Simulation-based comparison and validation of IEEE 802.11 MAC protocols.

Nicola Bucciol[1], **Gianluca Marcon**[2], **and William Rizzo**[3]

[1] *nicola.bucciol@studenti.unipd.it*
[2] *gianluca.marcon.1@studenti.unipd.it*
[3] *william.rizzo@studenti.unipd.it*
* *Code repository: https://github.com/geeanlooca/wsn-simulator*

Compiled October 11, 2017

ABSTRACT

In IEEE 802.11 wireless networks, channel access is managed by the Distributed Coordination Function (DCF) mechanism. In order to avoid collisions, each station that wants to transmit senses the channel for a random amount of time: if the channel is sensed free, the transmission begins, otherwise the process is deferred to a later time. This solution is inefficient in case of networks with a large number of users, so a lot of research has been done in order to design new MAC protocols with the aim of satisfying the strong delay and throughput requirements of modern wireless networks. An interesting subset of these new schemes consists of MAC protocols that resolve the channel contention in a constant time. In this work we have implemented a discrete-event simulator with the purpose of comparing the performances of two of these protocols with DCF. We also provide some useful theoretical aspects and considerations on the results.

1. INTRODUCTION

The *Medium Access Control* (MAC) protocol adopted in the IEEE 802.11 standard is called *Distributed Coordination Function* (DCF). According to DCF, every station contends for the wireless channel using the *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) protocol with a Binary slotted Exponential Backoff (BEB). This mechanism tries to avoid packet collisions by randomizing the amount of time for which each station must sense a "free" channel before transmitting. In recent years, with the rise in popularity of wireless-enabled devices such as Smart Phones and wearable devices, as well as the birth of the Internet of Things paradigm, the amount of terminals in a wireless network has grown. It is then mandatory to spend channel resources in an efficient manner, in order to guarantee certain Quality of Service (QoS) requirements even when the amount of users in a network is large. Many studies have been devoted to this cause and a lot of new MAC protocols have been proposed. One branch of protocols investigates the possibility of resolving the contention of the channel in a constant time, using jamming-based techniques, instead of random-length contention schemes like in DCF. One of such protocols was first presented in [1] and later revised in [2], where the authors proposed a new scheme that provides access by resolving the contention between stations in a fixed number of time slots using a jamming-based technique, with the objective of minimizing the packet collision rate and consequently maximizing the network throughput. Since the protocol aims to decide which station has to transmit in a CONstant TIme, they called it CONTI.

The authors of the paper proposed several comparisons between their new protocol and other schemes in the literature, emphasizing its gain with respect to DCF.

Another similar scheme, presented in more theoretical framework, has been proposed by Galtier in [3, 4]. He investigated the performance of his protocol against both DCF and the first version of CONTI [1], guaranteeing that no other constant time slot-based scheme could do better. Since this protocol has not been given a name, we decided to refer to it

with its author's name throughout this report.
The purpose of this work is to review and compare the performance of these three protocols through the implementation of an event driven 802.11b simulator, extracting useful metrics such as collision rate, throughput, delay and fairness, and, whenever possible comparing them to the analytical results.
Summarizing, the report is organized as follows: in Section 2 we describe each protocol in detail, providing the equations that the authors used and the assumptions made in the analysis. Section 3 describes the simulation environment, the algorithms used to extract the parameters of each protocol and the results we obtained. Finally, in Section 4 we summarize what we learned and we propose a way of extending this work.

2. PROTOCOLS OVERVIEW

Before we start illustrating the operating principle of the protocols that we tested, we will briefly list the assumptions that either we or the authors of the papers made. Firstly, we will consider networks in which every station is backlogged, that is, there is always a packet to transmit. This let's us evaluate the *saturation throughput*. In fig. 1 we can see that increasing the offered traffic at the MAC layer, we eventually reach a point in which the throughput stabilizes at a value which is smaller than the maximum achievable.

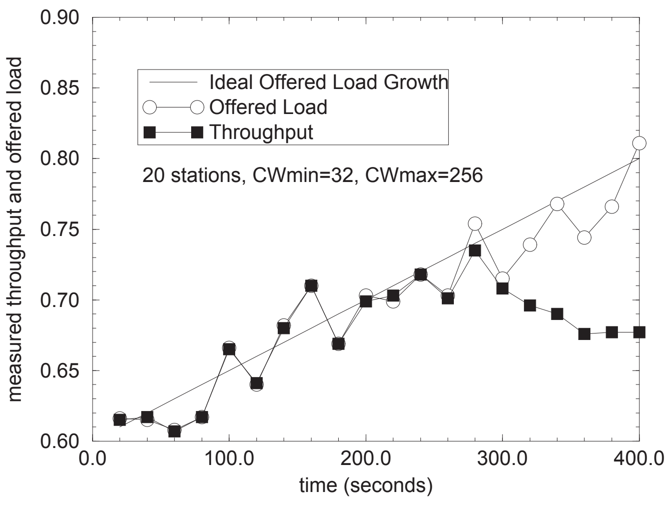

Fig. 1. Measured throughput with slowly increasing offered load for 802.11 DCF [5].

Moreover, we assume that the transmission of a packet fails only if there is a collision, meaning that the channel is ideal, i.e. $P_{bit} = 0$. Also the maximum number of retransmissions is ∞.

We considered the case in which every station is in transmission range of each other. This assumption leads to the absence of the Hidden Terminal problem, which, coupled with the fact that a) that the stations are always backlogged and b) jamming based techniques rely on the ability to hear jamming signals, could lead to a huge reduction of the throughput and high delays.

A. Distributed Coordination Function

The Distributed Coordination Function (DCF) is the MAC scheme used by the IEEE 802.11 [6], and nowadays the most widely used in WLANs. It is a discrete-time Contention Window (CW) technique based on CSMA/CA.
According to the basic access mechanism of DCF, a station that needs to transmit a packet has to monitor the channel activity beforehand. If the channel results idle for a period of time equal to a *Distributed InterFrame Space* (DIFS), the station transmits. However, if the channel is busy the station continues to listen the channel until it is measured free for a DIFS. When this happens, the station waits a number of time slots before transmitting. This number of time slots, called backoff (BO) counter, is randomly chosen from a uniform distribution in the range $[0, CW)$. The value CW is called Contention Window, and depends on the number of failed transmissions in a row by a station. The CW is initially assigned to a preset value, CW_{min} and doubled after each unsuccessful transmission, until it reaches the maximum value equal to CW_{max}. The station decreases the BO counter by one for every time slot the channel is idle, it freezes the BO when a transmission is detected on the channel, and it resumes the BO when the channel is sensed idle again for more than a DIFS. The station transmits when the BO counter reaches zero. A collision occurs when the counters of two or more stations reach zero in the same slot.
The random BO usage represents the Collision Avoidance feature of the protocol, useful to minimize the probability of collision with packets being transmitted by other stations.

As an example let's consider fig. 2. Station A and station B contend the same wireless channel. Station B ends the packet transmission, then it waits for an idle period of time of length DIFS and randomly chooses a backoff counter equal to 8. While Station B is decreasing its BO counter, Station A comes with

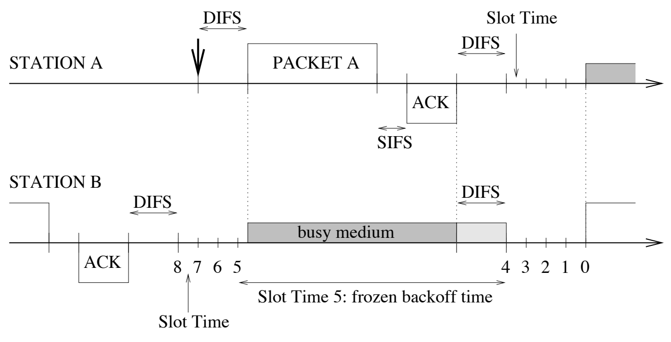

Fig. 2. Instance of the DCF mechanism [5].

its first packet (indicated with the arrow in the time line). Since Station A finds the channel idle for a DIFS it starts transmitting. Station B still has 5 BO slots to spend, but when Station A starts transmitting it freezes its BO counter and waits for the channel to become idle again. Since the CSMA/CA does not rely on the capability of the stations to detect a collision by hearing their own transmission, a positive acknowledgement (ACK) is transmitted by the destination station to signal the successful packet reception. The ACK is transmitted by the receiving station after a period of time called *Short InterFrame Space* (SIFS). The SIFS is designed to be shorter than a DIFS, so that no other station is able to detect an idle channel for a DIFS until the end of the ACK. Station B then resume its countdown after listening and idle channel for a DIFS.

A.1. Theoretical Analysis

A complete and thorough analysis of the DCF mechanism can be found in [5]. The author modeled the protocol with a Markov Chain and was able to obtain the expression for τ, the probability that a station transmits in a generic slot time, and p, the probability of a collision seen by a packet being transmitted on the channel, i.e. the packet collision probability. The two quantities are related to each other through equations 1 and 2.

$$p = g(\tau) = 1 - (1-\tau)^{n-1} \tag{1}$$

$$\begin{aligned}\tau = f(p) &= \frac{2}{1 + W + pW\sum_{i=0}^{m-1}(2p)^i} \\ &= \frac{2(1-2p)}{(1-2p)(W+1) + pW\left[1-(2p)^m\right]}.\end{aligned} \tag{2}$$

The parameter W is simply $CW_{\min}$ and m is such that

$$CW_{\max} = 2^m CW_{\min}. \tag{3}$$

As we can see, the two form a system of nonlinear equations in τ and p, which has been proved to have a unique solution, and hence can be easily solved with numerical techniques. In fig. 3 we picture the two equations and their solution for a few values of n. Once we obtain a solution (p, τ), we can express

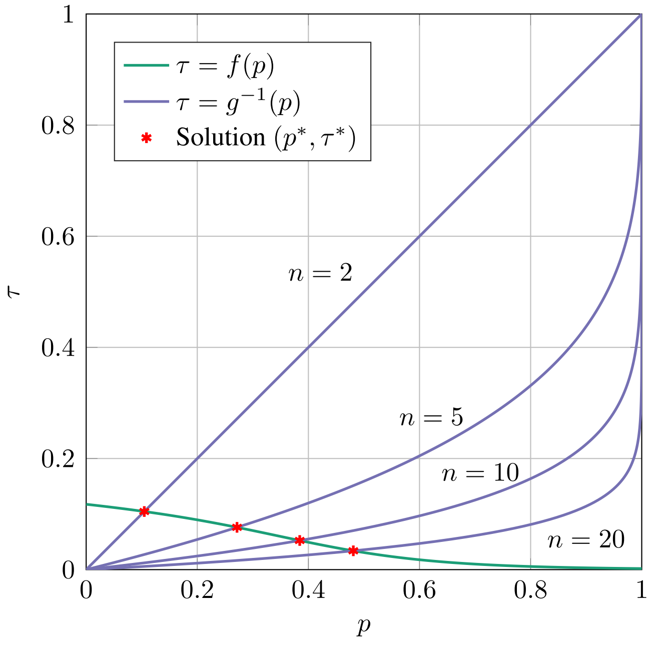

Fig. 3. Solutions of the nonlinear system for determining the packet collision probability, for different values of n.

the probability p_{tr} that there is at least one transmission in a certain contention slot. Since each station transmits with probability τ, we obtain

$$p_{tr} = 1 - (1-\tau)^n, \tag{4}$$

as $(1-\tau)^n$ is the probability that none of the n stations transmit. The collision probability, seen as the probability that the channel contention is not solved successfully, is then given by the complementary probability that only one station transmits given that at least one station transmits:

$$\begin{aligned}p_{coll} &= 1 - \frac{n\tau(1-\tau)^{n-1}}{p_{tr}} \\ &= 1 - \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n}.\end{aligned} \tag{5}$$

B. CONTI

The CONTI protocol is a jamming-based contention resolution protocol initially proposed in [1] and then extended with a more precise and theoretical analysis in [2]. In a generic network with n nodes, CONTI tries to solve the channel contention within a constant number of slots k. At each node is assigned the same probability vector $p : \{p_1, p_2, \ldots, p_k\}$ where

p_i is the probability of choosing signal 1 in the i-th contention slot. During a contention slot, nodes with signal 1 transmit a jamming signal and remain in the contention while nodes with signal 0 listen to the channel. A node associated to signal 0 can experience two situations:

1. it hears the jamming, in this case it is called *preempted* and it leaves the contention;
2. it doesn't hear the jamming and it stays in the contention.

This behavior is summarized in Alg. 1 and an example of contention resolution is illustrated in fig. 4. At the end of the contention phase all the nodes that have not been preempted transmit the frame. It is then clear that a collision occurs only if 2 or more nodes are not preempted in the last contention slot. The value of vector p is optimized to minimize the collision rate while the number of slot k is minimized to reduce the time spent in the contention phase.

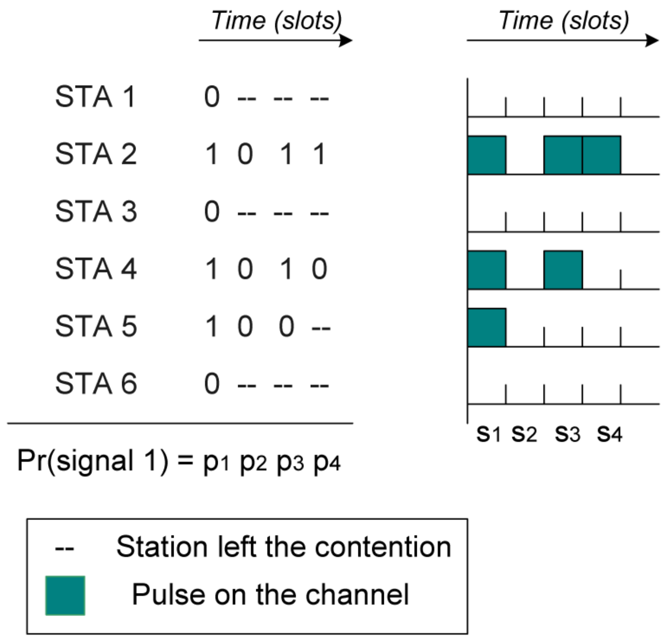

Fig. 4. Contention resolution using CONTI

To ensure the "synchronization" between nodes, before contending, the station must monitor the channel for DIFS seconds of silence.

B.1. Theoretical Analysis

In [2] it was proved that the problem of finding the probability vector p that minimizes the collision rate has the necessary structure to be solved in a *greedy* approach, that is, the choices of the probabilities p_i of emitting a jamming signal in a given contention slot s_i can be optimized individually. This means that at each time slot we want to minimize the number of stations still in contention. The parameter k, which determines the total number of contention slots, will be the minimum number of slots which guarantees a collision rate smaller then a predetermined threshold p^*_{coll}, which was set to 6% by the authors. Intuitively, increasing k will result in a small collision rate. In fig. 5 we see this effect for different values of n and a probability vector with entries equal to 0.5.

Algorithm 1. Contention Resolution with CONTI.

```
 1: retire := false
 2: $i := 0$
 3: while $i \le k$ do
 4:     if (retire = true) then
 5:         defer($t_{slot}$)                      ▷ Station exits contention
 6:     else
 7:         signal ~ Bernoulli($p_i$)              ▷ Choose signal
 8:         if (signal = 1) then
 9:             pulse($t_{slot}$)                  ▷ Emit pulse
10:         else
11:             pulseDetected ← listen($t_{slot}$)
12:             if (pulseDetected) then            ▷ Busy channel
13:                 retire := true                 ▷ Retire
14:     $i \leftarrow i + 1$
```

When u stations are contending at the beginning of a slot characterized by a certain value of p_i, v stations will remain in contention at the end of the slot with probability given by

$$\tau_{u,v}(p_i) = \begin{cases} \binom{u}{v}(p_i)^v(1-p_i)^{u-v}, & 1 \le v \le u-1, \\ (p_i)^u + (1-p_i)^u, & v = u. \end{cases} \tag{6}$$

In fact, u stations remain only if a) every station chooses to send a jamming signal or b) every station chooses to defer. The optimal value of p_i is the one that minimizes the expected number of stations v still in contention. Taking the expectation of eq. 6 we obtain

$$\mathbb{E}[v] = \left[\sum_{v=1}^{u} v\binom{u}{v}p^v(1-p)^{u-v}\right] + u(1-p)^u. \tag{7}$$

This objective function has to be minimized with the use of numerical techniques. For a given u, eq. 7 is minimized to find p^*_i, but the probability that the event $v = \mathbb{E}[v]$ happens is small. For this reason the authors computed the Cumulative Distribution Function (CDF) of v, in order to have a higher confidence

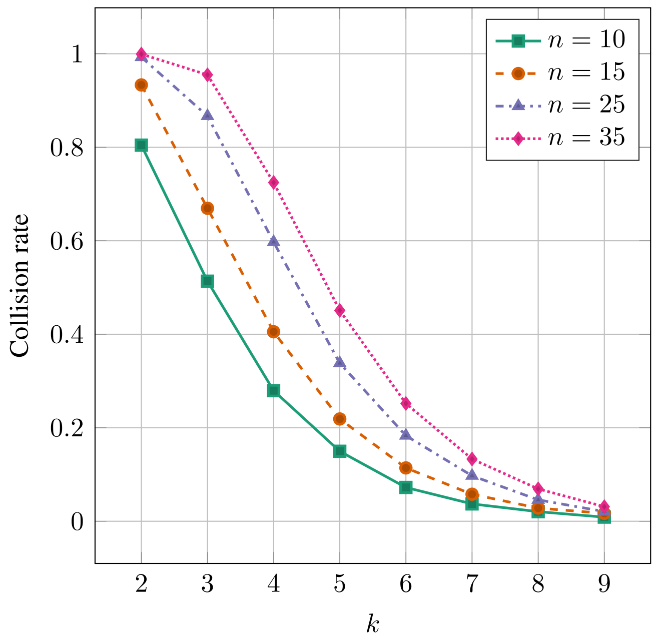

Fig. 5. Impact of the number of contention slots on the collision rate, for different number of stations n.

on the remaining number of stations after a slot. The CDF is defined as

$$\mathrm{CDF}(v) = \theta_{u,v}(p_i) = \sum_{j=1}^{v} \tau_{u,j}(p_i). \qquad \textbf{(8)}$$

This let's us define a value v^* of number of stations such that $\mathrm{CDF}(v) = \mathrm{P}[v \leq v^*]$ is high. The probability vector p is computed as follows:

1. For the first time slot, set $u = n$, since we assume that every station is backlogged, and compute p_1 and the upper bound v_1^*;
2. For the second time slot, assume that $u = v_1^*$, and compute p_2 and v_2^*;
3. Continue until the collision probability is smaller than $p_{coll}^* = 0.06$.

For a given probability vector p, the expression for the collision probability at the end of k slots, when n stations are contending, is given by the following recursive formula

$$\sigma(n,k,p) = \sum_{i=0}^{n-1} \left[\tau_{n,n-i}(p_1)\sigma(n-i,k-1,\pi_2)\right]. \qquad \textbf{(9)}$$

where the base cases are

$$\sigma(1,i,\pi_{k-i+1}) = 1, \qquad 1 \leq i \leq k, \qquad \textbf{(10a)}$$
$$\sigma(v,1,\pi_k) = \tau_{v,1}(p_k), \qquad 1 \leq i \leq k, \qquad \textbf{(10b)}$$

and $\pi_i = \{p_i, p_{i+1}, \dots, p_k\}$ is the suffix subvector of p.

C. Galtier

The protocol proposed by Galtier in [3, 4] is also a slotted jamming-based scheme that solves channel contention in constant time. The operating principle is almost identical to CONTI, with the difference that the probability that a station emits a jamming signal in slot s_i is not predetermined. In fact, this probability depends on the sequence of actions performed by the station during the $i-1$ previous slots.

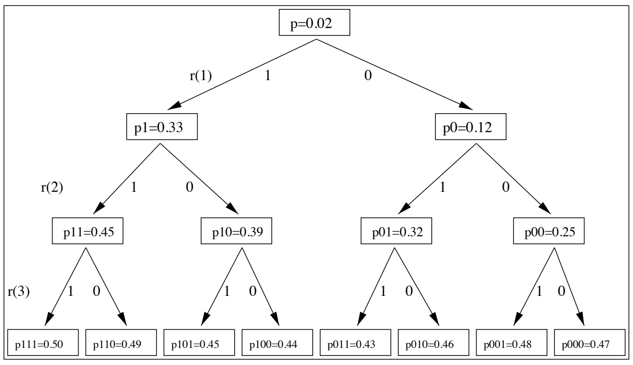

Fig. 6. Probability tree used by Galtier to solve the contention [4].

We illustrate this mechanism with an example. With reference to fig. 6, in the first slot a station chooses to jam with constant probability $p = 0.02$. If this event is verified, in the second slot the jamming probability $p_1 = 0.33$ is found by choosing the left path of the tree, marked by "1". Otherwise, if the station did not emit a signal during slot 0 and did not hear any signal from other stations, the jamming probability $p_0 = 0.12$ is found by choosing the right path of the tree, marked by "0". If instead the station hears a signal from another terminal, it defers and leaves the contention. This procedure is repeated for each time slot. If the number of slots is k, the number of probabilities that we need to store is $2^k - 1$, which is considerably more than what we needed with CONTI.

C.1. Theoretical Analysis

The major difference between CONTI and Galtier's protocol is in the way the optimization problem is formulated. Galtier's approach was to jointly optimize the probability tree instead of following a greedy approach. In addition, the number of contending stations n was not fixed as in [1, 2], but described through a probability distribution q_n. Necessarily, $q_n \geq 0$ and

$$\sum_{n \geq 1} q_n = 1. \qquad \textbf{(11)}$$

The analysis is then carried out by determining how this distribution evolves at each contention slot. This can be done by introducing what the author calls the *generating function* of the distribution of stations, which is defined by

$$f(x) = \sum_{n\geq 1} q_n x^n. \quad \textbf{(12)}$$

In the first slot, the probability of emitting a signal is p, which means that the probability that n stations emit a signal is given by

$$\text{P}[n \text{ stations emit}] = \sum_{i\geq n} q_i \binom{i}{n} p^n (1-p)^{i-n}. \quad \textbf{(13)}$$

Therefore, the generating function of the distribution of the number of stations that emit is

$$\begin{aligned} f_1(x) &= \sum_{i\geq 1} \text{P}[i \text{ stations emit}] \\ &= \sum_{n\geq 1} \sum_{1\leq i\leq n} q_n \binom{n}{i} p^i (1-p)^{n-i} x^i \\ &= \sum_{n\geq 1} q_n[(px+1-p)^n - (1-p)^n] \\ &= f(px+1-p) - f(1-p). \end{aligned} \quad \textbf{(14)}$$

Similarly, the generating function of the distribution of stations that remain silent is given by

$$\begin{aligned} f_0(x) &= \sum_{i\geq 0} q_i (1-p)^i x^i \\ &= \sum_{i\geq 0} q_i [x(1-p)]^i = f((1-p)x)). \end{aligned} \quad \textbf{(15)}$$

The distribution of the surviving stations after the first contention slot is described by $f_1(x) + f_0(x)$. This is because the law of total probability holds. We can generalize these two equations to obtain the distribution after an arbitrary number of slots. Defining w as a word in $\{0, 1\}$, and $w0$ the same word with 0 appended (equivalent with $w1$) and definind p_w and f_w the jamming probability and generating function for step w, the following equations hold

$$\begin{cases} f_{w1}(x) = f_w(p_w x + 1 - p_w) - f_w(1-p_w) \\ f_{w0}(x) = f_w((1-p_w)x), \end{cases} \quad \textbf{(16)}$$

with the convention the convention that $f_\emptyset(x) = f(x)$. The word w represents the series of actions of a station, where 0 indicates that the station was listening while 1 indicates that the station was emitting a signal. After k slots we can then obtain the distribution of surviving stations by summing the generating functions over all possible words w of length $l(w) = k$:

$$g(x) = \sum_{w\,:\,l(w)=k} f_w(x). \quad \textbf{(17)}$$

Recalling that the coefficient of the n-th power of a power series $p(x) = p_0 + p_1 x + p_2 x^2 + \ldots$ is given by

$$p_n = \frac{1}{n!} \frac{\partial^n p(x)}{\partial x^n}\bigg|_{x=0}, \quad \textbf{(18)}$$

we can obtain the success probability ρ as $g'(0)$ since the contention ends successfully only if one station remains, therefore

$$\rho = \sum_{w\,:\,l(w)=k} f'_w(0). \quad \textbf{(19)}$$

The author was able to prove that the previous equation is equivalent to the following

$$\rho = \sum_{i\in\{1,\ldots,m\}} (z_i - z_{i-1}) f'(z_{i-1}) \quad \textbf{(20)}$$

for $m - 1 = 2^k - 1$ real numbers in $(0, 1)$, z_i. What this means is that the success probability can be computed as the Riemann approximation of the integral of f', where f is the generating function of the initial distribution. In this way, the collision probability $1 - \rho$ is given by the approximation error of the integral. To minimize this error we need to carefully select the values of z_i which in turn determine the probabilities p_w. The derivation will be omitted in this report since it's rather long and unintuitive.

Along with a practical method to estimate the optimal probabilities, the author also provided a tight lower bound for the collision rate, which can be asymptotically met by this method if a sufficient number of slots is used. The collision rate is approximated as

$$p_{coll} = 1 - \rho \approx \frac{1}{2m} \left(\int_0^1 \sqrt{f''(t)} dt \right)^2. \quad \textbf{(21)}$$

Given a station distribution q_n and its respective generating function f, we can define $\hat{h}(x) = \sqrt{f''(x)}$ and select the values of z_i as

$$\begin{cases} z_0 = 0, \\ z_j = \dfrac{1}{M} \min\left\{ i \,:\, \dfrac{H(i)}{H(M)} \geq \dfrac{j}{m} \right\}, \\ z_m = 1 \end{cases} \quad \textbf{(22)}$$

with $M \gg m = 2^k$ and H defined by

$$\begin{cases} H(0) = 0, \\ H(i+1) = H(i) + \hat{h}\left(\frac{i+1/2}{M}\right). \end{cases} \quad \textbf{(23)}$$

Finally, the probabilities p_w are given by

$$p_w = \frac{z_{\#(w)2^{k-l(w)}+2^{k-l(w)-1}} - z_{\#(w)2^{k-l(w)}+2^{k-l(w)}}}{z_{\#(w)2^{k-l(w)}} - z_{\#(w)2^{k-l(w)}+2^{k-l(w)}}} \quad \textbf{(24)}$$

where $\#(w)$ is the numerical value of the binary word w and $l(w)$ is its length.

3. SIMULATION RESULTS

A. The Simulator

In order to perform the comparison between the protocols and obtain the output metrics with that parameters we built a Discrete-Event Network Simulator from scratch. The entire simulator is implemented in Java in order to take advantage of the Object-Oriented Programming (OOP) paradigm, which let's us naturally define nodes, events, transmitted packets and the general behavior of the network. It's also a good compromise between performance and ease of use.

The principle behind this type of simulators is that the behavior of the system we are modeling is defined by a series of events which 1) modify the state of the system and 2) trigger the birth of new events. For example, when modeling the packet arrival process at the queue of a node, we might want to define a `PacketArrivalEvent` at a certain time t_0 which adds a new packet to the buffer and then *schedules* a new event of the same type at time $t_0 + t$, where t is a value drawn from the distribution defining the arrival process itself. Another example is the beginning of the transmission of a packet by a node in the network: we might want to set a `BusyChannel` flag to `true`, and then schedule an event determining the end of the transmission.

When an event is scheduled it is added to a data structure called *Priority Queue* (PQ) with a `time` value assigned, which is the instant of time in which the event "fires". This `time` variable is used by the PQ to maintain its elements sorted. Events are continuously removed from the PQ in increasing order of `time` and executed until either a) the PQ is empty or b) the `time` of the last event is greater or equal than the maximum simulation time.

Table 1. 802.11b physical layer parameters.

Parameter	Value
Slot time	20 μs
SIFS time	10 μs
DIFS time	50 μs
CW_{min}	16
CW_{max}	1024
PLCP overhead	192 μs
Transmission rate	11 Mbps
Control rate	1 Mbps

B. Physical Layer

Emulating [2], we decided to use the Physical Layer characteristics of the 802.11b standard. The main parameters are summarized in Tab. 1. The slot time parameter is the same for each protocol, as to guarantee a fair comparison. The PLCP (Physical Layer Convergence Procedure) overhead was added to the transmission time of both frame and ACK. The size of the ACK was not specified in the paper so we took the value given in [6] of 14 bytes, which are transmitted at the control rate of 1 Mbps.

C. Validation and parameter testing

C.1. DCF

To assess the correctness of our simulator, we compare the results for collision rate and packer collision probability against the numerical solution of equations 2 and 5, obtained through the Matlab routine `fsolve`, for different values of n. Looking at fig. 7. We note that the simulation results are slightly below the analytical values, by about 2%. The disparity grows larger for an increasing number of transmitting stations. In [7] a similar behavior has been observed using the ns-2 simulator and a different DCF model, which, however, turned out to be equivalent to the one proposed in [5]. We omitted this curve from the figure as it perfectly overlaps with the model we considered in Section 2.

C.2. CONTI

To determine the optimal parameters for the CONTI protocol, we solved the optimization problem detailed in Section 2. Specifically, we searched for the value of p which minimizes the objective function $\mathcal{J}(p,u) = \mathbb{E}[v]$ of eq. 7. We computed its gradient with respect to v and implemented a Gradient Descent Algorithm (GDA), whose pseudocode is listed in Alg. 2. The parameter γ is called *learning rate*, while ϵ is the minimum improvement to the objective

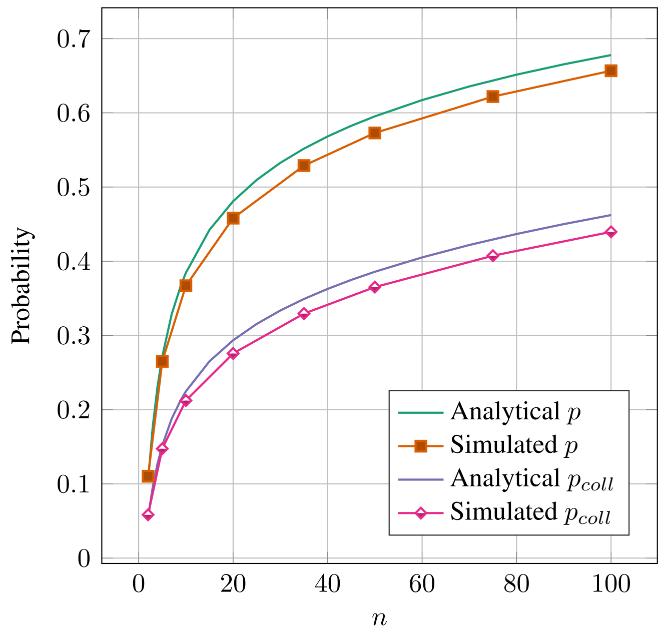

Fig. 7. Probability of collision and probability of failed packet transmission for DCF.

function below which the algorithm stops. To ensure that the algorithm terminates a maximum number of iterations is imposed. We found that $\gamma = 5 \cdot 10^{-5}$ and $\epsilon = 10^{-6}$ are a good compromise between running time and quality of the solution.

Algorithm 2. Gradient descent algorithm for the minimization of $\mathcal{J}(p,u) = \mathbb{E}[v]$.

```
 1: function OPTIMIZE($\gamma$,$\epsilon$,$u$)
 2:     $k \leftarrow 1$
 3:     $p^{(0)} = 0.5$
 4:     while ($k \leq$ max_iter) do
 5:         $p^{(k)} = p^{(k-1)} - \gamma \nabla_p \mathcal{J}(p,u) |_{p=p^{(k-1)}}$
 6:         $\Delta\mathcal{J} = \mathcal{J}\left(p^{(k)},u\right) - \mathcal{J}\left(p^{(k-1)},u\right)$
 7:         if ($\Delta\mathcal{J} < \epsilon$) then
 8:             break                ▷ Stop if improvement is small
 9:         else
10:             $k \leftarrow k+1$
11:     return $p^{(k)}$
```

Looking at fig. 8 we see that the solutions found by the GDA (×) are indeed correct. In fact, we found that the values of p_i's listed in [2] yield an higher value for $\mathbb{E}[v]$. For example, we see that the solution given for $u = 4$ (×) is quite different (in terms of p_i) from the one we found. In Tab. 2 we list the p_i's given by the paper and our solution, along with the corresponding CDF(v^*). What we can observe is that, while our solution yields a lower expected number of remaining stations (values not reported for space constraints), it is not true that P$[v \leq v^*]$ is smaller. We can see this in fig. 9 where we plot the pmd (left) and CDF (right) of $\tau_{10,v}(p_i)$ for p_i given by

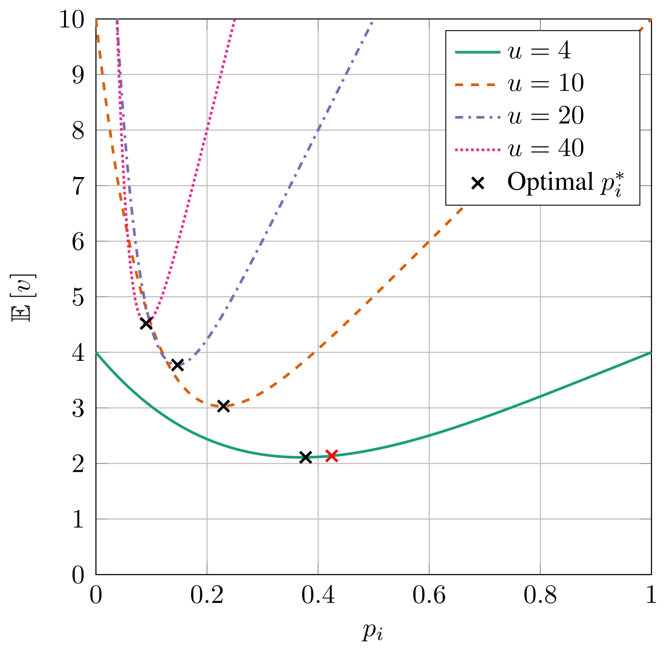

Fig. 8. Expected number of stations v contending at the end of any slot where the probability of emitting a pulse is p_i.

the paper (green) and p_i^* found by the GDA. We see that with our solution we have a higher probability to find 1 or 2 stations after the contention slot, but also an higher probability to find 10 stations. As a consequence, the probability vectors we computed with our values of p_i resulted in a (slightly) higher collision probability! For this reason we decided to simply test the probability vectors provided in the paper. In fig. 10 we plot the resulting collision rate, computed with eq. 9, those available in the paper, and the values we obtained with the simulator. As we can see, the values given in the paper are slightly off, probably due to a numerical issue or a bad implementation of eq. 9.

Since for each value of n there's an optimal p, every node would have to 1) know how many nodes are present in the network or how many are contending and 2) either compute the optimal p_i on the fly or read it from memory. For this reason the authors propose to use a unique p for each n. First, they consider using the value of p obtained for $n = 100$ for every n, and then compute a new optimal value of p, which we call p^{opt}, by maximizing the average time utilization across different scenarios (changing n and the frame size). Solving the optimization problem, both k and p^{opt} are obtained. Given that the number of constraints and the number of variables that we have to optimize is itself a (integer) variable, finding a solution turned out to

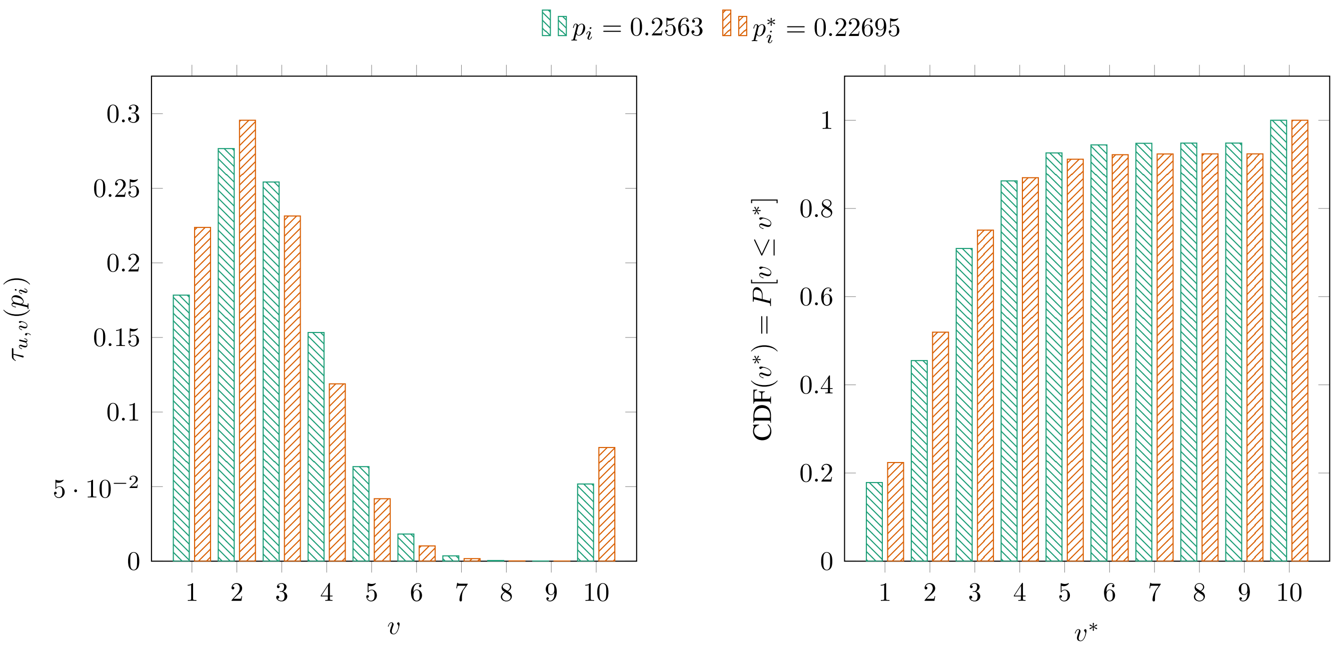

Fig. 9. Comparison between the pmd (left) and CDF (right) of the number of remaining stations v for two different values of p_i.

Table 2. Comparison between the values found in [2] and those we found.

u	v^*	p_i	CDF(v^*)	$\widetilde{p}_i$	$\widetilde{\mathrm{CDF}}(v^*)$
2	1	0.5	0.5	0.5	0.5
3	2	0.4314	0.735	0.4364	0.738
4	3	0.4245	0.857	0.37754	0.83
5	4	0.36715	0.891	0.33603	0.867
10	5	0.2563	0.925	0.22695	0.912
20	9	0.14955	0.960	0.14618	0.957
30	10	0.11945	0.977	0.11091	0.97
40	10	0.09425	0.98	0.09032	0.977
50	10	0.0783	0.981	0.07678	0.98
60	10	0.0673	0.982	0.06707	0.983
70	10	0.064	0.985	0.05973	0.984
80	10	0.05705	0.985	0.05397	0.984
90	10	0.0516	0.984	0.04931	0.985
100	10	0.04715	0.984	0.04546	0.985

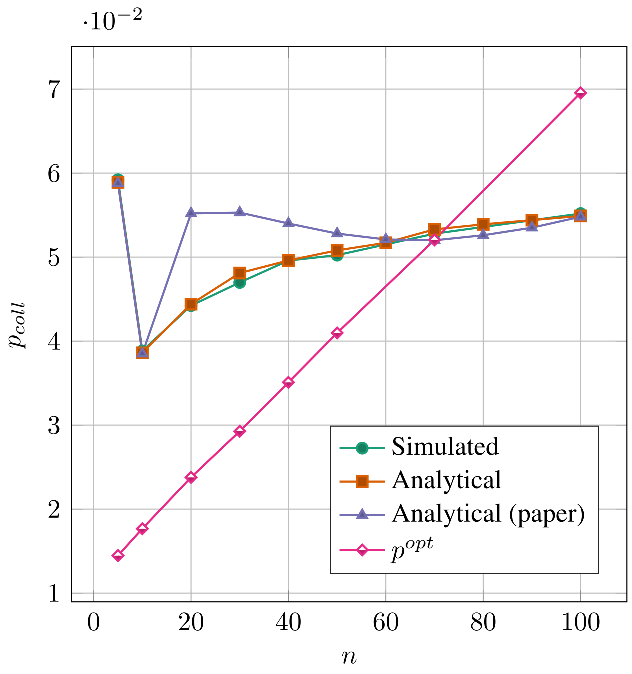

Fig. 10. Comparison of the collision rate for CONTI, for different values of n.

be harder than expected, and we settled for the solution found by the authors, which is $k^{opt} = 7$ and $p^{opt} = \{0.18, 0.31, 0.4, 0.48, 0.48, 0.49, 0.49\}$. The collision rate obtained through simulation is visible in fig. 10. The big gain obtained for $n < 70$ with respect to the other curves is due to the fact that in those cases the number of slots is at most $k = 6$. The cost of having one slot more is offset by the fact that this probability vector can be used for any n. In the remaining part we will always use this probability vector for CONTI.

C.3. Galtier

For this protocol we tried two different approaches. First, we tried and set the station distribution q_n to be equal to 1 only for a specific value of n, since we know that every station is always backlogged. For each different number of contending stations we computed the probabilities p_w. In this way

$$f(x) = x^n \qquad \textbf{(25)}$$

and hence

$$p_{coll} \approx \frac{1}{2m}\left(\int_0^1 \sqrt{n(n-1)x^{n-2}}dx\right)^2 = \frac{2}{m}\frac{n-1}{n}. \quad \textbf{(26)}$$

This approach is highly impractical because:

1. for $k = 7$ we need to compute $2^k - 1 = 127$ probability values and store them in each station for every different n;

2. the values computed for a certain n cannot be used in case there are $n' \neq n$ stations. For example, for $n = 100$, the probability of emitting a signal in the first slot is $p = 0.013393$. When $n' \ll n$, with high probability every station will defer and listen to the channel. In the next slot, they will emit with a probability $p_0 = 0.014027$, which is barely higher than p. Again, with high probability, every station will defer and go onto the next slot. Since $p_{00\cdots0}$ will remain pretty much constant, we would require a high number of slots before a station decides to emit and win the contention. For $k = 6$ or $k = 7$, the collision rate is too high (>60%) for this to be considered a useful approach.

The second approach we took was to use the same distribution q_n as in [4], that is,

$$q_n = \frac{n^{-\alpha}}{\sum_{i=2}^{N} i^{-\alpha}}, \quad \textbf{(27)}$$

which can be tuned by the parameters α and N to take into account both loaded and non-loaded networks. After recomputing the probabilities p_w given in [4] for $\alpha = 0.7$, $N = 100$ and $k = 6$ to test our implementation, we recomputed them setting $k = 7$ in order to have a fair comparison with CONTI. For these parameters we computed an estimated collision probability $p_{coll} = 0.021$ using eq. 21. The values of p_{coll} for the different approaches are depicted in fig. 11. We can see that the approximations given by eq. 21 are fairly accurate. We can also notice the improvement given by setting $f(x) = x^n$ with respect the more generic distribution suggested by the author (for $k = 6$). However, the drawbacks that we previously discussed of using this generating function hardly justify this approach.

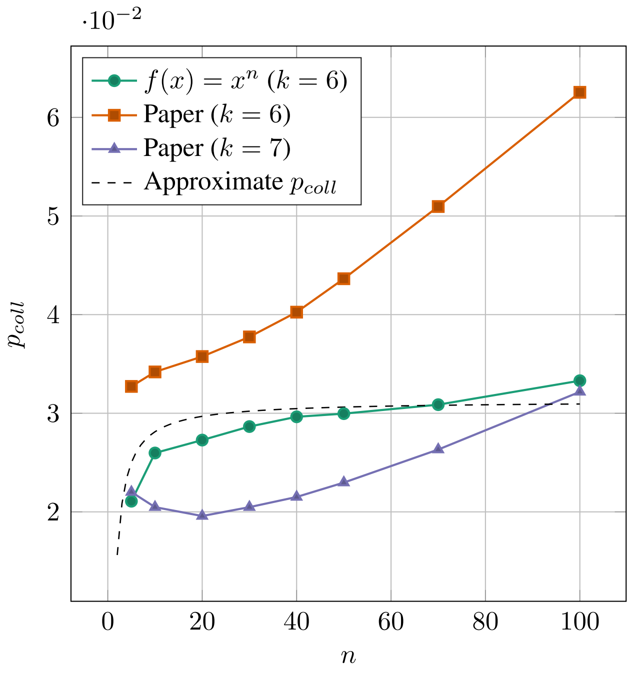

Fig. 11. Comparison of the collision rates for different approaches with Galtier.

D. Collision rate

The first thing we tested was the collision rate of the various protocols, which has been estimated as the ratio between the number unsuccessful contention rounds and the total number of contention rounds. We ran every simulation for 2400 seconds to obtain a stable result.
We have reported the results in fig. 12, running the simulation we have used the optimal parameters for CONTI (the ones suggested in [2]) and those for the distribution in eq. 27 (both $k = 6$ and $k = 7$) for Galtier. From the graph we can notice the gain achieved by the new proposed protocols with respect to the standardized DCF. Also for an high number of stations we the collision rate is always smaller than 10%, while DCF goes up to 40%.

E. Average Number of Slots in a Contention

We have evaluated the average number of time slots that contenting stations spent to decide which one has to transmit. The results are reported in Tab. 3. CONTI and Galtier, depending on the implementation, use a fixed number of time slots to solve the contention. To compute the number of slots in DCF, we have kept track of the minimum backoff counters of the listening stations during the simulation. In fact, the minimum backoff indicates the time slots that we have to wait before the next transmission. This is the reason why the DCF results decrease with an increasing number of stations: with a larger number of

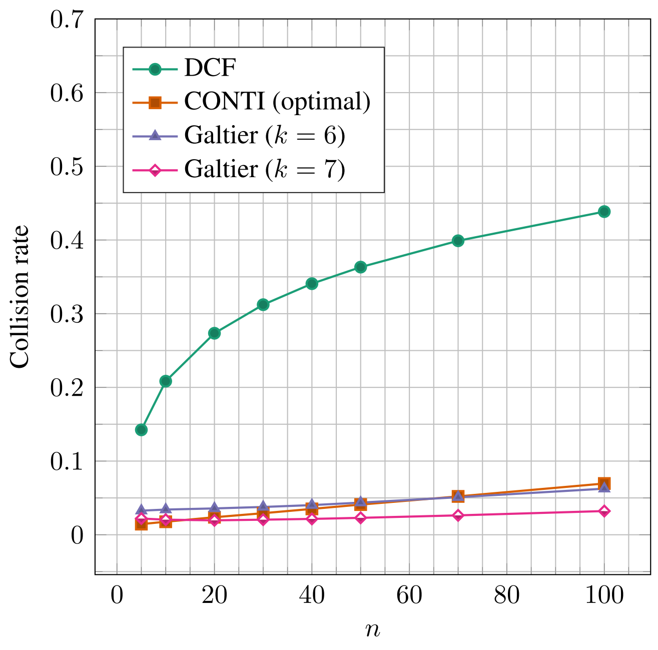

Fig. 12. Collision rate of the 3 protocols as a function of the number of stations in the network.

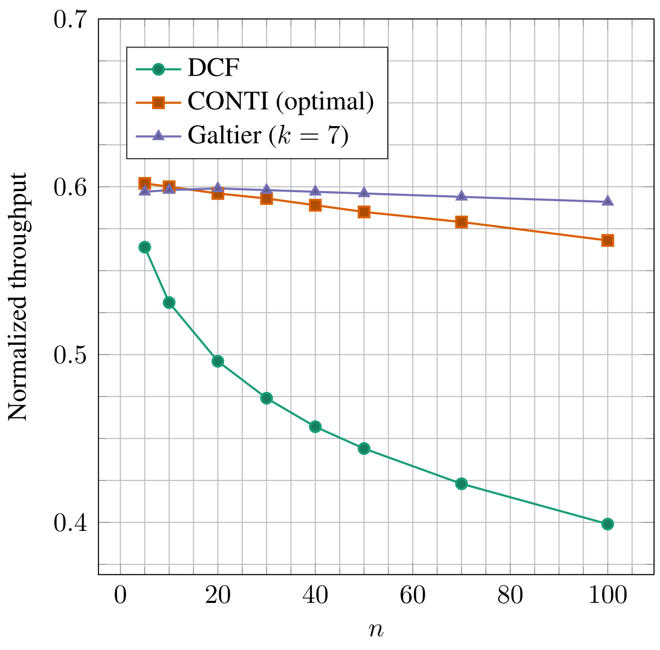

Fig. 13. Throughput comparison as a function of the number of contending stations (1500 bytes).

station there is an higher probability that small BO counters will be picked.
The obtained results are slightly below the ones tabulated in [2].

Table 3. Average Number of Contention Slots

Number of Stations	DCF	CONTI	Galtier
10	2.22	7	7
20	1.85	7	7
30	1.70	7	7
50	1.55	7	7
70	1.46	7	7
100	1.38	7	7

F. Throughput

The normalized throughput was estimated in the simulator as:

$$\eta = \frac{\text{\# of successful TX} \times \text{frame size}}{\text{total simul. time} \times \text{TX rate}} \quad \textbf{(28)}$$

We plot in fig. 14 the throughput behavior as a function of the frame size, with a network having 5 and 50 stations. From the graphs we can notice that the throughput increases with larger frame size. This happens since with larger frames there is more time spent in transmission and less time spent in contention.
For what concerns the relation between the throughput and the number of stations, we have summarized the trends of the previous graphs into the graph in fig.13. We can notice that for a low number of stations (e.g. 5 stations) the performance among the protocols are good and quite similar. However when the number of stations increases DCF shows a huge decay with respect to CONTI and Galtier. This is the well known limitation of DCF protocol. The lower collision rate for Galtier also guarantees an improvement with respect to CONTI, which becomes more substantial as n grows.

G. Delay

Formally, the delay on the MAC layer is defined as the time elapsing between the instant of time when the packet reaches the station's head of the queue and the instant of time when the packet is successfully transmitted [2]. As the number of retransmissions per packet is ∞, this delay is potentially unbounded.

Figure 15 shows that the mean delay experienced by each station increases dramatically with the number of nodes, independently of the protocol used. This is because a) winning the contention against a lot of other contenting stations is less probable and b) the stations commit more collisions, thus new transmissions need to be scheduled and the delay grows. The behavior for CONTI and Galtier is practically identical, with a minor gain for Galtier for large n. The improvement with respect to DCF is substantial: the disparity grows almost to 100 ms for $n = 100$.

The results as a function of the frame size are

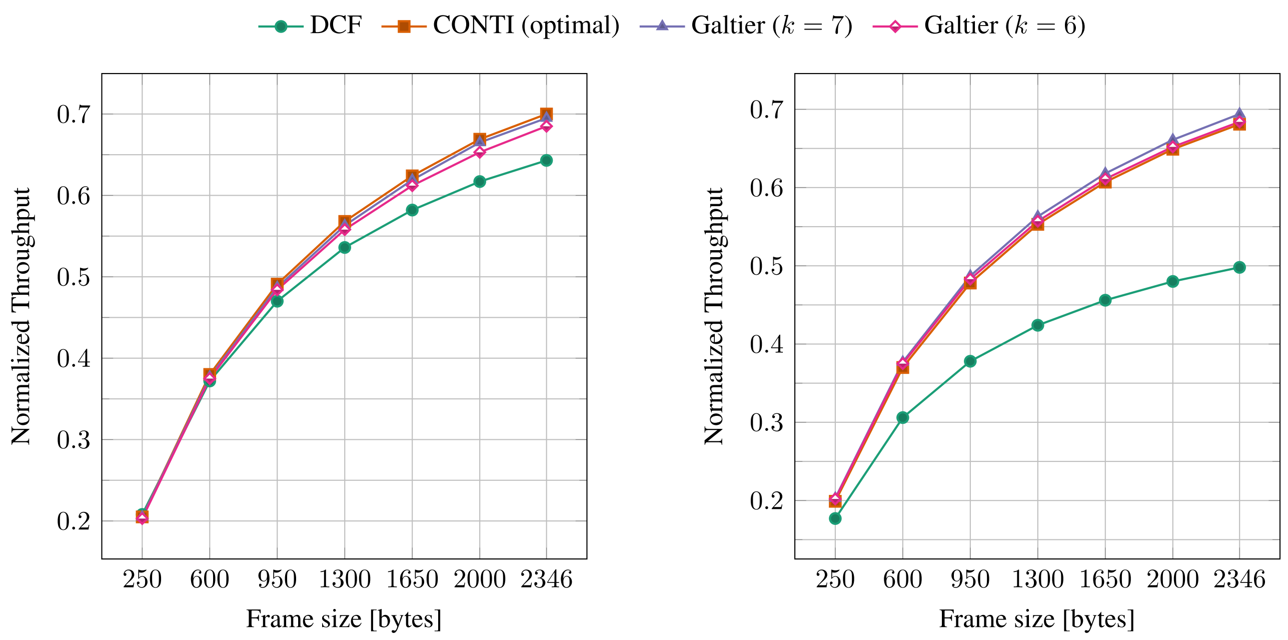

Fig. 14. Throughput as a function of the frame size for $n = 5$ (left) and $n = 50$ (right).

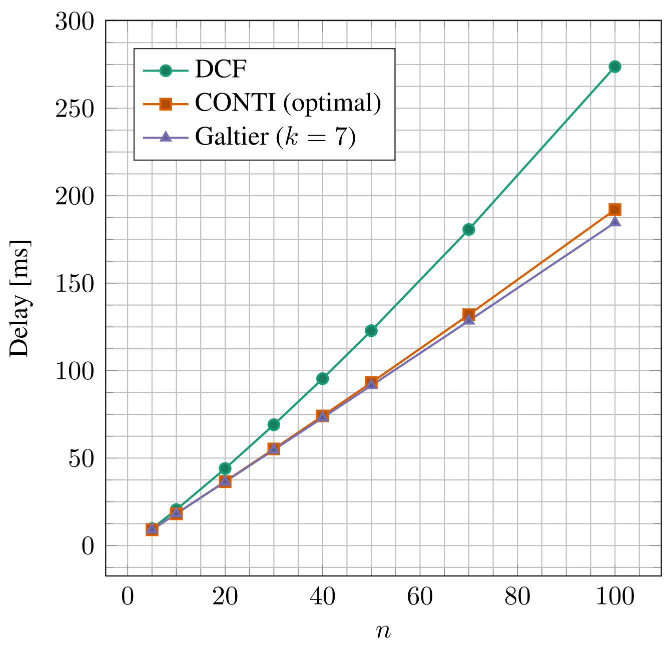

Fig. 15. Delay comparison as a function of the number of nodes (1500 bytes).

shown in fig. 16. As we can expect we get increased delays for the larger packets, since the transmission time will be bigger.

To gain an insight on the distribution of the delays for the three protocols, we plot the histograms in fig. 17 for $n = 20$ and frame size of 1650 bytes. For clarity we selected a logarithmic scale for the y-axis. The width of the bins corresponds to 100 ms. We notice that CONTI and Galtier produced the same distribution. This is not surprising given the similarities between the two. Also, while, DCF is characterized by a (slightly) higher probability of having a delay in the first bin, i.e. smaller than 100ms, we see that the tail of its distribution extends way further than those of CONTI and Galtier, which guarantee delays smaller than 400 milliseconds.

H. Fairness

Whenever there are a set of resources shared by a number of users the Fairness metric describes the "quality" of user allocation.
Although fairness is underlying a qualitative metric, usually it is calculated using the Jain's fairness index [8]. More precisely, *if a system allocates resources to n contending users, such that the i^{th} user receives an allocation x_i, then the Jain fairness index for the system is computed as:*

$$f(x) = \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n \cdot \sum_{i=1}^{n} {x_i}^2} \tag{29}$$

If all users get the same amount, i.e. x_i's are all equal, then the fairness index is 1, and the system is 100% fair. As the disparity increases, fairness decreases and a scheme which favors only a selected few users has fairness near 0.
Authors of CONTI [2] use Jain's index to evaluate an indicator on how fair is the channel contention among the stations. They consider n as the number of active stations and x_i, $i = 1, 2...n$, the ratio between the number of successfully transmitted packets by the station i-th and the total number of successfully transmitted packet by all the stations.

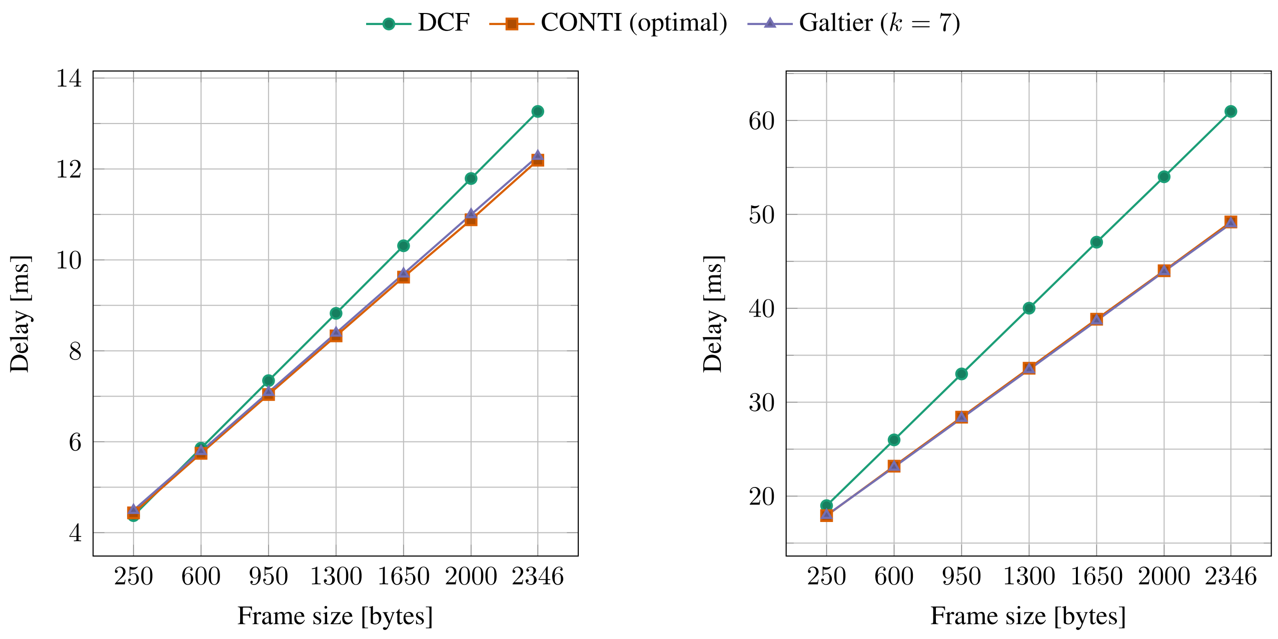

Fig. 16. Delay as a function of the frame size for $n = 5$ (left) and $n = 50$ (right).

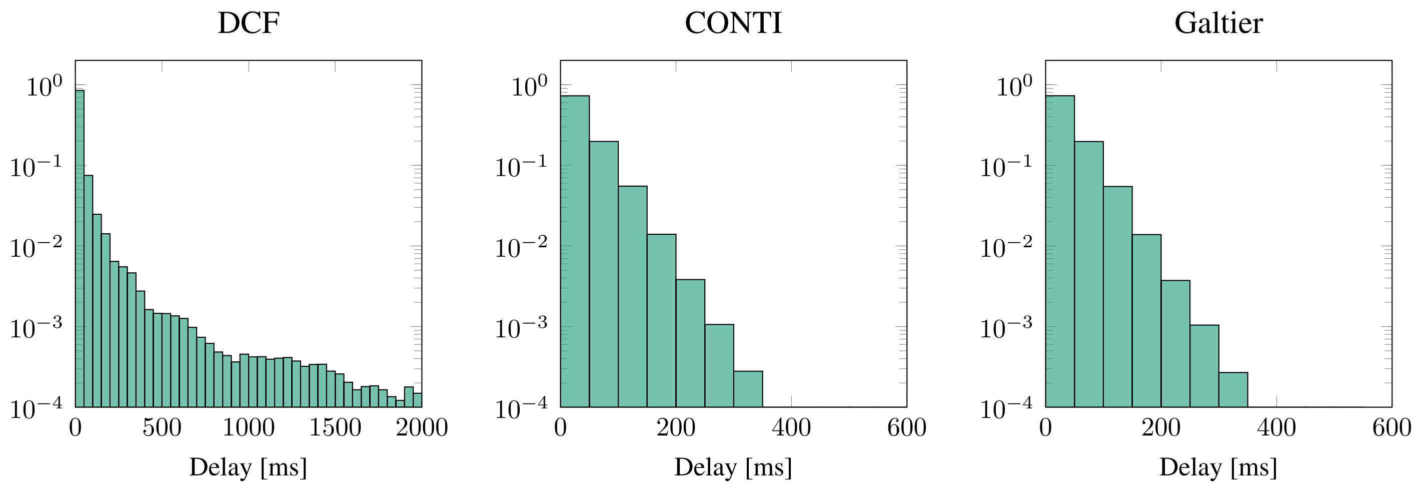

Fig. 17. Delay distribution for $n = 20$ and 1650 bytes of frame size.

The fairness can be affected by several factors, one of them is the time-scale. The time-scale regards the amount of transmission events on which we evaluate the fairness index: a short range of events defines the short-term fairness, instead a larger one defines the long-term fairness.

Short-term fairness is extremely important for obtaining low latency: if a MAC layer presents short-term fairness, each host can expect to access the channel during short intervals, which in turn results in short delays.
To evaluate these alternatives we have implemented a sliding window mechanism over the *transmission trace*, that is a set of the station ID's in the order in which they transmitted. Starting from the first trace element we have calculated the Jain's index in a window of w elements, then we have shifted on the right on the trace and we have evaluated again the fairness index on the window. Averaging the results on traces of 5000 events and varying the window size w from 40 to 1000 elements, which correspond to the short-term and long-term fairness, we have obtained the results in fig. 18.
From the figure is it clear that DCF achieves the lower performance, especially in the short-term, whereas CONTI and Galtier always perform better. According to [9], the main source of short-term unfairness in DCF is the binary exponential backoff algorithm (BEB) used for collision avoidance. The station will double its contention window (CW) in case of collision, and reset its CW to the minimum value CW_{min} in case of success. This leads to unfairness in that

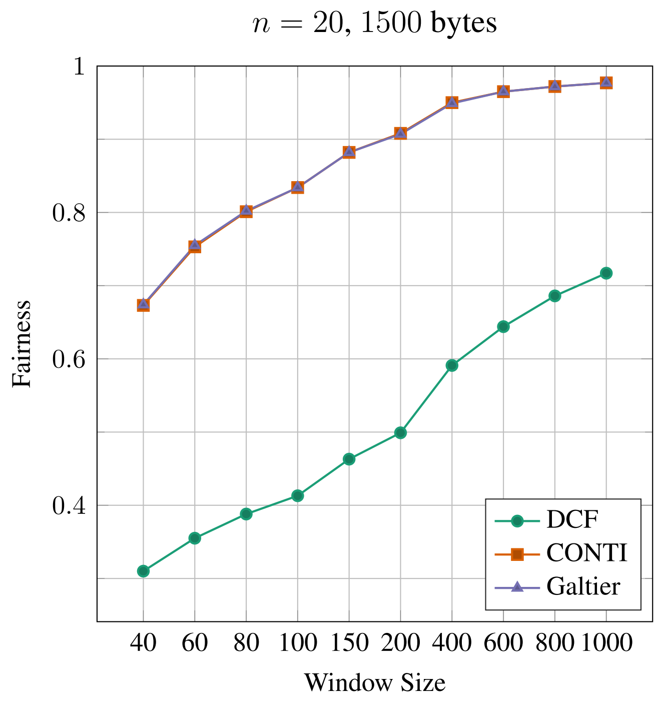

Fig. 18. Fairness comparison as a function of the window length.

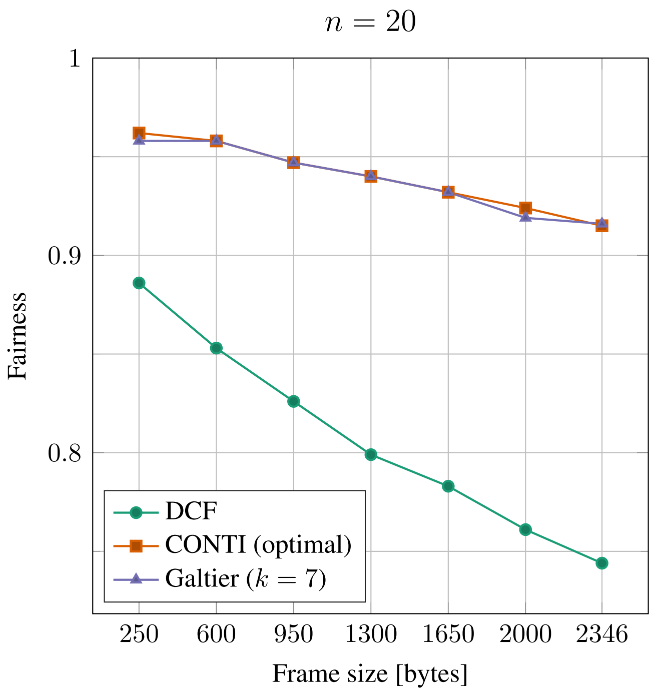

Fig. 19. Fairness comparison as a function of the frame size.

BEB favors last successful user to compete for channel access opportunity again with small CW while punishing other users with increasing delay to access channel. As the window size increase the performance of both the protocols improve, since the fairness is evaluated on a longer simulation time.

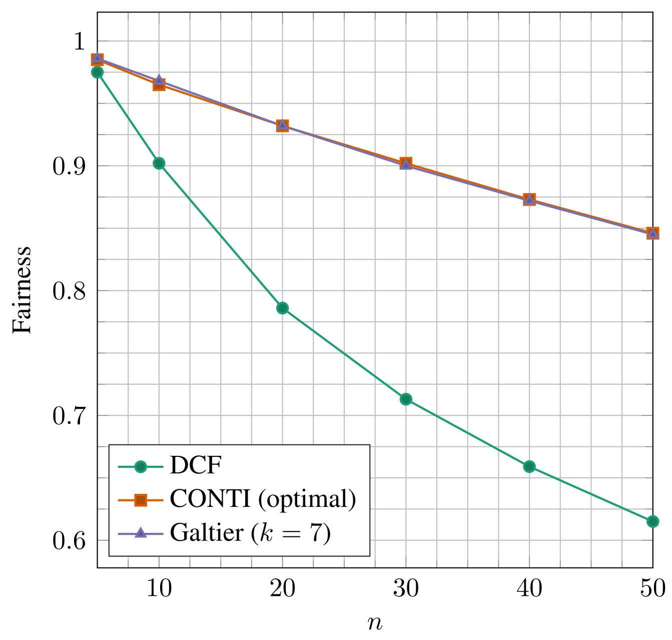

Fig. 20. Fairness comparison as a function of the number of nodes.

Other parameters that influence the fairness are the frame size and the number of stations that contend the channel. For both of them there is an interesting behavior on the short-term, that the authors of [2] estimate on a window of 2 second of simulation time for DCF and 0.5 seconds for CONTI/Galtier. Figure 19 shows the fairness comparison as a function of the frame size. Using small frame sizes rather then longer ones, in a short observation time, leads to an higher fairness since it is possible to transmit an higher number of frames.
The relation of fairness and the number of nodes in the network is reported in fig. 20. An increasing number of stations leads to an unfairness behavior. Caused by the BEB algorithm some stations may experience collisions consecutively while other stations can transmit packets without experiencing any collision. With an higher number of nodes the collision rate increases, thus fairness gets worse.

4. CONCLUSIONS

We compared the performance of two new MAC protocols found in the literature to the 802.11 standard DCF, extracting useful performance metrics such as throughput, delay and fairness. Based on these results we saw the clear improvements brought by these more sophisticated schemes, which are the result of careful analysis and optimization techniques.

During the development of this project we understood the difficulties of building a working network simulator from scratch. We also learned how to validate the simulation results with analytical tools, and to search the literature in order to gain insights on the assumptions and techniques used when they were not specified.

This project can be further extended by testing the protocols in specific scenarios, introducing different packet arrival models, a non-ideal channel, and also testing the effect of mobility when not every node is in transmission range.

REFERENCES

1. Z. G. Abichar and J. M. Chang, “CONTI : Constant-Time Contention Resolution for WLAN Access,” pp. 358–369, 2005.
2. Z. Abichar, S. Member, J. M. Chang, and S. Member, “A Medium Access Control Scheme for Wireless LANs with Constant-Time Contention,” vol. 10, no. 2, pp. 191–204, 2011.
3. J. Galtier, “Tournament MAC with Constant Size Congestion Window for WLAN,” dec 2007.
4. J. Galtier, “Analysis and optimization of MAC with constant size congestion window for WLAN,” in *2007 Second International Conference on Systems and Networks Communications (ICSNC 2007)*, no. Icsnc, pp. 25–25, IEEE, aug 2007.
5. G. Bianchi, “Performance Analysis of the IEEE 802 . 11 Distributed Coordination Function,”
6. “Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements- part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications,” *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, pp. i–513, 2003.
7. T. Sakurai and H. L. Vu, “MAC access delay of IEEE 802.11 DCF,” *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1702–1710, 2007.
8. R. Jain, D. Chiu, and W. Hawe, “A quantitative measure of fairness and discrimination for resource allocation in shared computer systems,” *CoRR*, vol. cs.NI/9809099, 1998.
9. J. Mao, Y. M. Chiu, and S. Leng, “An adaptive transmission control mac scheme for dcf to improve throughput and short-term fairness,” *Second International Symposium on Intelligent Information Technology Application*, pp. 781–785, 2008.