda_i) \;\; = \;\; J_i(\tau_i) + \lambda_i(1 - \tau_i) \tag{13}$$

where $\lambda_i \geq 0$ is a Lagrangian multiplier. Based on the Lagrangian function $L(\tau_i, \lambda_i)$, we can obtain necessary and sufficient conditions for J_i to be maximized for each player $i \in \mathcal{C}$. It is known from the convex optimization theory (with inequality constraints) that in order for the Kuhn-Tucker first order conditions to be sufficient for J_i to be maximized, J_i has to be a concave function, whereas constraint $\tau_i \leq 1$ has to be convex (or quasiconvex) [23]. Indeed, $\tau_i \leq 1$ is convex. Furthermore, for $T^s = T^c$, J_i is a concave function in τ_i; i.e., $\frac{\partial^2 J_i}{\partial \tau_i^2} = -\frac{2c_1^i c_2^i c_3^i}{(\tau_i c_2^i + c_3^i)^3} \leq 0$. As can be seen from Figure 5, the assumption that $T^s = T^c$ results in a negligible decrease in the player's payoff at the optimal value of τ_i with respect to his payoff achieved for $T^s > T^c$. Now, for each player i, the sufficient conditions for J_i to be maximized are:

$$\begin{array}{c} \frac{\partial J_i}{\partial \tau_i} - \lambda_i \leq 0, \tau_i \geq 0 \text{ and } \tau_i\Big(\frac{\partial J_i}{\partial \tau_i} - \lambda_i\Big) = 0 \\ \tau_i \leq 1, \lambda_i \geq 0 \text{ and } \lambda_i(\tau_i - 1) = 0. \end{array} \tag{14}$$

Solving system (14), we obtain the following optimality conditions for player i: (i) $\tau_i = 0$ if $\frac{\partial J_i}{\partial \tau_i} \leq 0$; (ii) $\tau_i \in (0,1)$ if $\frac{\partial J_i}{\partial \tau_i} = 0$ and (iii) $\tau_i = 1$ if $\frac{\partial J_i}{\partial \tau_i} \geq 0$. Let us replace τ_i in equation (12) by $\gamma_i \in \mathbb{R}$ (note that we can have $\gamma_i < 0$ or $\gamma_i > 1$). Let γ_i^* denote the solution to equation $\frac{\partial J_i}{\partial \gamma_i} = 0$, i.e.,

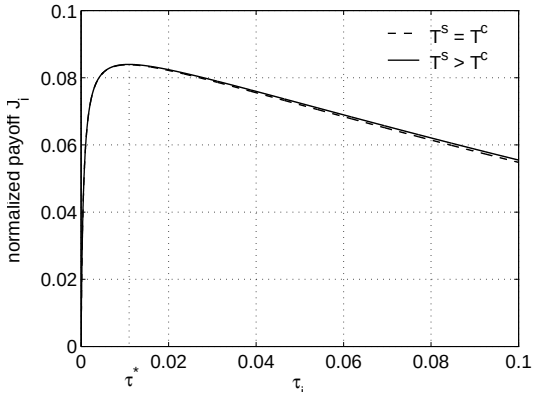


Figure 5: Payoffs achieved for $T^s = T^c$ and $T^s > T^c$ (10 cheaters, $\tau_i = \tau_j,\ \forall i,j \in \mathcal{C}$)

$$\gamma_i^* = \frac{1}{c_2^i}\Big(\sqrt{\frac{c_1^i c_3^i}{k_i}} - c_3^i\Big). \tag{15}$$

It can be shown[4] that the optimality conditions derived above are all expressible in terms of γ_i^* as stated below:

LEMMA 1. *Assume $T^s = T^c$. Then for each player i, a strategy prescribing: (i) play $\tau_i = 0$ if $\gamma_i^* < 0$, (ii) play $\tau_i = \gamma_i^*$ if $\gamma_i^* \in [0,1]$, and (iii) play $\tau_i = 1$ if $\gamma_i^* > 1$, is a unique Nash equilibrium strategy for the game (12).*

It is interesting to note that even if $T^s > T^c$, it still can be shown that by an appropriate selection of penalty constant k_i the optimality conditions of Lemma 1 still hold.

5.2 Equilibria of a dynamic game

Lemma 1 reveals an interesting point about our game (12): If we can find conditions for which $\gamma_i^* \in (0,1)$ for all players $i \in \mathcal{C}$, then no player i will ever play $\tau_i = 1$ (i.e., the network will not collapse). We extend this observation in the following proposition:

PROPOSITION 3. *Any profile $\tau = (\tau_1, \tau_2, \ldots, \tau_C)$, with $\tau_i \in (0,1),\ \forall i \in \mathcal{C}$, can be supported as a Nash equilibrium point of the game (12).*

PROOF. First note that in CSMA/CA networks only one station can transmit successfully at a time. This separation allows us to inflict a penalty on one player without affecting any other player.

Since $\tau_i \in (0,1),\ \forall i \in \mathcal{C}$, the following is satisfied: $c_1^i > 0$, $c_2^i > 0$ and $c_3^i > 0$. By taking the derivative of γ_i^* with

[4]We omit the proof due to the lack of space.

respect to k_i we obtain: $\frac{\partial \gamma_i^*}{\partial k_i} = -\frac{1}{2c_2^i}\sqrt{\frac{c_1^i c_3^i}{k_i^3}} < 0$, i.e., γ_i^* is a decreasing function in k_i. Actually, $\frac{\partial \gamma_i^*}{\partial k_i} = 0$ only for $k_i = \infty$. We further observe that $\lim_{k_i \to 0} \gamma_i^* = \infty$ and $\lim_{k_i \to \infty} \gamma_i^* = -\frac{c_3}{c_2}$. Since γ_i^* is a decreasing function in k_i that takes values from interval $I = [\infty, -\frac{c_3}{c_2}]$, we conclude that there exists $0 < k_i < \infty$ that generates an arbitrary $\gamma_i^* \in (0,1) \subseteq I$. But this value of γ_i^* is a Nash equilibrium of our game according to Lemma 1, which concludes the proof. □

This type of result is similar to the *Nash Folk theorem* in economics [19]. In the following section, using insights from Lemma 1 and Proposition 3, we show how to convert an arbitrary profile $\tau = (\underline{\tau}, \ldots, \underline{\tau})$, $\underline{\tau} \in (0,1)$, into a unique Nash equilibrium. We also propose a simple algorithm that converges to this unique equilibrium point.

5.3 Achieving a Nash equilibrium point

Let us assume that we want to convert $\tau_i = \underline{\tau} \in (0,1)$, $\forall i \in \mathcal{C}$ to a unique Nash equilibrium point. Assume first that $T^s = T^c$. Then we know from Lemma 1 that at a Nash equilibrium τ_i should satisfy equation (15). As we want to make $\underline{\tau}$ a Nash equilibrium, we simply set $\gamma_i^* = \underline{\tau}$. Then rewrite (15) and obtain:

$$k_i = \frac{c_1^i c_3^i}{(\underline{\tau} c_2^i + c_3^i)^2}. \tag{16}$$

Thus, the player i's payoff function becomes:

$$J_i = \frac{\tau_i c_1^i}{\tau_i c_2^i + c_3^i} - \frac{c_1^i c_3^i}{(\underline{\tau} c_2^i + c_3^i)^2}(\tau_i - \underline{\tau}). \tag{17}$$

An example of J_i is shown in Figure 6. Note that at the Nash equilibrium point $\underline{\tau}$ the following holds: $J_i(\underline{\tau}) = U_i(\underline{\tau})$. It is interesting to notice that even if P_i takes negative values for $\tau_i < \underline{\tau}$, in which case P_i can be seen as a reward function, the best choice for the node is still $\tau_i = \underline{\tau}$, where $P_i = 0$. This suggests the following redefinition of player i's payoff function J_i:

$$J_i = \frac{\tau_i c_1^i}{\tau_i c_2^i + c_3^i} - \frac{c_1^i c_3^i}{(\underline{\tau} c_2^i + c_3^i)^2} \times \begin{cases} (\tau_i - \underline{\tau}), & \tau_i \geq \underline{\tau}; \\ 0, & \tau_i < \underline{\tau}. \end{cases} \tag{18}$$

Thus, by an appropriate selection of k_i, we have made $\underline{\tau}$ a unique Nash equilibrium for all players $i \in \mathcal{C}$. Next we describe a simple algorithm that leads the players to the unique Nash equilibrium point $\underline{\tau}$. Let us assume, for the moment, that penalty functions P_i, $\forall i \in \mathcal{C}$ are evaluated and inflicted on the players by some *oracle*. Later in this section, we will show how the role of the oracle can be delegated to the players themselves, in which case our algorithm turns into a distributed algorithm. Assume also the players' payoff function is given by equation (17). Consider the following algorithm followed by each player $i \in \mathcal{C}$:

$$\dot{\tau}_i \triangleq \frac{d\tau_i}{dt} = \frac{\partial J_i}{\partial \tau_i}. \tag{19}$$

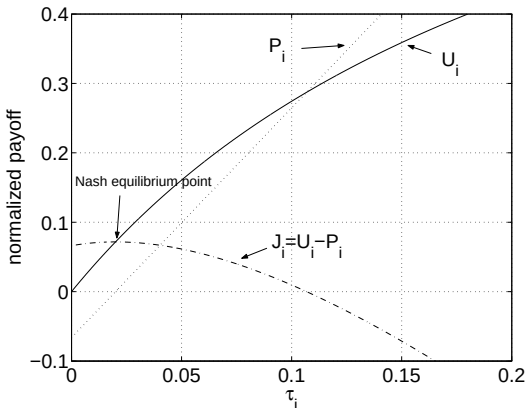


Figure 6: Plot of payoff, utility and penalty functions (10 cheaters, $\underline{\tau} = 0.02$, $\tau_i = \underline{\tau}$ for $i \in \{1, 2, \ldots, 7\}$, $\tau_8 = 0.027$ and $\tau_9 = 0.04$)

PROPOSITION 4. *Algorithm (19) converges to a unique Nash equilibrium $\tau_i = \underline{\tau}$, $\forall i \in \mathcal{C}$.*

PROOF. Following Lyapunov stability theory [24], we first define function $V(\boldsymbol{\tau}) = \frac{1}{2}\sum_{i \in \mathcal{C}} \tau_i^2$. Now shift $V(\boldsymbol{\tau})$ to an equilibrium point $\underline{\boldsymbol{\tau}}$ to obtain $V(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}}) = \frac{1}{2}\sum_{i \in \mathcal{C}}(\tau_i - \underline{\tau})^2$. Note that $V(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}})$ is positive definite, since $V(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}}) > 0$ except for $\boldsymbol{\tau} = \underline{\boldsymbol{\tau}}$ (i.e., $V(\mathbf{0}) = 0$). Next we take the derivative of $V(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}})$ to obtain $\dot{V}(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}}) = \sum_{i \in \mathcal{C}}(\tau_i - \underline{\tau})\dot{\tau}_i$. Combining this with algorithm (19) leads to:

$$\dot{V}(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}}) = \sum_{i \in \mathcal{C}}(\tau_i - \underline{\tau})\Big[\frac{c_1^i c_3^i}{(\tau_i c_2^i + c_3^i)^2} - \frac{c_1^i c_3^i}{(\underline{\tau} c_2^i + c_3^i)^2}\Big] \tag{20}$$

From equation (20) we can conclude that $\dot{V}(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}}) < 0$ if $\tau_i < \underline{\tau}$ or $\tau_i > \underline{\tau}$, while $\dot{V}(\boldsymbol{\tau} - \underline{\boldsymbol{\tau}}) = 0$ only if $\tau_i = \underline{\tau}$, $\forall i \in \mathcal{C}$. Therefore, by Lyapunov stability theorem [24], algorithm (19) converges to $\underline{\tau}$ that is a globally asymptotically stable equilibrium point. □

We observe here that by taking J_i as defined in (18), all the results obtained for the payoff as defined by (17) still hold: (i) $\underline{\tau}$ is still a Nash equilibrium, and (ii) the algorithm (19) (its discrete version) converges to $\underline{\tau}$. To see that this is indeed the case, it suffices to observe that for (17) we have $P_i = 0$ at the Nash equilibrium, whereas for $P_i < 0$ ($\tau_i < \underline{\tau}$) we have $U_i - P_i > U_i$, i.e., in both cases player i has no incentive to stay at $\tau_i < \underline{\tau}$ (see Figure 6).

Next we show how to delegate the role of the oracle to the cheaters themselves and thus convert algorithm (19) to a distributed algorithm. We saw that the role of the oracle is: (i) to correctly estimate the penalty function P_i, and (ii) to apply P_i afterwards. Therefore, in order to be able to delegate these tasks to the players, we have to develop an appropriate *penalization mechanism* as well as a *detection mechanism* to be used by the players. Recall that the punishment P_i of any player $i \in \mathcal{C}$ should not affect

any other player's payoff (as stated in the proof of Proposition 3)). This for example can be done by *selectively jamming* player i's packets. We study the penalization and detection mechanisms in more detail in Section 6.

A still pending question is: how do the players agree on the Nash equilibrium point $\underline{\tau} \in (0,1)$ to which they all should converge? A simple way to resolve this is to define:

$$\underline{\tau} \stackrel{\triangle}{=} \min_{k \in C} \tau_k, \tag{21}$$

in which case P_i can be seen as a penalty that a player with the lowest accessing probability $\tau_j = \underline{\tau}$ (the highest contention window W_j) inflicts on player i. Note that player i is indeed able to determine $\tau_j, \forall i \in \mathcal{C}$; player i is equipped with an appropriate detection mechanism (Section 6). A potential problem with definition (21) is that by following algorithm (19) everyone are guided to $\underline{\tau} \to 0$. This happens because any player i playing $\tau_i < \underline{\tau}$ defines new $\underline{\tau}$ and in turn penalizes all the other players thus guiding them to the new equilibrium point. To overcome this potential danger, we impose the following constraint on all the players:

REMARK 1. *While running the adaptive strategy as suggested in algorithm (19), a player* ***must not*** *use the penalization mechanism.*

Note that the player i satisfying $i = \arg\min_{k \in \mathcal{C}} \tau_k$ does not use the adaptive strategy of algorithm (19), since i knows, by means of the detection mechanism, the he has the lowest access probability.

The remaining challenge of how to achieve the most efficient $\underline{\tau}$, i.e., the Pareto-optimal point in the sense of Section 4.3, we study in the following section.

5.4 Achieving the Pareto-optimal point

Let τ_i^* denote the value of τ_i at which the cheaters jointly (i.e., $\tau_i = \tau_j,\ \forall i,j \in \mathcal{C}$) maximize their respective payoffs. To lead the cheaters to the Pareto-optimal point, we use the fact that the first derivative of J_i with respect to τ_i changes sign at τ^* when $\tau_i = \tau_j,\ \forall i,j \in \mathcal{C}$ (see Figure 5). For this reason we term our algorithm *Gradient-search mechanism* (GSM). In a nutshell, GSM works as follows. The cheaters are running algorithm (19) until they stabilize at some equilibrium point $\tau_i = \underline{\tau},\ \forall i \in \mathcal{C}$. One player, say i, will eventually decrease τ_i by some value δ. This will in turn trigger the penalizing mechanism of player i and pull the other players to a new equilibrium point, since they run algorithm (19). At this stage, each player will compare its current payoff J_j (at the new Nash equilibrium point) to the payoff J_j^{old} achieved at the previous Nash equilibrium point. If $J_j - J_j^{old} < 0$, the players terminate GSM, since they have reached close proximity of the Pareto-optimal point. Since every point at which the players stabilize is a Nash equilibrium point (Proposition 4), we have achieved our goal of making the Pareto-optimal point a Nash equilibrium point. In Sections 6 and 7 we elaborate GSM algorithm in more detail. There we also study convergence and stability properties of GSM.

One important point to notice about GSM is that it requires players to deviate from a Nash equilibrium point to reach the Pareto equilibrium point. At the first sight this may look confusing: why should any player deviate from a Nash equilibrium point? The answer here is simple. The game in our new model is played infinitely long (an infinitely repeated game). For this reason, the players can afford themselves any finite number ($k < \infty$) of deviations from a Nash equilibrium point (e.g., for $0 < k < \infty$, $lim_{T\to\infty} \sum_{t=k}^{T} J_i^t/T = lim_{T\to\infty} \sum_{t=0}^{T} J_i^t/T$). Being aware of the existence of the Pareto-optimal point τ^*, the players clearly have incentive to follow our protocol and achieve better payoffs (> 0) than the one predicted by Corollary 1.

6. DISTRIBUTED IMPLEMENTATION

In this section we move beyond the realms of theoretical results towards a prototype implementation of an efficient cheating protocol in the context of ad-hoc networks.

In general, nodes do not have direct control over there access probability. They can only change their contention window sizes. We note that Bianchi's model provides a one-to-one relationship between access probability and contention window size. This allows us to make a direct transition from access probability to contention window size; there exists $\underline{W}$ (corresponding to $\underline{\tau}$) and W_i^* (corresponding to τ_i^*) at which Nash equilibrium and Pareto equilibrium can be achieved respectively.

The two key building blocks for the model of repeated games are the detection mechanism and the penalizing mechanism. In this section we will propose a distributed implementation of these building blocks, where each node take a decision solely based on its local information.

6.1 Detection mechanism

A non-cooperative cheater increases its access probability, τ_i, by decreasing the contention window size below the equilibrium point, $W_i < \underline{W}$. As a result, it gets more throughput (payoff) than other cheaters in the system. We use this difference in throughputs as the metric for gauging the magnitude of misbehavior by a non-cooperative cheater.

We use the simple approach of communist fairness as the detection scheme: Every cheater must get the same throughput at the point of equilibrium. This property follows from equations (3) and (4); all cheaters with similar traffic constraints and same W_i should get similar throughputs. Note that at the point of equilibrium $W_i = \underline{W},\ \forall i \in \mathcal{C}$. Each node measures the throughput of all nodes, including itself. This is indeed feasible due to the broadcast nature of the wireless medium. If a node observes a difference in throughputs with some other node, it characterizes the other node as the misbehaving cheater.

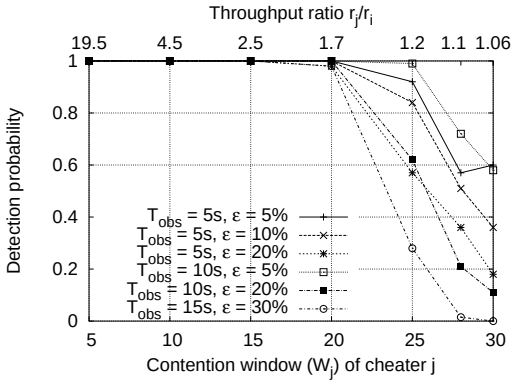


Figure 7: Performance of cheating detection based on throughput measurements

Let r_i and r_j be the measured throughput of nodes i and j, respectively. Due to the inherent unfairness of the IEEE 802.11 MAC [25], and to increase the efficiency of the detection mechanism, we introduce two parameters: the observation time-window size T_{obs}, in seconds, and the tolerance margin ϵ, in percentage of throughput. After measuring the throughput of each node for T_{obs} seconds, node i detects that node j is deviating whenever the throughput of node j exceeds the one of node i i.e., $r_j/r_i > 1 + \epsilon$.

We have implemented this detection mechanism in *ns-2*, with $N = C = 30$ nodes. We vary the contention window size (W_j) of a single node j, while setting others' contention window sizes to 30 ($W_{-j} = 30$). Figure 7 shows the performance of the detection mechanism for different values of T_{obs} and ϵ. The probability of false positives corresponds to the detection probability read at $W_j = 30$ only; at this point, node j uses a contention window value equal to that of node i, but still gets a higher throughput, $r_j/r_i = 1.06$, due to the IEEE 802.11 unfairness. Therefore node j gets detected as deviating (detection probability > 0). To reduce the false positives (at contention window size 30), one can consider large ϵ values ($> 10\%$). However, this comes at the expense of lower detection probabilities if cheater j uses contention window sizes slightly lower than 30. Similarly, large T_{obs} values ($\geq 15s$) will reduce the effect of the inherent IEEE 802.11 unfairness, and therefore the corresponding false positives. This also comes at the expense of lower detection probabilities if cheater j uses contention window sizes slightly lower than 30. Therefore, choosing appropriate values for T_{obs} and ϵ is crucial to our detection mechanism. For very low contention window sizes of cheater j ($W_j \leq 20$), the throughput ratio r_j/r_i is much larger than $1 + \epsilon$, easily detecting cheater j's deviation.

The detection mechanism works on the simple principle of communist fairness i.e., every cheater in the system should get the same throughput. Besides being simple, this detection mechanism is also flexible. It can be used even if cheaters adopt a different cheating model for example abusing the DIFS duration. Moreover, even if different cheaters in the system use different models for cheating, the same detection mechanism can be used for all of them. Note that our detection mechanism assumes homogeneous conditions among the cheaters in the system i.e., similar traffic constraint and channel conditions. In Section 7.3, we discuss how this assumption can be relaxed.

Related security and detection issues. The detection mechanisms explored so far assume that nodes maintain a consistent and unique identity at the MAC layer. Thus, they are not robust to the *sybil attack* [26]. We will require a MAC layer authentication scheme to counter such attacks [27]. Another possibility is to consider a game in which a node plays *against all others* (i.e., the network as a whole), which requires no authentication of other players. However, this game shows relatively low stability and efficiency of the network throughput.

6.2 Penalizing mechanism

The action taken by cheaters in response to non-cooperation by another cheater is termed as penalizing mechanism. In Section 5, we called the resultant cost of this penalizing scheme (imposed by other nodes) on a node as the penalty function. We noted that penalizing scheme should be designed so that it does not brings any performance degradation on the imposing nodes.

This can be achieved by selectively jamming the misbehaving cheater nodes packets. Again, this is made feasible by the broadcast nature of the wireless medium. We have designed a simple penalizing scheme, in which the packets of the non-cooperative cheater are selectively jammed for a short duration of time, T_{jam}, by the other cheaters in the system. Suppose cheater i detects the presence of a non-cooperative cheater j. Thereafter, if node i listens to a transmitted packet corresponding to node j, it switches to transmission mode and jams enough bits so that the packet cannot be recovered at the receiver. Meanwhile, all other nodes in the system should be able to read the header of the jammed frame and properly update their NAV (network allocation vector). This is to avoid waiting for EIFS [14] which reduces the system's efficiency. Therefore, jamming should be done on frame payloads rather than frame headers. This is indeed possible since the transceiver's turnaround time, which is of the order of 5 μs [14], is much shorter than the data frame transmission time, which is of the order of 700 μs.

Let the throughput obtained by the two considered nodes over the last observation window, T_{obs}, be r_i and r_j, respectively, where $r_j/r_i > 1 + \epsilon$. As we saw in Section 5, the penalizing mechanism is aimed at guiding the cheaters to the same Nash equilibrium point. Thus, the throughput received by nodes i and j should be the same over the total time duration of $T_{obs} + T_{jam}$. Node i calculates the value of T_{jam} as follows:

$$\begin{aligned} r_i \times T_{obs} + r_i \times T_{jam} &= r_j \times T_{obs} + 0 \times T_{jam} \\ T_{jam} = \tfrac{(r_j - r_i)}{r_i} \times T_{obs} &= \left(\tfrac{r_j}{r_i} - 1\right) \times T_{obs} \end{aligned} \quad (22)$$

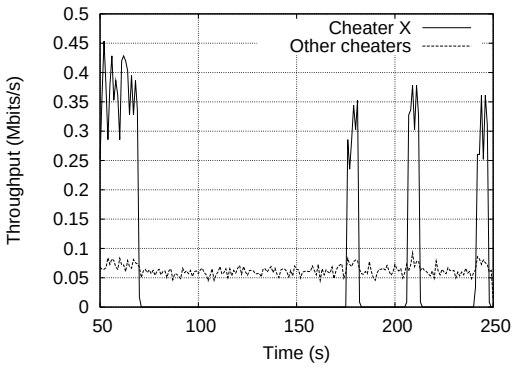


Figure 8: Throughput obtained by cheaters over time, in presence of a non-cooperative cheater X and the jamming mechanism

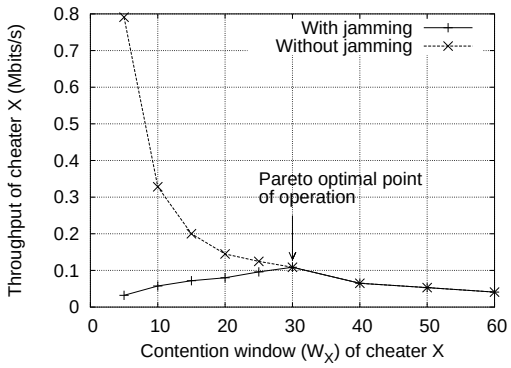


Figure 9: Unilateral deviation by a cheater before and after the introduction of the penalizing scheme

Note that during the jamming duration, node j gets no throughput and node i will continue to get a throughput of r_i. Some of the key features of this penalizing mechanism are as follows:

(a) The jamming mechanism is self-adaptive, i.e., the amount of penalty imposed on a non-cooperative cheater (T_{jam}) is directly proportional to his level of misbehavior $\left(r_j/r_i - 1\right)$.

(b) The responsibility of jamming the packets is collectively carried out by all the cheaters in the system. Moreover, every cheater takes his own independent decision about jamming the non-cooperative cheater, just relying on his local information. Thus the proposed jamming mechanism is completely distributed.

(c) The proposed jamming mechanism is completely anonymous. A penalized node cannot know the identity of the penalizing node. This prevents the node from retaliating against a specific node in the system. In the worst case, it can retaliate against every other cheater in the system. However, as shown earlier, such a malicious behavior will eventually result in a total network collapse (the Nash equilibrium of a static game).

Normally, jamming is associated with malicious behavior in the system. In this context, the introduction of jamming looks counterintuitive considering the assumption of rational cheaters in the system. However, in our proposed mechanism, cheaters use jamming just to support better outcomes than those obtained in the static game (Corollary 1). Thus, the proposed jamming mechanism is consistent with the rational goal of maximizing one's own throughput.

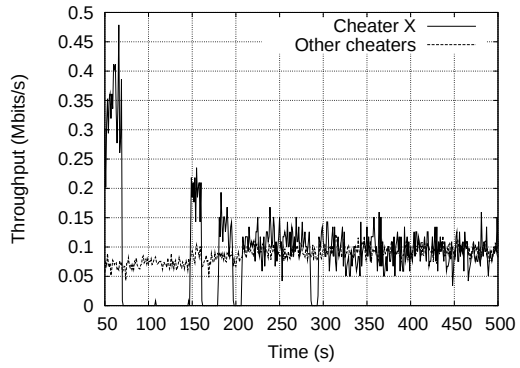
Instead of using the penalizing mechanism, some researchers introduced the notion of cost for reaching a stable point of operation. In [2, 1], battery power is used as a cost mechanism. However, such a mechanism relies on the participating players to respect this cost mechanism, making it susceptible to be abused by non-cooperative cheaters. Jamming is a mutual cost imposed by other players and is not controlled by the player himself. This makes it an *imposed* cost for each non-cooperative player, ensuring that the system has a stable point of operation.

We have implemented the jamming mechanism in *ns-2*. The simulation setup is the same as in Section 4.3 ($N = 20, C = 10$). We randomly pick up a cheater, designated as node X, and fix his contention window size to be 10. The contention window size for all the other cheaters in the system is fixed to the Pareto-optimal (W^*) value of 30. Figure 8 plots the obtained throughput by the cheaters in the system over time, with and without the penalizing scheme. We use the observation window size, T_{obs}, of 20 seconds, and the tolerance margin, ϵ, of 0.5 in the detection mechanism. As can be observed from Figure 8, node X gets detected by other cheaters in the system and is penalized for its misbehavior. The throughput of node X drops then to 0. Thus, the detection and penalizing schemes are successful in preventing the non-cooperative cheater from obtaining higher throughput than other cheaters in the system.

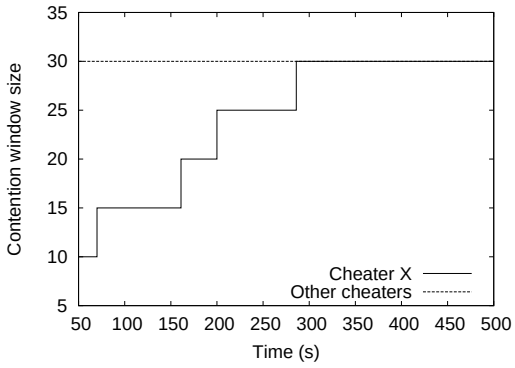
We next carry out a similar stability analysis as in Section 4.3 for the system with the detection and penalizing scheme. Figure 9 plots the average throughput obtained by cheater node X, when it unilaterally deviates from the Pareto-optimal point of operation ($W^* = 30$). The results are averaged over a duration of 1000 seconds. As can be observed from Figure 9, after the introduction of the detection and penalizing scheme, cheater X gets maximum throughput by operating at the Pareto-optimal point of operation. Any unilateral deviation from this point, $W_X < W^*$ or $W_X > W^*$, brings less payoff (throughput) for cheater X. No cheater will have any incentive for deviating unilaterally from the Pareto-optimal point of operation and hence it is at Nash equilibrium.

Last, we should note that the penalizing scheme, proposed here for throughput-based detection, can be easily adapted to work with any other detection mechanism.

7. DISTRIBUTED CHEATING PROTOCOL



(a) Throughput of cheaters



(b) Contention window size of cheaters

Figure 10: Performance of system with adaptive strategy

In this section, we will build a comprehensive, distributed and efficient cheating protocol on the two building blocks mentioned in the previous section. We are still missing one key element in the protocol design What action nodes take after getting penalized by other nodes in the network? We call the scheme followed by the misbehaving nodes as the adaptive strategy.

7.1 Adaptive strategy

Adaptive strategy is inspired by algorithm (19). When a cheater observes that he is being jammed consecutively for Δ seconds, he gradually increases his contention window by steps of size γ. Note that a cheater can easily decide whether he is being jammed by observing his own throughput. The choice of Δ determines the efficiency of the system. A high value of Δ might let a non-cooperative cheater escape from being penalized. However, choosing a small value of Δ might magnify the effect of misdetection by unnecessarily causing a cheater to increase his contention window size. This will eventually lead the whole system towards an inefficient point of operation. The choice of the step size, γ, offers a tradeoff between convergence time and efficiency. If we increase the contention window in large steps, though the system will stabilize in less time, the point of operation might be far away from the Pareto-optimal point (W^*), resulting in an inefficient system and vice-versa.

We have implemented this adaptive strategy in *ns-2*. The simulation setup is the same as in the previous section ($N = 20, C = 10, W^* = 30$). We randomly pick up a cheater, designated as node X, and fix his contention window size to 10. The contention window size for all the other cheaters in the system is fixed to W^*. We fix Δ to be 5 seconds and γ to be 5. Figure 10(a) plots the obtained throughput by different cheaters in the system over time. Figure 10(b) plots the evolution of contention window size of node X over time. As can be observed from Figure 10(b), node X adapts its contention window size following the adaptive strategy and eventually converges to a window size of 30, equal to W^*. Thus the other cheaters in the system are successful in guiding the misbehaving cheater to reach the optimal point of operation. As can be seen from Figure 10(b), node X reaches this point at around 300s. Thereafter, the system continues to operate at this stable point of operation. Table 2 summarizes the average throughput obtained by different nodes over a time interval of 1000 seconds. As can be observed from Table 2, the jamming and detection mechanism combined with the adaptive strategy, besides being fair to all the cheaters in the system, is also the most efficient.

Table 2: Throughput obtained by different nodes (bytes/s)

	Strategy	
	Non-adaptive	Adaptive
Cheater X	7650	11577
Other cheaters	7826	11448
Well-behaved nodes	1286	2318

Note that even the introduction of the adaptive strategy does not encourage the abuse of jamming. Cheater X might try to unnecessarily jam other cheaters, hoping that an increase in the contention window sizes by all the cheaters (following the adaptive strategy) will get him more throughput. However, eventually cheater X will be identified as a misbehaving cheater, because of the throughput difference, by the other cheaters in the system. In turn, cheater X will be forced to increase his own contention window size, following the penalizing mechanism. As a result, every cheater i, including cheater X, will now be operating at an inefficient point of operation ($W_i > W^*$). Thus cheaters have no incentive to over-jam other cheaters in the system.

Finally, we evaluate the performance of our protocol in a scenario consisting of multiple levels of misbehavior in the system. The simulation setup is the same as in the previous section ($N = 20, C = 10, W^* = 30$). We ran-

Table 3: Throughput obtained by different nodes (bytes/s) with multiple levels of misbehavior

	Strategy	
	Non-adaptive	Adaptive
Cheater X	2843	10356
Cheater Y	2686	10185
Cheater Z	2565	10239
Other cheaters	2544	10172
Well-behaved nodes	270	1981

domly pick up three cheaters, designated as node X, Y and Z respectively. We fix their contention window sizes to be 5, 10 and 15, respectively. The contention window size for all the other cheaters in the system is fixed to W^*. Figure 11 plots the evolution of the contention window sizes of the different cheaters over time. Since different

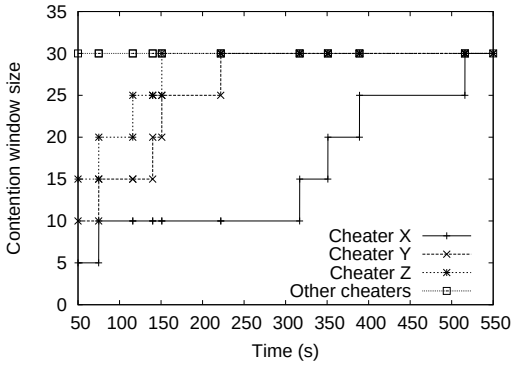


Figure 11: Performance of a system with varying levels of misbehavior

cheaters are punished in proportion to their misbehavior, every cheater eventually converges to W^* and the system continues to operate at this stable point of operation. Thus, our protocol self-adapts to the different levels of misbehavior in the system. Table 3 summarizes the average throughput obtained by different nodes over an interval of 1000 seconds. As can be observed from Table 3, the jamming mechanism combined with the adaptive strategy results in the optimal and fair performance, even with multiple levels of misbehavior in the system.

7.2 Distributed Gradient-search mechanism

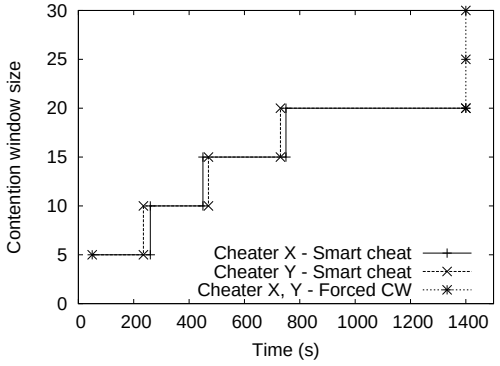
An accurate implementation of detection, penalizing and adaptive strategy will lead the nodes to reach an equilibrium point, $\underline{W}$. However, the intention is to reach the most efficient equilibrium point i.e the Pareto optimal point of operation, W^*. As mentioned in Section 5, this can be achieved by using a simple Gradient Search Mechanism.

At the onset of the system, $W_i = W_i^{in}$ for all cheaters. Every cheater sets up a random timer to increase his contention window by step size, γ. One of the cheaters, say X, will eventually increase his contention window size to $W_X^{in} + \gamma$. Based on the detection mechanism, node X will conclude that all other cheaters in the system are deviating and will begin penalizing them. If a cheater observes that he is being penalized, he will disable the timer. Eventually the system will stabilize, when $W_i = W_i^{in} + \gamma$ for all cheaters. The cheaters realize that they have reached a new stable point of operation, when they all begin getting the same throughput. At this point in time, a cheater i compares his throughput at $W_i = W_i^{in} + \gamma$ with the throughput at $W_i = W_i^{in}$. If a cheater observes a decrease in his throughput, he will terminate the search for W^*. Otherwise he again sets up the random timer to increase his contention window size by γ. Note that even if only one cheater observes an increase in his throughput, eventually the whole system will move towards a new point of operation.

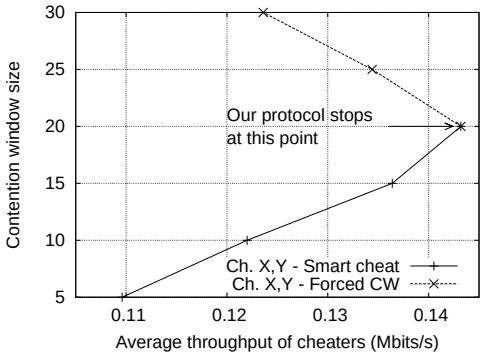
We have implemented this protocol in *ns-2*. The simulation setup consists of 20 nodes and 7 cheaters ($N = 20, C = 7$). All the cheaters follow Protocol 1 above. The cheaters initialize their contention window sizes to 5 ($W_i^{in} = 5$). The cheaters continue their search for W^* only if they see an increase of 10% or more in their throughput from the last stable point of operation. Figure 12(a) plots a sample evolution of the contention window for 2 cheaters, X and Y, in the system. Note that all of the cheaters follow a similar pattern and eventually converge to a window size of 20. We are unable to show their evolution in the same plot as it simply generates overlapping lines. Figure 12(b) plots the average throughput obtained by the cheaters at different contention window sizes. As can be seen from Figure 12(b), the throughput is maximized at $W_i = 20$. In reality, the cheaters will stop at W_i equal to 20 and the system will continue to operate at this point of operation. For completeness, we obtain the "dotted" curve in Figure 12(b) by deliberately forcing the cheaters to go beyond $W_i = 20$.

Next we evaluate the performance of our protocol by varying the number of cheaters in the system. We run our protocol and measure the window size at which all the cheaters eventually converge. Thus, according to our protocol, this point of convergence is the Pareto-optimal point of operation. We evaluate the actual Pareto-optimal point (W^*), under the same network settings, through *ns-2* simulations (similar to Section 4.3). We also evaluate the Pareto-optimal point (W^*) analytically, using Bianchi's model with Matlab. Figure 13 plots the obtained results. The results are averaged over 5 simulation runs. The results obtained by our protocol closely match the analytical results obtained using Bianchi's model. Note that the minimum resolution of our protocol is equal to the step size, $\gamma = 5$. As can be seen from Figure 13, the discrepancy is bounded by $\pm\gamma$, which clearly proves the efficiency of our protocol.

The protocol operates in a completely distributed manner, without requiring any *a priori* knowledge about the optimal point of operation or of the total number of nodes/cheaters in the system. However, we rely on the fact that the numbers of nodes and of cheaters does not



(a) Evolution of the contention windows



(b) Contention window vs. Average throughput

Figure 12: Effect of the Smart cheating protocol, with $N = 20$ and $C = 7$. The axes in (b) are swapped for the convenience of matching them with (a)

change in the system. In general, ad hoc networks are highly dynamic in nature, where new nodes/cheaters can enter or existing nodes/cheaters can leave the system. In order to cope with such uncertainties, we propose that cheaters time out periodically and re-run the whole protocol from the beginning. Note that there can be transient phases during which the system operates at a sub-optimal point of operation.

7.3 Discussion

As can be seen from Table 3, well-behaved nodes, which continue to follow the IEEE 802.11 protocol, obtain negligible throughput from the system. Thus, in the presence of cheaters in the system, the only rational behavior for the nodes (with the capability of cheating) is to adapt and start using our protocol. Therefore, we speculate that eventually all the nodes in the system will begin behaving as cheaters. As we strive for optimal contention window size, even in such a scenario, our protocol will be at least as efficient as the normal IEEE 802.11 protocol. Note that our goal in this paper is to save the network from collapse, in an efficient way, rather than finding the optimal contention window for a network of $N = C$ nodes [28, 17]. However, our adaptive cheating algorithm leads to the same optimal point, without explicit knowledge of the number of contending nodes.

As mentioned earlier, we adopted throughput-based detection to simplify the presentation. However, the use of more adequate detection mechanisms, such as *backoff-based* detection (i.e., comparing the nodes average backoffs that do not depend on the traffic shape) is needed in general, for example in the following scenarios:

(i) Hidden terminal problem. The throughput-based detection mechanism relies on the "communist fairness" that drives all the cheaters to the fairness with the least throughput. Communist fairness may correspond to a point of operation beyond the Pareto-optimal one. This is typically the case of a node belonging to two different

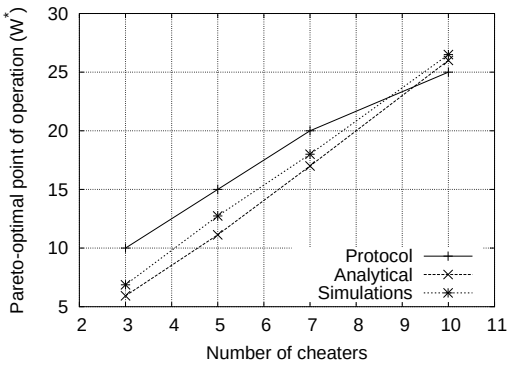


Figure 13: Variation of Pareto-optimal point with the number of cheaters

clusters, hidden to each other. The common node will suffer unfair shares in both clusters, therefore it brings the whole system to a sub-optimal point of operation. To maximize the efficiency of a system with hidden nodes, a detection mechanism based on max-min fairness [21] can be used. This is the case of backoff-based detection mechanisms. On this "packet-scale" measurements, the hidden node problem can be identified and avoided, therefore enforcing max-min fairness while keeping the system highly efficient.

(ii) Different traffic constraints. Since CSMA/CA is time-fair rather than throughput-fair, flows with different constraints (e.g., different packet lengths) will result in different throughputs, even without cheating. Hence, throughput-based detection mechanisms cannot be applied since nodes are not supposed to know each other's traffic constraints. Again, backoff-based detection mechanisms are more appropriate.

8. CONCLUSIONS

In this paper we have addressed the problem of cheating in CSMA/CA ad hoc networks. For this purpose, we

have developed a game-theoretical model and verified our findings by appropriate simulations.

We have made several contributions. First, we have provided a formalism for the systematic study of rational cheating. Second, we have studied the simple cases (i) of a single cheater and (ii) of several cheaters acting without restraint. Third, we have identified the Pareto-optimal point of operation of a network with multiple cheaters. Fourth, we have shown how it is possible to transform this Pareto-optimal point into a Nash equilibrium. Fifth, we have shown that smart cheaters can collectively find this point.

We believe these contributions to be very relevant in ad hoc networks, as each user can alter the behavior of his or her own device.

In terms of future work, we intend to study in more detail the solution we have proposed for the hidden terminal problem. We will also try to define a punishment technique which is less intrusive than jamming. Finally, we intend to adapt this approach to problems beyond CSMA/CA networks: first, we will study how smart cheating could become a technique to collectively fine-tune the behavior of a protocol; second, we intend to apply this approach to other MAC protocols, including those based on CDMA principles.

9. ACKNOWLEDGMENTS

We thank Naouel Ben Salem, Sonja Buchegger, Mark Felegyhazi, Božidar Radunović and Thierry Turletti for helpful discussions and comments.

10. REFERENCES

[1] A. B. MacKenzie and S. B. Wicker, "Stability of Multipacket Slotted Aloha with Selfish Users and Perfect Information," in *Proceedings of IEEE INFOCOM*, 2003.

[2] E. Altman, R. El Azouzi, and T. Jiménez, "Slotted Aloha as a stochastic game with partial information," in *Proceedings of WiOpt*, 2002.

[3] Y. Jin and G. Kesidis, "Equilibria of a Noncooperative Game for Heterogeneous Users of an ALOHA Network," *IEEE Comm. Letters*, vol. 6, 2002.

[4] P. Kyasanur and N. Vaidya, "Detection and handling of MAC layer misbehavior in wireless networks," in *Dependable Systems and Networks*, June 2003.

[5] J. Konorski, "Multiple access in ad hoc wireless LANs with noncooperative stations," in *NETWORKING*, Springer, 2002, vol. 2345 of LNCS.

[6] A. Akella, S. Seshan, R. Karp, and S. Shenker, "Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP," in *Proceedings of ACM SIGCOMM*, 2002.

[7] A. Orda, R. Rom, and N. Shimkin, "Competitive Routing in Multi-User Communication Networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 5, pp. 510–521, 1993.

[8] Y. A. Korilis, A. A. Lazar, and A. Orda, "Architecting Noncooperative Networks," *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 7, pp. 1241–1251, 1995.

[9] R. J. La and V. Anantharam, "Optimal Routing Control: Repeated Game Approach," *IEEE Transactions on Automatic Control*, March 2002.

[10] H. Yache, R.R. Mazumdar, and C. Rosenberg, "A Game Theoretic Framework for Bandwidth Allocation and Pricing in Braodband Networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 667–578, October 2000.

[11] C. U. Saraydar, N. B. Mandayam, and D. J. Goodman, "Efficient Power Control Via Pricing In Wireless Data Networks," *IEEE Transactions on Communications*, vol. 50, no. 2, pp. 291–303, February 2002.

[12] T. Alpcan, T. Basar, R. Srikant, and E. Altman, "CDMA uplink power control as a noncooperative game," in *Proceedengs of 40th IEEE Conference on Decision and Control*, 2001.

[13] R.-F. Liao, Rita H. Wouhaybi, and A.T. Campbell, "Incentive Engineering in Wireless LAN Based Access Networks," in *Proceedings of 10th International Conference on Network Protocols (ICNP 2002)*, 2002.

[14] LAN/MAN Standards Committee, *ANSI/IEEE Std 802.11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Computer Society, 1999.

[15] *Proxim ORiNOCO 802.11A/B/G Gold*, http://www.proxim.com/products/wifi/11bg/.

[16] IEEE 802.11 WG, *ANSI/IEEE Std 802.11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS) IEEE 802.11/D2.0*, IEEE, 2001.

[17] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal of Selected Areas in Communications*, vol. 18, 2000.

[18] *Network Simulator (NS)*, http://www.isi.edu/nsnam/ns/.

[19] D. Fudenberg and J. Tirole, *Game Theory*, The MIT Press, 1991.

[20] K. M. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, 1999.

[21] D. Bertsekas and R. Gallager, *Data networks*, Prentice-Hall International, Inc., 1992.

[22] T. Başar and G.J. Olsder, *Dynamic Noncooperative Game Theory*, SIAM, 1999.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[24] S. Sastry, *Nonlinear System: Analysis, Stability, and Control*, Springer-Verlag Telos, 1999.

[25] C. E. Koksal, H. Kassab, and H. Balakrishnan, "An analysis of short-term fairness in wireless media access protocols," in *Proceedings of ACM Sigmetrics*, 2000.

[26] J. R. Douceur, "The Sybil Attack," in *Proceedings of IPTPS*, March 2002.

[27] J. Edney and W.A. Arbaugh, *Real 802.11 security: Wi-Fi protected access and 802.11i*, Addison-Wesley, 2004.

[28] F. Calì, M. Conti, and E. Gregori, "Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Transactions on Networking*, 2000.