Kalman Filter Estimation of the Number of Competing Terminals in an IEEE 802.11 network

Giuseppe Bianchi, Ilenia Tinnirello
Universita' di Palermo, Dipartimento di Ingegneria Elettrica
Viale delle Scienze, 90128, Palermo, Italy
bianchi@elet.polimi.it, ilenia.tinnirello@tti.unipa.it

***Abstract*— Throughput performance of the IEEE 802.11 Distributed Coordination Function (DCF) is very sensitive to the number n of competing stations. The contribute of this paper is threefold. First, we show that n can be expressed as function of the collision probability encountered on the channel; hence, it can be estimated based on run-time measurements. Second, we show that the estimation of n, based on exponential smoothing of the measured collision probability (specifically, an ARMA filter), results to be a biased estimation, with poor performance in terms of accuracy/tracking trade-offs. Third, we propose a methodology to estimate n, based on an extended Kalman filter coupled with a change detection mechanism. This approach shows both high accuracy as well as prompt reactivity to changes in the network occupancy status. Numerical results show that, although devised in the assumption of saturated terminals, our proposed approach results effective also in non saturated conditions, and specifically in tracking the average number of competing terminals.**

I. Introduction

IEEE 802.11 [1] employs DCF (Distributed Coordination Function) as primary mechanism to access the medium. DCF is a random access scheme, based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol and binary exponential backoff. Several performance evaluation studies of the IEEE 802.11 DCF [2], [3], [4], [5] show that performance is very sensitive to the number of stations competing on the channel, especially when the Basic Access mode is employed. Specifically, performance strongly depends on the number n of "competing" stations, i.e. the number of terminals that are simultaneously trying to send a packet on the shared medium. This information cannot be retrieved from the protocol operation. On one side, DCF does not rely on a privileged station to control the access to the channel. But even considering the existence of an Access Point (AP), as in Infrastructured 802.11 Networks, the information available at the AP is limited to the number of "associated" stations, a number which may be very different from the number of competing stations, i.e. stations that are actually in the process of transmitting packets.

The ability to acquire knowledge of n leads to several implications. It has been shown [6], [7] that, in order to maximize the system performance, the backoff window should be made depend upon n. While, in the standard IEEE 802.11 protocol [1], the backoff parameters were hard-wired in the PHY layer, the idea of adaptively setting the backoff window has been recently taken into consideration in the activities of the 802.11e working group.

Indeed, the knowledge of n has several possible practical implications also in currently deployed 802.11 networks. The 802.11 standard is designed to allow both Basic Access and RTS/CTS access modes to coexist. The standard suggests that the RTS/CTS access mode should be chosen when the packet payload exceed a given RTS threshold. However, it has been shown [4] that the RTS threshold which maximizes the system throughput is not a constant value, but significantly depends on the number n of competing stations. Specifically, as the number of stations in the network increases, the Basic Access becomes ineffective and it results convenient to switch to the RTS/CTS mode even in the presence of short packets. Clearly, this operation requires each station to be capable of estimating n.

A second application scenario of emerging importance occurs when Infrastructured 802.11 networks are arranged in a cellular-like pattern, to provide wireless access in confined high-populated terrestrial areas, called "hot spots", such as convention centers, malls, university campus, residential areas, etc. It appears that, in the very last months, 802.11 is becoming a complementary (or even an alternative) access infrastructure to 3G systems, thus offering new perspectives and market shares for emerging wireless Internet providers. In this cellular-like 802.11 scenario, the estimated knowledge of traffic load and number of terminals sharing an 802.11 cell might effectively drive load-balancing and handover algorithms to achieve better network resource utilization.

In this paper, we propose an efficient tecnique to estimate the number of competing stations in an 802.11 network. Our technique is based on an Extended Kalman filter approach, coupled with a change detection mechanism to capture variations in the number of competing terminals in the network. The estimation methodology builds on the existence of a mathematical relationship between the number of competing stations and the packet collision probability encountered on the shared medium. Such a relation is a straightforward, although originally unforeseen, extension of the analysis carried out in [4]. It is obtained in the assumption of terminals in saturation conditions (i.e. always having a packet waiting for transmission), and in the assumption of ideal channel conditions, i.e. no packet corruption and no hidden terminals and capture [8], [9]. Since this relationship is independent of the access mode adopted, it is suited for application to any DCF access mode scenario, including hybrid Basic-RTS/CTS operation.

PHY	Slot Time (σ)	$CW_{\min}$	$CW_{\max}$
FHSS	50 μs	16	1024
DSSS	20 μs	32	1024
IR	8 μs	64	1024

TABLE I

SLOT TIME, MINIMUM, AND MAXIMUM CONTENTION WINDOW VALUES FOR THE THREE PHY SPECIFIED BY THE 802.11 STANDARD: FREQUENCY HOPPING SPREAD SPECTRUM (FHSS), DIRECT SEQUENCE SPREAD SPECTRUM (DSSS), INFRARED (IR)

While the extension of the work to account for non ideal channel conditions is left for future research activity, we will show that the proposed estimation mechanisms apply also to the non-saturated regime. Specifically, in such conditions, our estimation mechanism allows us to determine the average number of competing terminals (rather than the total number of terminals, as in the saturated regime).

The rest of this paper is organized as follows. In Section II, we briefly review the IEEE 802.11 Distributed Coordination Function. In Section III we derive the relationship between the number n of competing stations and the packet collision probability, and we discuss its application to the problem of estimating n. In section IV we discuss the several limits of a simple exponentially weighted (ARMA) run-time estimation algorithm, and we motivate the need for a more effective estimation approach. This is proposed in section V, where we combine an estimation technique based on an Extended Kalman Filter, with a change detection mechanism designed to estimate abrupt changes in the network utilization, and consequently adjust the Kalman Filter operation. Since the complexity of the proposed Kalman Filter approach is comparable with that of the simple ARMA filter, its adoption has no drawbacks in terms of practical implementation. In Section VI we evaluate the performance of the proposed estimation technique in both saturated and non saturated regime. Finally, concluding remarks are given in Section VII.

II. 802.11 DISTRIBUTED COORDINATION FUNCTION

The IEEE 802.11 Distributed Coordination Function (DCF) is briefly summarized as follows. A station with a new packet to transmit monitors the channel activity. If the channel is idle for a period of time equal to a Distributed InterFrame Space (DIFS), the station transmits. Otherwise, if the channel is sensed busy (either immediately or during the DIFS), the station persists to monitor the channel until is measured idle for a DIFS. At this point, the station generates a random backoff interval before transmitting (this is the Collision Avoidance feature of the protocol), to minimize the probability of collision with packets being transmitted by other stations. In addition, to avoid channel capture, a station must wait a random backoff time between two consecutive new packet transmissions, even if the medium is sensed idle in the DIFS time.

For efficiency reasons, DCF employs a discrete-time backoff scale. The time immediately following an idle DIFS is slotted,

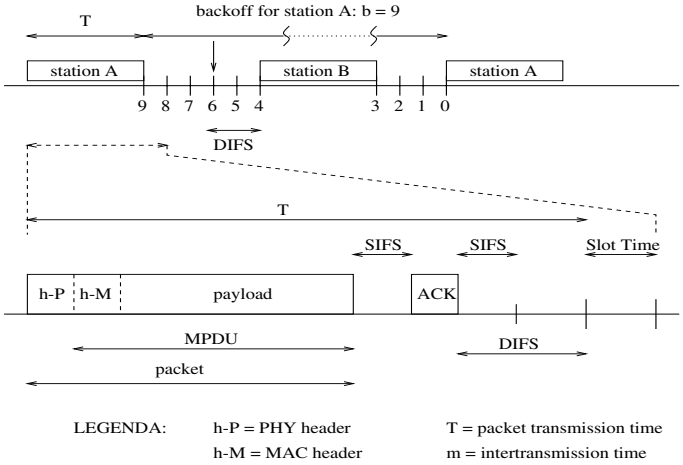

Fig. 1. Example of Basic Access Mechanism

and a station is allowed to transmit only at the beginning of each *Slot Time*.

DCF adopts an exponential backoff scheme. At each packet transmission, the backoff time is uniformly chosen in the range $(0, w-1)$. The value w is called *Contention Window*, and depends on the number of transmissions failed for the packet. At the first transmission attempt, w is set equal to a value $CW_{\min}$ called minimum contention window. After each unsuccessful transmission, w is doubled, up to a maximum value $CW_{\max} = 2^m CW_{\min}$. The values $CW_{\min}$ and $CW_{\max}$ reported in the final version of the standard [1] are PHY-specific and are summarized in table I.

The backoff time counter is decremented as long as the channel is sensed idle, "frozen" when a transmission is detected on the channel, and reactivated when the channel is sensed idle again for more than a DIFS. The station transmits when the backoff time reaches 0.

Figure 1 illustrates this operation. Two stations A and B share the same wireless channel. At the end of the packet transmission, station A waits for a DIFS and then chooses a backoff time equal to 9, before transmitting the next packet. We assume that the first packet of station B arrives at the time indicated with an arrow in the figure. After a DIFS, the packet is transmitted. Note that the transmission of packet B occurs during the Slot Time corresponding to a backoff value, for station A, equal to 4. As a consequence of the channel sensed busy, the backoff time is frozen to its value 4, and the backoff counter decrements again only when the channel is sensed idle for a DIFS.

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 immediately transmitted at the end of the packet, after a period of time called Short InterFrame Space (SIFS). As the SIFS (plus the propagation delay) is shorter than a DIFS, no other station is able to detect the channel idle for a DIFS until the end of the ACK. If the transmitting station does

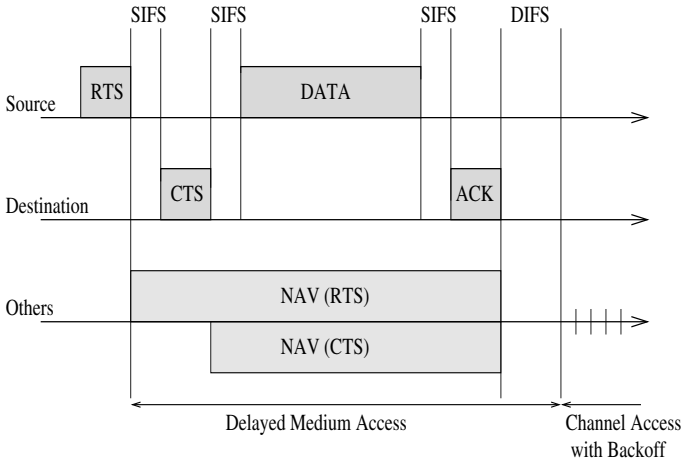

Fig. 2. RTS/CTS Access Mechanism

not receive the ACK within a specified ACK_Timeout, or it detects the transmission of a different packet on the channel, it reschedules the packet transmission according to the given backoff rules.

The above described two-way handshaking technique for the packet transmission is called *Basic Access* mechanism. DCF defines an additional four-way handshaking technique to be optionally used for a packet transmission. This mechanism, known with the name RTS/CTS, is shown in figure 2. A station that wants to transmit a packet, waits until the channel is sensed idle for a DIFS, follows the backoff rules explained above, and then, instead of the packet, preliminarily transmits a special short frame called *Request To Send* (RTS). When the receiving station detects an RTS frame, it responds, after a SIFS, with a *Clear To Send* (CTS) frame. The transmitting station is allowed to transmit its packet only if the CTS frame is correctly received.

The RTS/CTS mechanism provides two fundamental advantages in terms of system performance. First, the RTS/CTS mechanism standardized in 802.11 has been specifically designed to combat the so called problem of *Hidden Terminals* [10], which occurs when pairs of mobile stations result to be unable to hear each other. In fact, the frames RTS and CTS explicitly carry, in their payload, the information of the length of the packet to be transmitted. This information can be read by any listening station, which is then able to update a *Network Allocation Vector* (NAV) containing the information of the period of time in which the channel will remain busy. Therefore, when a station is *hidden* from either the transmitting or the receiving station, by detecting just one frame among the RTS and CTS frames, it can suitably delay further transmission, and thus avoid collision.

Second, the RTS/CTS is proven to effectively increase, in most cases, the throughput performance even in ideal channel conditions. When two colliding stations employ the RTS/CTS mechanism, collision occurs only on the RTS frames, and it is early detected by the transmitting stations by the lack of CTS responses. Since, after the lack of CTS reception, packets are no more transmitted, the duration of a collision is considerably reduced, especially when long packets are involved. The price to pay is a slightly increased transmission overhead (i.e. the RTS/CTS frame exchange) in the case of successful transmissions. A detailed performance discussion can be found in [4].

III. Estimation of the Number of Competing Stations

In this section, we show that, starting from the model proposed in [4], it is immediate to derive a formula that explicitly relates the number of competing stations to a performance figure that can be measured run-time by each station.

The analysis proposed in [4] stems from the observation that the modeling of DCF can be greatly simplified by using a non-uniform discrete time scale, where each slot corresponds either to an empty slot (thus lasting a slot-time σ), or to a transmission or collision slot (e.g. slot 4 in figure 1), where the slot duration corresponds to the (random) duration of a transmission or collision. This approach allows to derive results independent of the access mode considered (Basic, RTS/CTS or a combination of the two), since the access mode employed only affects the duration of the busy slots.

Following [4], we consider a scenario composed of a fixed number n of contending stations, each operating in *saturation* conditions, i.e. whose transmission queue always contains at least a packet ready for transmission. Channel conditions are ideal: no hidden terminals and no packet corruption is considered. For convenience, the exponential backoff parameters are expressed as W and m, where $W = CW_{min}$ and $CW_{max} = 2^m CW_{min}$, i.e. $m = \log_2(CW_{max}/CW_{min})$.

Let p be the probability (called *conditional collision probability*) that a packet being transmitted on the channel collides, and let τ be the probability that a station transmits in a randomly chosen slot time. In the fundamental assumption that, regardless of the number of retransmissions suffered, the probability p is constant and independent at each transmission attempt, it has been shown in [4] that:

1) the probability τ can be expressed as a function of p as:
$$\tau = \frac{2(1-2p)}{(1-2p)(W+1) + pW(1-(2p)^m)} \quad (1)$$
2) the probability p can be expressed as a function of τ and n as:
$$p = 1 - (1-\tau)^{n-1} \quad (2)$$

Substituting τ, as expressed by (1), into equation (2), and solving the equation with respect to n, we obtain:

$$n = f(p) = 1 + \frac{\log(1-p)}{\log\left(1 - \frac{2(1-2p)}{(1-2p)(W+1)+pW(1-(2p)^m)}\right)} \quad (3)$$

This equation is of fundamental importance for the remaining of this paper. In fact, it provides an explicit expression of n versus the conditional collision probability p, and the (known and constant) backoff parameters m and W. Since the conditional collision probability p can be independently

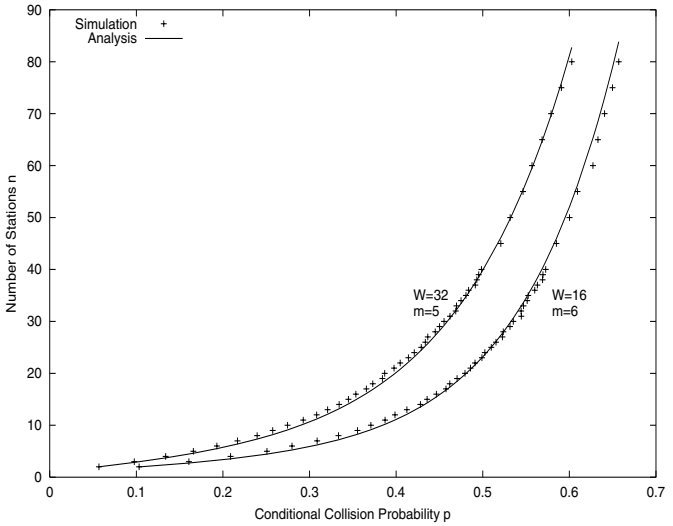

Fig. 3. Number of Stations versus Conditional Collision Probability: analysis vs simulation

measured by each station by simply monitoring the channel activity, it follows that each station can estimate the number of competing stations.

Specifically, each individual station can efficiently measure p as follows. We recall that the conditional collision probability p is defined as the probability that a packet transmitted by the considered station fails. This happens if, in the slot time selected for transmission, another transmission occurs. It might appear that the estimation of p requires each station to count the number of failed transmission and divide such a number for the total transmission attempts. However, it is immediate to understand that a much more efficient procedure is to monitor *all* the slot times (thus significantly increasing the number of samples), regardless of the fact that a transmission attempt is performed or not. Since in each busy slot an eventual packet transmission would have failed, the conditional collision probability can be obtained by counting the number of experienced collisions, C_{coll}, as well as the number of observed busy slots, C_{busy}, and dividing this sum by the total number B of observed slots on which the measurement is taken, i.e.:

$$p = \frac{C_{busy} + C_{coll}}{B} \tag{4}$$

The agreement of formula (3) with respect to simulation results is shown in figure 3. This figure plots the number of contending stations n versus the conditional collision probability p, for two different sets of backoff parameters corresponding to the two different physical layer specifications (table I): FHSS, characterized by $W = 16$ and $m = 6$, and DSSS characterized by $W = 32, m = 5$. Lines represent the analytical relation given in (3), while symbols provide

packet payload	8184 bits
MAC header	272 bits
ACK lenght	112 bits + PHY header
PHY header	128 bits
Channel Bit Rate	1 Mbit/s
Propagation Delay	1 μs
RxTx_Turnaround_Time	20 μs
Busy_Detect_Time	29 μs
SIFS	28 μs
DIFS	130 μs
ACK_Timeout	300 μs
Slot Time (σ)	50 μs

TABLE II
PACKET FORMAT AND PARAMETER VALUES ADOPTED IN THE SIMULATIONS

simulation results[1]. Each simulation point has been obtained considering a constant number n of stations, each in saturation conditions, and measuring the resulting conditional collision probability p. The values of the parameters used in the simulation program are summarized in Table II. The packet size has been set to a constant value. No MSDU fragmentation occurs, so that each MSDU corresponds exactly to an MPDU. Each MPDU is composed of a payload, a MAC header, and a PHY header, whose sizes, shown in table II, are those defined in [1], except for the payload length that we have chosen equal to about half of the maximum value defined in the standard.

The figure shows that the agreement between simulation results (symbols) and analytical results (lines) is remarkable: the difference between simulation and analysis never exceeds 3%.

IV. ARMA Filter Estimation

To provide a run-time adaptive estimation of n, it is sufficient to define a convenient run-time estimation algorithm, so that (depending on the specific application in mind) each station or Access Point, on the basis of channel monitoring, can independently evaluate the time-varying number of competing stations in the network.

In general, run-time estimation is provided by simple mechanisms, such as AR (Auto Regressive) or ARMA (Auto Regressive Moving Average) filters. In particular, we have evaluated the effectiveness of the following estimator:

$$\begin{cases} \hat{p}(t+1) = \alpha\hat{p}(t) + \frac{(1-\alpha)}{q}\sum_{i=0}^{q-1} C_{t-i} \\ \\ \hat{n}(t+1) = f(\hat{p}(t+1)) \end{cases} \tag{5}$$

In this equation, $\hat{p}(t)$ is an ARMA smoothing of the conditional collision probability p. The number of competing stations is estimated from $\hat{p}(t)$ by using the non linear function

[1]Simulation results have been obtained using a custom-made object-oriented event-driven simulator software written in C++. The simulation program reproduces all the details of DCF, as defined in [1]. Simulation results have been obtained using the basic access procedure described in section II (but note that the same curve would be obtained with RTS/CTS). Each simulation run lasts 1000 seconds. To reach saturation conditions, the offered load has been set greater than the per-MS throughput, and a 10 seconds warm-up time has been added at the beginning of the simulation.

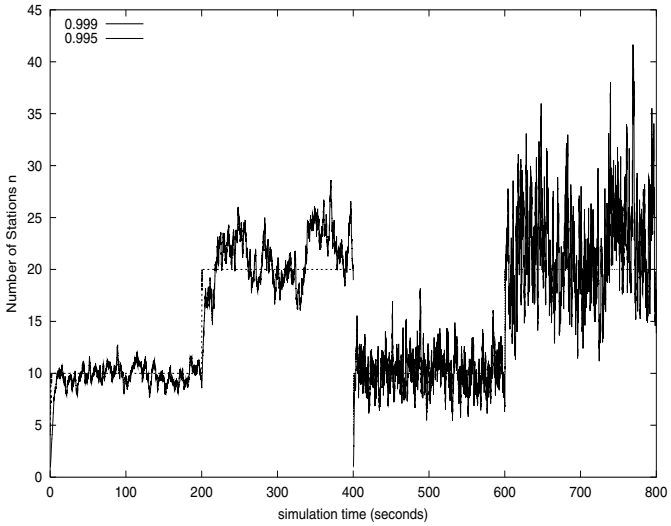

Fig. 4. Run-time ARMA Estimate of the number of contending stations n - $\alpha = 0.999$ and $\alpha = 0.995$

$f(.)$ given in equation (3). The estimation $\hat{p}(t)$ is built upon the computation of the number of busy/idle slots encountered on the channel. Specifically, C_{t-i}, with $i = 0, \cdots, q-1$ are the last q slot samples. C_i is equal to 0 if in the $i-$th slot, either the station does not transmits and sees an empty slot, or the station transmits with success. Conversely, C_i is equal to 1 if the channel is sensed busy during the $i-$th slot, or the station transmits without success[2]. We prefer an ARMA filter rather that a more traditional AR filter (i.e. q=1), since the moving average taken on the last q samples better smooths the fast time scale fluctuations due to the 0/1 gross quantization of each input sample C_{t-i}.

Figure 4 shows the temporal behavior of the running estimate for a reference station. In the figure, two simulation plots of 400 seconds each are reported (the simulation is restarted at time 400s with different parameters). The initial number of stations in the network is set to 10, and this number is doubled after 200 seconds, to simulate an abrupt change in the network state. In the figure, the leftmost plot (seconds 0 to 400) shows the case of $\alpha = 0.999$, while the righmost plot shows the case of $\alpha = 0.995$. In both cases, $q = 10$ has been used in the filter (5). Note that the value q is used to smooth the measurements that are then used to feed the exponentially weighted average, and only marginally affects the filter performance. Conversely, the selection of a suitable value α, i.e. the filter memory, determines the trade-off between estimation accuracy and response time in the case of changes in the number of competing stations.

From the analysis of figure 4, a number of interesting remarks can be drawn. First, we see that, for both values α considered, the estimate rapidly adapts to sudden changes in the network configuration[3]. However, figure 4 shows that the accuracy of the estimation degrades as the number of stations increases. This phenomenon is due to the slope of the curve shown in figure 3, which plots $n = f(p)$ as given by equation (3). As the number of stations increases, the slope increases, too. This implies that the errors in the collision probability estimate are amplified in the evaluation of the number of contending stations. Moreover, we see that, due to the non linearity of the relation $n = f(p)$, the estimation $\hat{n}$ is biased, as it is $f(E[\hat{p}]) \neq E[f(\hat{p})]$. Specifically, as clearly shown by the rightmost plot of figure 4, the average estimated value $\hat{n}$ is greater than the real value n.

This fact is clearly shown in figure 5, which plots the probability distributions $P_p(\hat{p})$ and $P_n(\hat{n})$ of both the collision probability estimate and the resulting network occupancy estimate. The plots have been obtained for the case of $n = 20$ contending stations, for $q = 10$, and for both values α considered in the previous figure 4. The x-axis is graduated in terms of percentual deviation from the nominal value $p = f^{-1}(n = 20)$. The spread of the P_p distributions depend on the filter parameters α and q. The little bias from the value 0 is due to the small mismatch between the analytical relation (3) and the simulation results. While the distributions P_p are almost symmetric, their images P_n through the non linear function $n = f(p)$ given in (3), are very distorted. The distorsion is more and more evident as the spread of the $\hat{p}$ distribution increases (i.e. as the α coefficient decreases). Summarizing, the considered ARMA estimation approach is biased, and an unbiased estimate of n as function of the p estimates is possible only asyntotically, if the p estimates are very accurate, i.e. the filter memory is set to a very large value.

V. Extended Kalman Filter Estimation

In the previous section, we have shown that a simple ARMA filtering approach results unsatisfactory in terms of accuracy/tracking ability tradeoff. We now show that a significantly

[2]Note that, according to equation (3), f(1) is not defined. However, since $\hat{p}(t)$ can be equal to 1 only asintotically, our estimation rule $\hat{n}(t) = f(\hat{p}(t))$ can be pratically applied.

[3] A closer look to the leftmost plot in figure 4 in the neighbor of time 0 and time 200 shows that the tracking performance decreases with an increasing number of stations. This behavior is motivated by the fact that, even if the value α is constant, the time constant of the filter (i.e. the filter memory), when expressed in seconds, is given by:

$$\frac{E[\text{slot time duration}]}{1-\alpha}$$

Therefore, it varies with the number of stations, as the average slot duration depends on the collision probability. In fact, recall that the slot time is variable, and is given by a slot time σ, when the slot is idle, and by the duration of a data packet frame (including overhead and ACK), if the slot time is busy. The average slot duration can be readily obtained from the steady state transmission probability τ, eq. (1) as:

$$E[\text{slot time duration}] = (1-\tau)^n\sigma + n\tau(1-\tau)^{n-1}T_s + \\ + \left[1-(1-\tau)^n - n\tau(1-\tau)^{n-1}\right]T_c$$

where T_s is the average time the channel is sensed busy (i.e. the slot time lasts) because of a successful transmission, and T_c is the average time the channel is sensed busy by each station during a collision. Explicit expressiond for T_s and T_c for both Basic and RTS/CTS access modes can be found in [4]. For an example, with the parameters of table II we obtain an average slot duration of 1.6 ms when n=5, while this value increases up to 3.8 ms when n=25.

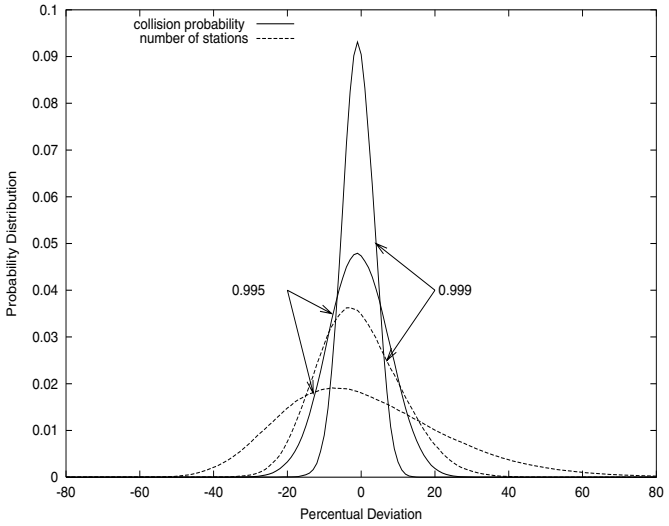

Fig. 5. Probability Density Function of the Estimates $\hat{p}$ and $\hat{n}$

better performance can be achieved by using an Extended Kalman filtering technique. The rationale for using a Kalman filter approach is threefold. First, it allows to adaptively tune the filter memory (say, the factor α in the ARMA filter) to faster track variations in the network occupancy status. Second, it allows us to significantly improve the accuracy of the estimation, by exploiting several additional information available in the model (i.e. state updating laws, and variance of the measurement of p), whereas this information cannot be included in an elementary AR or ARMA approach. Third, the resulting complexity is comparable with that of an elementary ARMA filter, i.e. the use of this filtering technique does not have practical drawbacks.

A. Discrete time state model

Let us first focus on the definition of the temporal steps at which the estimation is updated. Time is discretized in steps of B slot-times (we remark that, as discussed in note 3, the slot-time duration is not constant), where B is a constant pre-defined value.

Within each time step k, the considered MS (or AP) provides a measure p_k of the conditional collision probability based on equation (4), and rewritten here, for convenience, as:

$$p_k = \frac{1}{B} \sum_{i=(k-1)B}^{kB-1} C_i \tag{6}$$

where, as explained in the previous section, for slot-time i, $C_i = 0$ if the slot-time is empty or the station transmits with a success, while $C_i = 1$ if the slot-time is busy or the station transmits without a success. Being p the real (unknown) conditional collision probability suffered on the radio channel, then, for every slot-time i, $Prob(C_i = 1) = p$ and $Prob(C_i = 0) = 1-p$. Therefore, p_k is a random variable with binomial distribution:

$$Prob\left(p_k = \frac{b}{B}\right) = \left(\begin{array}{c} B \\ b \end{array} \right) p^b (1-p)^{B-b} \quad b \in (0, B) \tag{7}$$

The mean value and variance of the measure p_k are obviously $E[p_k] = p$ and $Var[p_k] = p(1-p)/B$

To devise a Kalman Filter estimation technique, we need to provide a state model which consists in: i) a state updating law for the system under consideration, and ii) a measurement model, i.e. the relationship between state and measures.

In our case, the system state is trivially represented by the number n_k of stations in the network at discrete time k. In most generality, the network state evolves as

$$n_k = n_{k-1} + w_k \tag{8}$$

where the number of stations n_k in the system at time k is given by the number of stations at time $k-1$ plus a random variable w_k (in kalman filtering terms, a "noise", hence hereafter referred to as *state noise*) which accounts for stations that have activated and/or terminated in the last time interval. The suitable statistical characterization of the (non stationary) state noise w_k is a key issue in the kalman filter design, and it will be discussed in section V-C. At the moment, just note that we won't assume any model for the arrival/departure of stations, from which the properties of w_k might be derived.

Regarding the measurement model, it is the measure of the conditional collision probability p that each station can carry out via the samples p_k obtained as in equation (6). If, at time k, there are n_k stations in the system, then, the conditional collision probability can be obtained as $h(n_k)$, where h is the inverse function of (3). Note that such inversion is possible, since the direct function is monotone. We can the thus rewrite p_k as:

$$p_k = f^{-1}(n_k) + v_k = h(n_k) + v_k \tag{9}$$

where, based on the previous consideration, v_k is a binomial random variable with zero mean and variance:

$$Var[v_k] = \frac{h(n_k) \cdot [1 - h(n_k)]}{B} \tag{10}$$

Equations (8) and (9) thus provide a complete description of the state model for the system under consideration.

B. The Extended Kalman Filter

Once the state model described by equations (8) and (9) is given, the definition of the Extended Kalman filter is a straightforward application of basic theory (see e.g. [11]). Let $\hat{n}_{k-1}$ be the network state estimated at time instant $k-1$ and P_{k-1} be the corresponding error variance. According to the particularly simple structure of equation (8), at each step, the one-step state prediction is equal to the previous state estimate. Hence, the estimate n_k of the number of stations at time k is computed from the estimate at the time instant $k-1$ by the relation:

$$\hat{n}_k = \hat{n}_{k-1} + K_k z_k \tag{11}$$

In this relation, z_k is the innovation apported by the k-th measure, given by

$$z_k = p_k - h(\hat{n}_{k-1}) \tag{12}$$

where we have applied also in this case the property that the one-step state prediction is equal to the previous state estimate. In equation (11) K_k is the Kalman gain, given by

$$K_k = \frac{(P_{k-1} + Q_k)h_k}{(P_{k-1} + Q_k)h_k^2 + R_k} \tag{13}$$

In the above Kalman gain computation equation, the following symbolism has been adopted:

- Q_k is the variance of the random variable w_k, i.e. the state noise introduced in the state updating law (8). The values adopted for Q_k will be discussed in section V-C.
- R_k is the variance of the measure p_k, i.e., with reference to equation (9), it represents the estimated variance of the random variable v_k, obtained from equation (10) by replacing the actual state n_k with $\hat{n}_{k-1}$ (being this estimate coincident with the one-step predicted state itself). Summarizing:

$$R_k = \frac{h(\hat{n}_{k-1}) \cdot (1 - h(\hat{n}_{k-1}))}{B}$$

- h_k is the sensitivity of the measurement, linearized around the state estimation $\hat{n}_{k-1}$. Coefficient h_k is computed by taking the derivative

$$h_k = \left. \frac{\partial h(n)}{\partial n} \right|_{n=\hat{n}_{k-1}}$$

Finally, the error variance of the new estimate is also recursively computed as:

$$P_k = (1 - K_k h_k)(P_{k-1} + Q_k) \tag{14}$$

Regarding the initial conditions, quick convergence is guaranteed when the initial error variance P_0 is set to a large value (in our numerical results, we have used $P_0 = 100$). In these conditions, the initial estimate of the state is not relevant, and can be set to any value (we used $n_0 = 1$).

C. Selection of State Noise statistics

In order to complete the design of the Extended Kalman Filter, it remains to specify the statistics of the state noise process w_k used in the state update equation (8). In several applications of the Kalman filter, it is generally assumed that w_k is a stationary process with a given constant variance Q. The tuning of the Kalman filter is then performed by appropriately selecting this variance.

However, this approach is quite simplicistic, as it leads again to the issue (discussed for the ARMA case) of trading estimation accuracy with tracking ability. In fact, high values for Q allow to quickly react to state changes, but imply a reduced accuracy in the estimation, i.e. high error variance P_k. Conversely, low Q values give accurate estimates in stationary conditions, but very slow transient phases when abrupt state variations occur.

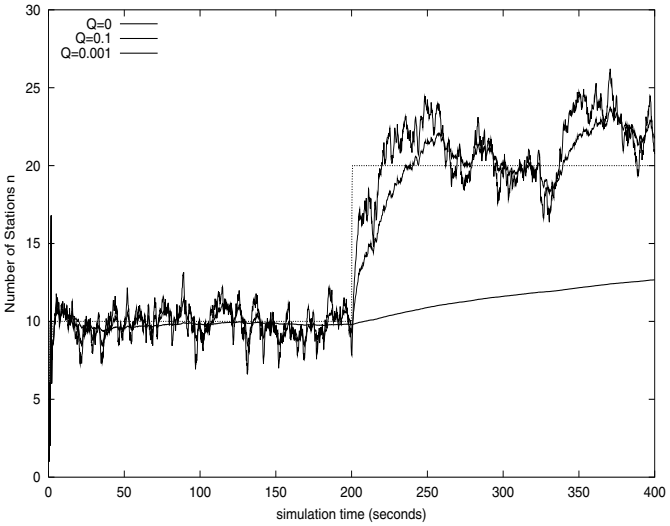

Fig. 6. Tracking ability of the filter varying the variance of the state noise

The unsatisfactory performance of such a constant noise variance approach is quantitatively shown in figure 6, which compares the performance of the Kalman filter for three different values Q, namely $Q = 0$, $Q = 0.1$ and $Q = 0.001$. The simulation scenario is the same adopted for figure 4: initially, 10 competing stations are considered, while, at time 200 seconds, additional 10 stations abruptly activate. The figure shows that the only case in which the transient time is kept in the order of just a few seconds is the case $Q = 0.001$. However, the price to pay is the occurrence of non marginal fluctuations (order of 20% error) in the estimation. The case of $Q = 0$ is also interesting. When the number of terminals is constant, the estimation is impressively accurate. However, in such a case, the resulting state update relation of equation (8) forces the estimation to remain, in practice, stuck to the initial value, which is a serious problem when, as in the test case shown in the figure, the number of stations varies.

The above considerations, combined with the flexibility of the state model (8) which does not require w_k to be a stationary process, suggest that a better approach consists in selecting a time-varying noise variance Q_k. In particular, Q_k should be set to 0, or at least to a very small value, when the number of competing stations in the network appears to remain constant. Conversely, if it appears that the network state has changed, it should be set to a possibly large value Q: it is sufficient to set $Q_k = Q$ for just a single time instant k, as the sudden increase in the noise variance is propagated in the estimation error variance P_k through equation (14). In the next section, we show how this operation can be automated.

D. Change Detection Mechanism

To automate the selection of the state noise variance Q_k, we have associated to the Kalman filter a second "change detection" filter, devised to estimate whether a state change has occurred. A change in the network state can be detected

by analyzing the innovation process z_k, defined in equation (12). If the network occupancy state is constant, the innovation process z_k is a white process with zero mean. Conversely, if the network state changes, the process z_k will move away from its zero mean value.

Among the several available statistical tests, we have implemented a change detection filter based on the CUSUM (CUmulative SUMmary) test [12], which is very intuitive to derive and very effective in terms of simplicity and performance. The CUSUM test is based on two filtered versions g_k^+ and g_k^- of the innovation process z_k. For convenience, we use a related process s_k which represents the innovation process normalized with respect to its standard deviation[4], i.e.:

$$s_k = \frac{z_k}{\sqrt{(P_{k-1} + Q_k)h_k^2 + R_k}} \tag{15}$$

The samples g_k^+ and g_k^- are constructed from the input values s_k as follows:

$$\begin{cases} g_k^+ = \max(0, g_{k-1}^+ + s_k - v) \\ \\ g_k^- = \min(0, g_{k-1}^- - s_k + v) \end{cases} \tag{16}$$

In these equations, v, called "drift parameter", is a filter design parameter. The smaller v is, the more sensible the test results to fluctuations of the process s_k. As initial conditions, $g_0^+ = 0$ and $g_0^- = 0$. If a change in the network state occurs, the magnitude of one between g_k^+ or g_k^- tends to increase unlimitedly. For example, suppose that a new station activates. Then, the collision probability predicted by the current state estimate $\hat{n}_k$ results lower, in average, than the measured one. Therefore, the mean value of the normalized innovation s_k (equations (15) and (12)) becomes positive, and the process g_k^+ starts diverging when such a mean value becomes greater than v. Conversely, divergence occurs for g_k^- when there is a station departure. Hence, an additional CUSUM test parameter, called "alarm threshold" h is defined, i.e. the change detection filter sends an alarm when $g_k^+ > h$ or $g_k^- < -h$. After an alarm, both sequences g_k^+ and g_k^- are restored to the value zero. The greater the value h, the lower the probability that a false alarm is detected, but the longer is the time to detect a state change. In this paper, we have set the parameters h and v to be constant values; indeed, we recall that techniques are available [12] to automate the setting of these test design parameters (e.g. to match a given false-alarm rate and change detection delay).

Alarms coming from the CUSUM test are used to adaptively set the variance Q_k of the noise w_k:

- when the change detection filter does not detect a state change (i.e. no alarm arrives at time k), $Q_k = 0$. This allows to use the new measure p_k (more precisely, the innovation z_k) to increase the accuracy of the former estimation.
- Conversely, upon an alarm generated at time k, the value Q_k is set (for the instant of time k only) to a sufficiently large constant value Q (as discussed in the next section, this parameter marginally affects the estimator performance). This represents a noise impulse in the state update equation (8), which allows the Kalman filter to "move away" from the former estimate and therefore to rapidly converge to a new estimate.

[4]The rationale behind using the normalized innovation process s_k instead of the innovation process z_k is that, in this manner, the design parameters v and h of the CUSUM test can be kept constant. If the innovation sequence z_k were used, then the CUSUM test parameters should have been configured to explicitly depend on the estimated network state and on the related error variance.

VI. Performance Evaluation

The effectiveness of the proposed Kalman filter estimate is demonstrated in figure 7. The configuration parameters for the change detection filter are $v = 0.5$ and $h = 10$. Upon a change detection alarm, the state noise variance is set to $Q = 5$, while it is set to 0 when no alarms occur. In this figure, we have simulated a scenario in which the number of stations in the network increases in steps (1, 2, 3, 5, 10, 25 and 15 stations). Although unrealistic, this scenario allows to prove that our proposed estimation technique is able to track abrupt variations in the network state, while keeping a very high level of accuracy in the estimation. The alarms coming from the change detection filter are also reported in the figure, as small impulses on the x-axis. They demonstrate that the parameter value Q is not very critical in terms of filter performance. In fact, as shown in the step from 10 to 25 stations (simulation time 350s) and from 25 to 15 stations (simulation time 450s), it may happen that the value $Q = 5$ adopted is too small to allow the Kalman filter to capture a large variation in the network state. Indeed, this is not a critical problem, as the change detection filter will eventually send a second alarm after a few seconds (so that the Kalman filter convergence to the state change results always possible). For comparison purposes, figure 7 reports results for two ARMA filters with $\alpha = 0.9995$ and $\alpha = 0.999$. Both the ARMA filters are not satisfactory: the first in terms of tracking ability, the second in terms of estimation accuracy.

Our model, as well as all simulation results up to now, has been developed in the assumption of saturated conditions, i.e. all stations in the network are assumed to always have a packet to transmit in their transmission buffer. Figure 8 shows the behavior of the proposed estimation technique when the terminals are not in saturated conditions. The Kalman filter parameters are the same as that used in the previous figure, while the ARMA filter has a memory factor $\alpha = 0.999$. In particular, figure 8 reports a simulation run for a network scenario of 20 stations. Packets arrive to each station according to a Poisson process, whose rate is initially set to be lower than the saturation throughput. The arrival rate is subsequently increased so that, at the end of the simulation time, all stations are in saturation conditions.

In the non saturated regime, the number of terminals attempting to transmit a packet (i.e. the number of competing terminals) shows fairly large and fast fluctuations, as highlighted by the dashed plot in figure 8. Neither our Kalman filter, nor an ARMA filter, are able to follow these fast fluctuations. Instead, they both appear to estimate the *average*

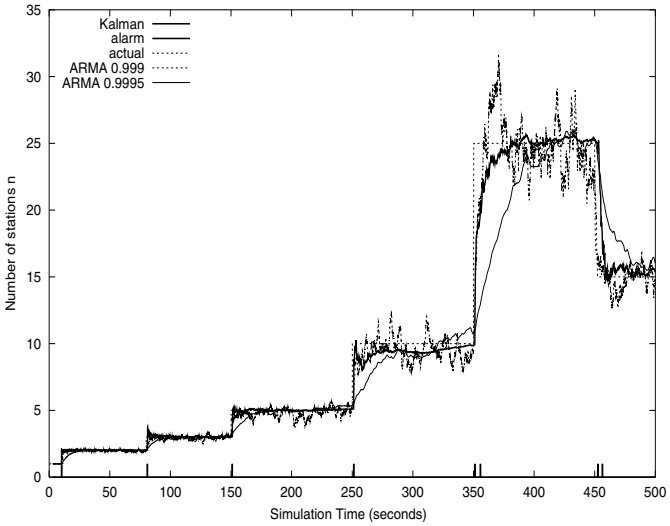

Fig. 7. Kalman filter estimation in saturated network conditions

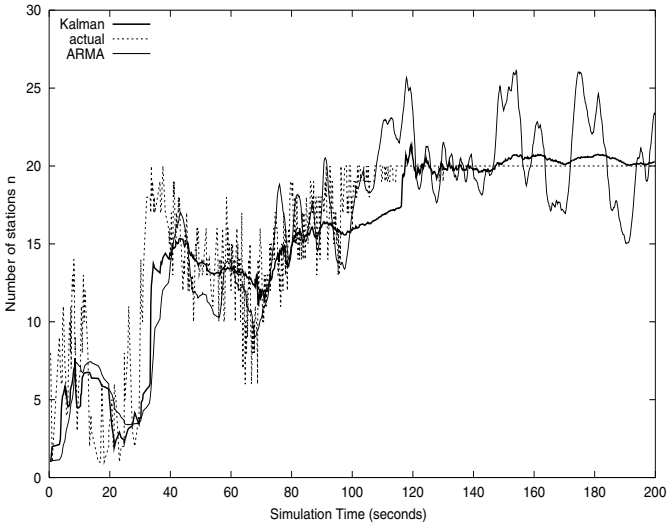

Fig. 8. Kalman filter estimation in non saturated network conditions

number of competing stations. In other words, the proposed estimation technique, devised in saturation assumption, appears to apply also to non saturated conditions, provided that the estimation target becomes the average number of competing stations (rather than the total number of stations in saturation conditions).

By comparing the Kalman estimator with the ARMA one, we see that our proposed mechanism appears to provide a smoother estimation in both saturated and non saturated regimes, although the different level of accuracy can be fully appreciated only in the saturated regime.

VII. Conclusions

This paper has discussed the problem of estimating the number of competing stations in an 802.11 Distributed Coordination Function Wireless LAN. The base for the estimation is provided by a numerically accurate closed form expression that relates the number of competing stations with the probability of a collision seen by a packet being transmitted on the channel by a selected station. By independently monitoring the transmissions eventually occurring within each slot-time, each station is in the condition to estimate, thsough this relation, the number of competing terminals.

We have then evaluated the effectiveness of a practical estimation technique based on ARMA filtering. We have found that such an approach has several drawbacks. First, as expected for all ARMA filters, estimation accuracy is traded off with tracking ability of the filter. Moreover, we have shown that, in our specifical problem, the ARMA estimation results to be biased, due to the strong non linearity of the expression that relates the number of terminals to the collision probability.

Hence, we have proposed a better approach, based on an extended Kalman filter estimate, in conjunction with a change detection mechanism devised to track variations in the network state and accordingly feed the Kalman filter.

Numerical results show that, although devised in the assumption of saturated terminals, our proposed approach results effective also in non saturated conditions, and specifically in tracking the average number of competing terminals.

References

[1] IEEE Standard 802.11 - 1999; Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; November 1999.

[2] B. P. Crow, I. Widjaja, J. G. Kim, P. T. Sakai, “IEEE 802.11 Wireless Local Area Networks”, IEEE Commun. magazine, September 1997, pp. 116-126.

[3] J. Weinmiller, M. Schlager, A. Festag, A. Wolisz, “Performance Study of Access Control in Wireless LANs IEEE 802.11 DFWMAC and ETSI RES 10 HIPERLAN”, Mobile Networks and Applications, Vol. 2, 1997, pp. 55-67.

[4] G. Bianchi, “Performance Analysis of the IEEE 802.11 Distributed Coordination Function”, IEEE Journal of Selected Areas in Telecommunications, Wireless series, Vol. 18, no. 3, March 2000, pp. 535-547.

[5] Y.C. Tay, K.C. Chua, “A capacity analysis for the IEEE 802.11 MAC protocol”, ACM/Baltzer Wireless Networks Vol. 7, No. 2, March 2001, pp. 159-171.

[6] G. Bianchi, L. Fratta, M. Oliveri, “Performance Evaluation and Enhancement of the CSMA/CA MAC Protocol for 802.11 Wireless LANs”, Proc. PIMRC 1996, October 1996, Taipei, Taiwan, pp. 392-396.

[7] F. Cali’, M. Conti, E. Gregori, “Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit”, Trans. on Networking, Vol. 8, No. 6, Dicember 2000, pp. 785-799.

[8] K. C. Huang, K. C. Chen, “Interference Analysis of Nonpersistent CSMA with Hidden Terminals in Multicell Wireless Data Networks”, Proc. of IEEE PIMRC ’95, Toronto, Canada, September 1995, pp. 907-911.

[9] H. S. Chhaya, S. Gupta, “Performance Modeling of Asynchronous Data Transfer Methods of IEEE 802.11 MAC Protocol”, Wireless Networks, Vol. 3, 1997, pp. 217-234.

[10] L. Kleinrock, F. Tobagi, “Packet Switching in radio channels, part II - the Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution”, IEEE Trans. Comm., Vol. COM-23, No. 12, December 1975, pp. 1417-1433.

[11] M. S. Grewal, A. P. Andrews, ”Kalman Filtering: Theory and Practice Using Matlab”, John Wiley & sons, Ltd, 2001

[12] F. Gustafsson, ”Adaptive filtering and change detection”, John Wiley & Sons, Ltd, 2000.