Performance Analysis of IEEE 802.11 DCF: Throughput, Delay, and Fairness

Zhifei Li
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218 USA
Email: zli19@jhu.edu

Amitabha Das, Anil K. Gupta
School of Computer Engineering
Nanyang Technological University
Singapore, 639798
Email: {asadas, asgupta}@ntu.edu.sg

Sukumar Nandi
Dept. of Computer Science & Engineering
Indian Institute of Technology
Guwahati, India, 781039
Email: sukumar@iitg.ernet.in

***Abstract*— Throughput, delay, and fairness are three most important performance metrics in IEEE 802.11-based wireless networks. In this paper, we provide a simple but precise analytical model to evaluate the above three metrics. Different from the previous models that have focused either on throughput or on delay, our model is unique as we obtain throughput, delay, and fairness all together. Moreover, by considering the fact that a packet may be dropped after a finite number of retrials, we distinguish five cases of packet delay, and derive the relationship among them. Using the proposed model, we carry out an extensive and insightful performance evaluation of IEEE 802.11 under different system parameters. Our results show that the throughput, delay, and fairness are quite sensitive to the system parameters being chosen, demonstrating the importance of performing dynamic tuning of the system parameters in IEEE 802.11. Moreover, the optimal values of the parameters are inconsistent with each other for different performance metrics, implying that some tradeoff among the metrics must be made in the dynamic tuning. Finally, our results also show that the previous models have overestimated the throughput as well as the packet delay.**

I. INTRODUCTION

Recently, wireless local area networks (LAN) and wireless ad-hoc networks have attracted considerable research interest. IEEE 802.11 [4] is the de facto standard for Wireless LANs, and it defines two MAC protocols: Point Coordination Function (PCF) and Distributed Coordination Function (DCF). DCF is a Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)-based MAC protocol. Due to the distributed nature, DCF is also popularly used in wireless ad-hoc networks.

In this paper, we focus on the performance evaluation of the DCF MAC protocol. Throughput, delay, and fairness are the three performance metrics that are of great interest to us. Most of the published work on the performance analysis of DCF focused either on throughput or on delay. In contrast, in this paper, we develop a simple analytical model that is able to evaluate the above three metrics all together. Our model assumes that a station transmits in a randomly chosen slot time with an independent probability τ, and the packet being transmitted experiences a collision with an independent probability p. In fact, τ and p characterize the main feature of the binary exponential back-off (BEB) algorithm used in DCF, and they can be obtained using the published models (e.g., [1]). With τ and p, we carry out the analysis for throughput, delay, and fairness. Using the proposed model, we carry out an extensive performance evaluation of DCF by varying the system parameters such as the retry limit, and the minimum and maximum contention window.

Our Main Contributions:

- In the throughput analysis, we provide a general analysis about the sensitivity of the throughput with respect to the system parameters, which is not available in the published papers. We also improve the preciseness of the throughput calculation by correcting some of the values used in [1].
- In the delay analysis, while the published models ([3], [10]) assume infinite retrials for a given packet, our model considers the fact of finite retrial in IEEE 802.11. Moreover, we have defined five kinds of delay and derived the relationship among these delays. Our analytical results show that these delays have very different values, showing the importance to distinguish them. Such an analysis is original and novel and has not been appeared in the published models on delay analysis.
- We have carried out an original fairness analysis based on the the probability distribution of the delay that is obtained in the delay analysis. The fairness analysis of IEEE 802.11 has not been done by the published models.
- Through an extensive performance evaluation of DCF based on the proposed model, we gain several insights about the behavior of IEEE 802.11. We arrive at several conclusions that are contrary to those obtained in previous models. For example, we find that the previous models assuming infinite retrials have overestimated the delay, and thus led to misleading conclusions. Our results also have great implications on the dynamic tuning of the system parameters in IEEE 802.11.

The remainder of the paper is organized as follows. In Section II, we describe the basic techniques of IEEE 802.11. The analytical model is presented in Section III. A thorough performance evaluation is carried out in Section IV. Future work and related work are given in Section V. The paper is concluded in Section VI.

II. BASIC TECHNIQUES IN IEEE 802.11 DCF

The DCF defines two methods for accessing the medium: the two-way handshake and the four-way handshake. In the two-way handshake, the sender first transmits a Data frame to

the receiver, which responds with an ACK frame if it receives the Data frame correctly. On the other hand, in the four-way handshake, a sequence of Request To Send (RTS), Clear To Send (CTS), Data, and ACK frames, is transmitted for the transmission of every single data packet.

The IEEE 802.11 adopts the well-known Binary Exponential Back-off (BEB) algorithm as its Contention Resolution (CR) mechanism, which is described as follows. Whenever an attempt to transmit a packet fails, a retransmission is scheduled, unless a retry limit (say *m*) is reached. Every station maintains a Contention Window (CW) and a back-off time counter. Before every transmission (including the retransmission), the station first backs off by a time according to Equation (1), unless the back-off time counter already contains a non-zero value, in which case it is unnecessary to generate a new random back-off counter.

$$Backoff\ Time = Random() \times SlotTime \tag{1}$$

The *SlotTime* is specified by the physical layer, and the *random* value is uniformly distributed over the range $[0, CW]$. For the *first* transmission attempt of a packet, the CW is set to CW_{min}. Whenever a retransmission is initiated, the CW is increased. The CW increases to its *maximum* value after $m^{'}$ retries. After that, the CW remains at the maximum value until the retry limit (i.e., m) is reached. Once the retry limit is reached, the packet is dropped, and the CW is reset to CW_{min}. The CW is also reset to CW_{min} whenever a transmission attempt is successful. For convenience, we call a retransmission attempt as a *stage*. If we denote the *stage number* by index *k*, which is in the range $[0, m]$, the CW can be expressed as:

$$CW_k = \begin{cases} (CW_{min}+1) \times 2^k - 1 & 0 \le k < m^{'} \\ (CW_{min}+1) \times 2^{m^{'}} - 1 & m^{'} \le k \le m \end{cases} \tag{2}$$

When a station (say *H*) is transmitting a packet, the other stations *freeze* their back-off counter. After station *H* completes the transmission of the packet and thus the medium becomes idle, all the contending stations first defer for a DCF Inter-Frame Space (DIFS) period. Then, station *H* generates a new random counter and backs off before it initiates another transmission. On the contrary, the other stations simply resume to count down from their *frozen* back-off counters. Contrary to a successful transmission, when a collision occurs, all the colliding stations will generate a new random counter.

III. Analytical Model

In the analysis, several popular assumptions have been made: (1) There are a finite number of stations (say n) in the network, and all the stations can hear each other (i.e., single-hop network); (2) Stations are always backlogged, meaning that a station always has packets waiting to be transmitted; (3) Ideal channel conditions (i.e., no transmission error or capture); (4) RTS/CTS handshake is used; and (5) The data packets have a fixed length.

A. Modelling Binary Exponential Back-off

In [1], by modelling the stochastic process representing the back-off counter at a given station as a discrete-time Markov chain, the values of two parameters that characterize the binary exponential back-off (BEB) algorithm are obtained. The first parameter is the *transmission probability*, τ, with which a station transmits in a randomly chosen slot time. The second parameter is the *conditional collision probability*, p, representing the probability of a collision experienced by a packet given that it is transmitted on the channel.

In the calculation of τ, the author in [1] did not consider the frame retry limit used in IEEE 802.11. In light of this, the authors in [12] have extended the model by considering the retry limit. According to [12], τ can be expressed in Equation (3), which is displayed on the top of the next page.

Moreover, the relationship between τ and p is as follows,

$$p = 1 - (1-\tau)^{n-1} \tag{4}$$

Equations (3) and (4) represent a nonlinear system with two unknowns τ and p, which can be solved using numerical techniques. With these two parameters, many performance metrics can be obtained. In the following subsections, we discuss how the throughput, delay, and fairness can be obtained. Note that the discussion below can also be made based on any other model (e.g., [2]) as long as τ and p can be obtained from the model.

B. Throughput Analysis

Now we calculate the normalized system throughput S, defined as the fraction of time that the channel is used to successfully transmit payload bits. The description in this subsection mainly follows from [1], except that we provide clearer insights in the model.

To compute S, we need to analyze what happens on the shared medium in a randomly chosen slot time. The slot may correspond to an idle slot, a collision, or a successful transmission. Let P_{idle}, P_{col}, and P_{succ} be the probabilities that a randomly chosen slot corresponds to an idle slot, a collision, or a successful transmission, respectively.

It is easy to see that P_{idle} is equal to the probability that $none$ of the n stations transmits on the shared medium in a randomly chosen slot time. Therefore,

$$P_{idle} = (1-\tau)^n \tag{5}$$

Clearly, if there is a successful transmission in a randomly chosen slot time, it means that *exactly one* station transmits on the shared medium in the slot. Therefore, P_{succ} can be expressed as,

$$P_{succ} = n\tau(1-\tau)^{n-1} \tag{6}$$

Obviously, whenever more than one stations transmit on the shared medium in a randomly chosen slot time, a collision occurs. Therefore, P_{col} is,

$$P_{col} = 1 - (1-\tau)^n - n\tau(1-\tau)^{n-1} \tag{7}$$

$$\tau = \begin{cases} \frac{2(1-2p)(1-p^{m+1})}{W(1-(2p)^{m+1})(1-p)+(1-2p)(1-p^{m+1})} & m \leq m^{'} \\ \frac{2(1-2p)(1-p^{m+1})}{W(1-(2p)^{m^{'}+1})(1-p)+(1-2p)(1-p^{m+1})+W2^{m^{'}}p^{m^{'}+1}(1-2p)(1-p^{m-m^{'}})} & m \geq m^{'} \end{cases} \quad (3)$$

where $W = CW_{min} + 1$.

Now, let σ, T_{col}, and T_{succ} be the duration of the slot corresponding to an idle slot, a collision, or a successful transmission, respectively. Therefore, the average duration (represented by T_{avg}) that a generic slot lasts is,

$$T_{avg} = P_{idle}\sigma + P_{succ}T_{succ} + P_{col}T_{col} \quad (8)$$

Now, the throughput S can be calculated as,

$$\begin{aligned} S = & \frac{E[payload\ information\ transmitted\ in\ a\ slot\ time]}{E[length\ of\ a\ slot\ time]} \\ = & \frac{P_{succ} \times E[P]}{T_{avg}} = \frac{P_{succ} \times E[P]}{P_{idle}\sigma + P_{succ}T_{succ} + P_{col}T_{col}} \end{aligned} \quad (9)$$

where $E[P]$ is the average payload size (in terms of time units, e.g., μs), and thus $P_{succ} \times E[P]$ is the average amount of payload information successfully transmitted in a generic slot time.

By dividing the numerator and denominator of Equation (9) by P_{succ}, the throughput can be expressed as follows,

$$S = \frac{E[P]}{T_{succ} + \frac{P_{col}}{P_{succ}}T_{col} + \frac{P_{idle}}{P_{succ}}\sigma} \quad (10)$$

The above formula is quite intuitive. The denominator represents the average amount of time spent in order to observe a successful packet transmission on the channel. Specifically, we can view the packet transmissions on the channel as the *bernoulli trials* with a success probability P_{succ}. Therefore, the number of slots (including the slot for the successful transmission) required to observe a successful transmission is given by the geometric distribution. Clearly, the average number of slots required is simply $1/P_{succ}$. Out of these slots, the average numbers of idle and collision slots are P_{idle}/P_{succ} and P_{col}/P_{succ}, respectively. In addition, there is one slot for the successful transmission. Moreover, as mentioned before, the duration of the idle, collision, and successful transmission slots are σ, T_{col}, and T_{succ}, respectively. This explains Equation (10).

The throughput S can also be expressed as follows,

$$S = \frac{E[P]/T_{succ}}{1 + \frac{P_{col}}{P_{succ}} \times \frac{T_{col}}{T_{succ}} + \frac{P_{idle}}{P_{succ}} \times \frac{\sigma}{T_{succ}}} \quad (11)$$

The sensitivity of the throughput to the system parameters (e.g., CW_{min}, CW_{max}, and m) can be easily analyzed from the above formula. Clearly, $E[P]$, T_{succ}, T_{col}, and σ are independent to the system parameters. On the other hand, as known from equations (5), (6) and (7), the values of P_{col}, P_{succ}, and P_{idle} depend on τ and p, which themselves depend on the system parameters (see equations (3) and (4)). Therefore, we only need to discuss the sensitivity of the throughput with respect to P_{col}, P_{succ}, and P_{idle}. From Equation (11), we know that the throughput S is sensitive to the value of P_{col}/P_{succ} only when the constant T_{col}/T_{succ} is large (compared to the first term of the denominator, i.e., unity). Similarly, S is sensitive to P_{idle}/P_{succ} only when the constant σ/T_{succ} is large. The sensitivity analysis gives an idea whether dynamic tuning of the system parameters is necessary or not. For example, if S is insensitive to the system parameters, it may not be necessary to do any dynamic tuning of the parameters.

As mentioned in [1], the above analysis applies to both the two-way and four-way handshakes. To specifically compute the throughput for a given handshake, we only need to specify the corresponding values of T_{col} and T_{succ}. Note that the idle slot time σ is specific to the physical layer, e.g., 20 μs for Direct Sequence Spread Spectrum (DSSS) [4].

The specific values of T_{succ} and T_{col} for the four-way handshake are as follows,

$$\begin{cases} T_{succ} = RTS + CTS + Data + ACK + 3SIFS + DIFS \\ T_{col} = RTS + DIFS + SIFS + CTS \end{cases} \quad (12)$$

It may be noticed that the above expression is different from that in [1]. First, the propagation delay δ is no longer included in the above formula since the Short Inter-Frame Space (SIFS) has already contained δ according to the standards [4]. Another difference is that in the expression of T_{col}, we have added an additional term, $SIFS + CTS$. As stated in [1], T_{col} is the period of time during which the channel is sensed busy by the *non-colliding* stations. After a collision, the colliding stations wait for a time equal to $CTSTimeout$ (or $ACKTimeout$ in the two-way handshake case). On the other hand, the non-colliding stations defer by an Extended Inter-Frame Space (EIFS) value since they cannot interpret the contents of the colliding frames. Moreover, the EIFS value is equal to $DIFS+SIFS+TxTime(ACK)$, which is large enough for the complete transmission of the CTS or ACK frame (note that CTS has the same length as ACK). Therefore, if $CTSTimeout$ or $ACKTimeout$ is set to the time needed for the transmission of the CTS/ACK frame (which is true in the popularly adopted NS-2 simulator), then, T_{col} for all the stations, whether involved in the collision or not, will be the same. Therefore, the $SIFS+CTS$ term should be included in T_{col}. Clearly, the above modifications are important to obtain a more precise value of the throughput. The model in [1] overestimates the throughput as it uses a smaller T_{col}.

C. Delay Analysis

In IEEE 802.11, there is no queue at the MAC layer itself, and thus IEEE 802.11 standards have not specified any queuing mechanisms. However, normally there should be a queue at the top of the MAC layer. Therefore, in general, the delay that a packet experiences should include two parts, i.e.

the delay experienced in the queue and the delay experienced at the MAC layer. We only focus on the MAC layer delay. Since there are *multiple* stations contending for the shared medium and IEEE 802.11 DCF is a *random* access protocol, the MAC layer delay is a *random* value, which requires a detailed analysis. Five types of delay at the MAC layer are relevant to this analysis, which we will discuss in a little while.

As we know, in IEEE 802.11, if a packet is handed to the MAC layer from the upper layer, the packet may be transmitted successfully in one or several trials, or the packet will be dropped if the retry limit is reached. In both cases, the MAC layer will notify the upper layer about the final status of the packet (i.e., transmitted successfully or dropped). Therefore, the MAC layer delay should be defined to be the time interval *from the instant* that the packet is handed to the MAC layer *to the instant* that the upper layer gets a notification from the MAC layer regarding the final status of the packet. Accordingly, we can define the following four kinds of delays.

Delay D_{succ}: if a packet is successfully transmitted, what is the delay experienced at the MAC layer?

Delay D_{drop}: if a packet is dropped, what is the delay experienced at the MAC layer?

Delay D_{notify}: combining the above two cases, what is the delay before the upper layer will get a notification from the MAC layer about the final status (transmitted or dropped) of the packet?

Delay $D_{intersucc}$: from the viewpoint of an upper layer at a *given* station, what is the delay between two successful packet transmissions?

Note that in the definition of $D_{intersucc}$, the successful transmissions must belong to a single station only. Also, as shown later, $D_{intersucc}$ is directly related to the throughput that a single station gets, from which we can obtain the total system throughput S that has been derived in the previous subsection.

At last, to show the error introduced by the infinite retrials assumption [3], [10], we are also interested in the following delay:

Delay $D_{infinite}$: if we assume that a packet will always be retransmitted (i.e., infinite retrials) until it is transmitted successfully, what is the delay experienced at the MAC layer?

In the following subsections, we discuss how to compute the above delays. In the calculations, we assume that the packet handed to the MAC layer must experience the back-off process, though the back-off process is unnecessary if the shared medium is idle at the instant the packet is handed to the MAC layer. However, under the saturation condition, the probability of such an event should be very small and thus we can safely ignore it.

1) Derivation of D_{succ}*:* Clearly, D_{succ} depends upon the number of back-off stages that a packet experiences, as well as upon the back-off counter generated at each stage. Figure 1 gives an example of the back-off process experienced by a packet at a given station. In the figure, the shaded blocks represent the events in which the given node is involved. In the example, the packet experiences three stages (from stage 0 to stage 2) before it is transmitted successfully, and the back-off counter (represented by B_0, B_1, and B_2, respectively) generated at the stages are 3, 4, 2, respectively.

From the figure, it is easy to find that the back-off counter at the given station decrements by one after every slot time if the slot corresponds to: (1) an idle slot, (2) a collision between *other* stations, or (3) a successful transmission between *other* stations[1]. Since we know that the duration of the idle, collision, and successful transmission slots are σ, T_{col}, and T_{succ}, respectively, we can easily get the delay that is contributed by the idle slots, as well as by the collisions among other stations, and by the successful transmissions of other stations. For instance, in the above example, during the back-off process for the given packet, the number of idle slots is 4, while the numbers of collisions and successful transmissions among other stations are 3, and 2, respectively. Therefore, the delay contributed by the back-off slots is $4\sigma + 3T_{col} + 2T_{succ}$. In addition to this, the delay D_{succ} should include the time spent on the collisions that the given station *itself* is involved, e.g., $2 \times T_{col}$ in the above example. Finally, the D_{succ} should also include the time needed for the successful transmission of the given packet, i.e., T_{succ}. Therefore, the delay D_{succ} is $(4\sigma + 3T_{col} + 2T_{succ}) + 2T_{col} + T_{succ}$. In the following, we provide a formal method to calculate D_{succ} for a general case.

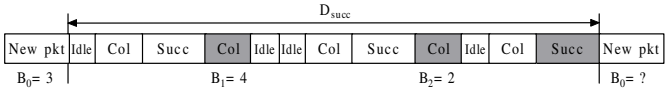

Fig. 1. Illustration of Delay for Successful Packet

Let B_i be the random back-off counter generated at stage i, and let $B(j)$ be the *sum* of the back-off slots generated at stages from 0 to j. Therefore,

$$B(j) = \sum_{i=0}^{j} B_i \tag{13}$$

We know that B_i at stage i follows the uniform distribution in the range $[0, CW_i]$, where CW_i is expressed by Equation (2). It is easy to get the probability mass function (pmf) of $B(j)$, which is simply a *convolution* of the pmfs of all B_i where $i \in [0, j]$. Clearly, $B(j)$ is in the range of $[0, B(j)_{max}]$, where $B(j)_{max} = \sum_{i=0}^{j} CW_i$.

Since the probability that a transmission experiences a collision is p, the probability that a packet is successfully transmitted at stage j (starting from stage 0) is,

$$P_{succ}(j) = p^j \times (1-p) \qquad 0 \le j \le m \tag{14}$$

Moreover, the probability that a packet is dropped because the retry limit (i.e., m) is reached is,

$$P_{drop} = 1 - \sum_{j=0}^{m} p^j \times (1-p) = p^{m+1} \tag{15}$$

Let $P_{succ}(j, b)$ be the probability that the packet is successfully transmitted at stage j ($0 \le j \le m$), and the sum of the

[1]In fact, according to IEEE 802.11, corresponding to a collision or a successful transmission, the counters at the stations do not decrement. However, the imprecision caused by such an approximation is insignificant.

back-off slots generated up to stage j is equal to b. Therefore, $P_{succ}(j,b)$ can be expressed as,

$$P_{succ}(j,b) = P_{succ}(j) \times Pr(B(j) = b) \quad 0 \le j \le m \quad (16)$$

Similarly, let $P_{drop}(b)$ be the probability that a packet is dropped, and the sum of the back-off slots generated up to stage m is equal to b. Therefore, $P_{drop}(b)$ can be expressed as,

$$P_{drop}(b) = P_{drop} \times Pr(B(m) = b) \quad (17)$$

Since D_{succ} represents the delay of a packet that is successfully transmitted, we should only consider the packets that are successfully transmitted while excluding the packets that are dropped. Therefore, we define the following conditional probability, $P^{'}_{succ}(j,b)$, which represents the probability that the back-off process for a packet ends at stage j and the sum of the back-off slots generated up to stage j is b, given that the packet is successfully transmitted. Therefore,

$$P^{'}_{succ}(j,b) = \frac{P_{succ}(j,b)}{1 - P_{drop}} \quad (18)$$

As a generalization of the example given in Figure 1, the corresponding delay, $D_{succ}(j,b)$, which represents the delay of a packet that is successfully transmitted at stage j and the sum of the back-off slots generated up to stage j is b, is as follows,

$$D_{succ}(j,b) = b \times T_{avg} + j \times T_{col} + T_{succ} \quad (19)$$

Note that there is an approximation in the above equation, that is, we have simply used $b \times T_{avg}$ to represent the duration of the b number of back-off slots. In fact, if, among the b number of back-off slots, there are b_{col} number of slots corresponding to collisions, b_{succ} number of slots corresponding to successful transmissions, and b_{idle} number of idle slots, the delay contributed by the b number of back-off slots is $(b_{col}T_{col} + b_{succ}T_{succ} + b_{idle}\sigma)$. The random variables b_{col}, b_{succ}, b_{idle}, and b can be characterized by a multinomial probability distribution. However, it makes the analysis extremely complex. Moreover, the error introduced by the above approximation is very small if b is relatively large, which is normally true in IEEE 802.11. For example, when the minimum contention window is 32, the average value b should be 16 even if all the packets experience only one stage (i.e., stage 0). Therefore, the above approximation is reasonable and will be used throughout this paper.

Equations (18) and (19) express the probability mass function of the delay D_{succ} in terms of the stage number (i.e., j) and the sum of the number of back-off slots (i.e., b). Therefore, the average of D_{succ} is,

$$\overline{D_{succ}} = \sum_{j=0}^{m} \sum_{b=0}^{B(j)_{max}} D_{succ}(j,b) \times P^{'}_{succ}(j,b) \quad (20)$$

After a few simple steps, we get the following,

$$\overline{D_{succ}} = T_{succ} + \frac{1}{1 - P_{drop}} \sum_{j=0}^{m} [T_{avg} \times \overline{B(j)} + j \times T_{col}] P_{succ}(j) \quad (21)$$

where $\overline{B(j)}$ is the average of $B(j)$, and can be expressed as,

$$\overline{B(j)} = \frac{B(j)_{max}}{2} = \frac{\sum_{i=0}^{j} CW_i}{2} \quad (22)$$

From Equation (21), we can easily calculate the value of $\overline{D_{succ}}$ under any given set of system parameters.

With the probability mass function of the delay expressed by equations (18) and (19), we can compute other important metrics, such as the standard deviation of the delay.

2) Derivation of D_{drop}: Similar to the derivation of D_{succ}, we present an example in Figure 2 for the case that a packet will be dropped. As shown in the figure, in contrast to the case of a successful transmission, if a packet is dropped, the back-off process experienced by the packet must reach the retry limit (i.e., stage m).

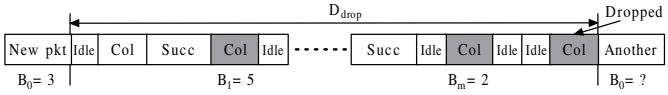

Fig. 2. Illustration of Delay for Dropped Packet

To compute D_{drop}, which represents the delay of a packet that is dropped, we define the following conditional probability as done in the derivation of D_{succ},

$$P^{'}_{drop}(b) = \frac{P_{drop}(b)}{P_{drop}} = Pr(B(m) = b) \quad (23)$$

where $P_{drop}(b)$, P_{drop}, and $B(m)$ have been defined in equations (17), (15), and (13), respectively.

The corresponding delay, using the same approximation as in Equation (19), can be expressed as follows,

$$D_{drop}(b) = b \times T_{avg} + m \times T_{col} + T_{col} \quad (24)$$

The above two equations express the probability mass function of the delay D_{drop}. The average of D_{drop} is,

$$\overline{D_{drop}} = \sum_{b=0}^{B(m)_{max}} D_{drop}(b) \times P^{'}_{drop}(b) \quad (25)$$

After some simple steps, we get the following expression,

$$\overline{D_{drop}} = T_{col} \times (m+1) + T_{avg} \times \overline{B(m)} \quad (26)$$

where $\overline{B(m)}$ is the average value of $B(m)$, and is equal to $B(m)_{max}/2$.

The variance of the delay can be obtained as follows (note that all B_i are independent to each other),

$$\begin{aligned} Var(D_{drop}) &= T^2_{avg} \times Var(B(m)) = T^2_{avg} \times Var(\sum_{i=0}^{m} B_i) \\ &= T^2_{avg} \times \sum_{i=0}^{m} Var(B_i) \end{aligned} \quad (27)$$

where $Var(B_i) = ({CW_i}^2 + 2CW_i)/12$.

3) Derivation of D_{notify}: The probability mass function of the delay D_{notify} is expressed by equations (16), (19), (17) and (24). Note that we should not use the conditional probabilities expressed in equations (18) and (23) since D_{notify} represents the delay for a general packet (that is either dropped or successfully transmitted). The average of D_{notify} can be obtained as follows,

$$\begin{aligned}\overline{D_{notify}} &= \sum_{j=0}^{m}\sum_{b=0}^{B(j)_{max}}[D_{succ}(j,b)\times P_{succ}(j,b)] \\ &+ \sum_{b=0}^{B(m)_{max}}[D_{drop}(b)\times P_{drop}(b)] \end{aligned} \quad (28)$$

After some simple steps, we get the following expression,

$$\overline{D_{notify}} = (1-P_{drop})\overline{D_{succ}} + P_{drop}\overline{D_{drop}} \quad (29)$$

The above result is quite intuitive. The first term expresses the fact that the packet is successfully transmitted with a probability $(1-P_{drop})$ and the corresponding average delay is $\overline{D_{succ}}$, while the second term expresses the fact that the packet is dropped with a probability P_{drop} and the corresponding average delay is $\overline{D_{drop}}$. We can also compute the standard deviation of D_{notify}.

4) Derivation of $D_{intersucc}$: In contrast to D_{succ}, D_{drop}, and D_{notify} discussed above, we are not able to get the probability mass function of the delay $D_{intersucc}$. However, we can get the average value of $D_{intersucc}$. Specifically, we can view the packet transmissions at a given station as a *bernoulli experiment* with the probability of failure P_{drop}, and the probability of success $(1-P_{drop})$. Therefore, to observe a successful transmission at the given station, the number of trials required follows a geometric distribution. Clearly, the average number of trials required is simply $\frac{1}{1-P_{drop}}$. Among these trials, a fraction of P_{drop} trials fail and the packets are dropped, and each of the failing trial lasts on an average $\overline{D_{drop}}$ duration. Obviously, among the trials the last trial must be successful, and it lasts on an average $\overline{D_{succ}}$ duration. Therefore, the average delay $\overline{D_{intersucc}}$ can be obtained as,

$$\begin{aligned}\overline{D_{intersucc}} &= \frac{P_{drop}}{1-P_{drop}}\overline{D_{drop}} + \frac{1-P_{drop}}{1-P_{drop}}\overline{D_{succ}} \\ &= \frac{\overline{D_{notify}}}{1-P_{drop}} \end{aligned} \quad (30)$$

The last expression is quite intuitive. To observe a successful transmission at a given station, on an average, the station gets $1/(1-P_{drop})$ notifications and the average duration between two consecutive notifications is $\overline{D_{notify}}$.

Interestingly, using the above result, we can also obtain the throughput at any single station, which is simply equal to $E(P)/\overline{D_{intersucc}}$. Observing that every station gets the same throughput in a long term, if there are n number of stations in the network, then the total system throughput S can be obtained as follows,

$$S = \frac{E(P)}{\overline{D_{intersucc}}}\times n = \frac{E(P)}{\overline{D_{succ}} + \frac{P_{drop}}{1-P_{drop}}\overline{D_{drop}}}\times n \quad (31)$$

Note that in the above we obtain the system throughput by examining the events at a typical station. This is different from the throughput analysis discussed in Section III-B, where the throughput is obtained by examining the events on the shared channel.

Equation (31) relates the delay, throughput, and packet loss probability, and thus it may be of great importance in developing dynamic control algorithms that aim to achieve a tradeoff among the delay, throughput, and packet loss probability.

5) Derivation of $D_{infinite}$: Now we derive the delay under the infinite retrials assumption that is adopted in [3], [10]. Let $P_{infinite}(j,b)$ be the probability that a packet is successfully transmitted at stage j ($0 \le j < \infty$), and the sum of the backoff slots generated up to stage j is equal to b. Therefore,

$$P_{infinite}(j,b) = P_{succ}(j)\times Pr(B(j)=b) \qquad j\in[0,\infty) \quad (32)$$

where $P_{succ}(j)$ is defined in Equation (14) except that the range of j is changed to $[0,\infty)$, and $B(j)$ is defined in Equation (13). Clearly, the corresponding delay is as follows,

$$D_{infinite}(j,b) = b\times T_{avg} + j\times T_{col} + T_{succ} \qquad j\in[0,\infty) \quad (33)$$

The above two formulas form the probability mass function of the delay $D_{infinite}$. Therefore, the average of $D_{infinite}$ is,

$$\begin{aligned}\overline{D_{infinite}} &= \sum_{j=0}^{\infty}\sum_{b=0}^{B(j)_{max}}[D_{infinite}(j,b)\times P_{infinite}(j,b)] \\ &= \sum_{j=0}^{m}\sum_{b=0}^{B(j)_{max}}[D_{infinite}(j,b)\times P_{infinite}(j,b)] \\ &+ \sum_{j=m+1}^{\infty}\sum_{b=0}^{B(j)_{max}}[D_{infinite}(j,b)\times P_{infinite}(j,b)] \end{aligned} \quad (34)$$

After several simple steps, we find that the first term is simply equal to $(1-P_{drop})\overline{D_{succ}}$. This is very intuitive as explained now. Under the infinite retrials assumption, the probability that a packet is successfully transmitted at a stage later than stage m is equal to the probability that the packet is dropped if the retry limit is m. Therefore, the probability that a packet is transmitted successfully at one of the initial m stages is $(1-P_{drop})$, and the average delay of such a packet is $\overline{D_{succ}}$. This explains the first term. After some simple steps, the second term of Equation (34) (let us represent it by T_2), can be expressed as follows,

$$\begin{aligned}T_2 = P_{drop}[&T_{succ} + (m+\frac{1}{1-p})T_{col} + \\ &(\frac{B(m)_{max}}{2} + \frac{CW_{max}}{2}\frac{1}{1-p})T_{avg}] \end{aligned} \quad (35)$$

The above term is also quite intuitive. If a packet experiences more than m stages (with a probability P_{drop}), it will first experience m collisions. Then, the average number of additional collisions experienced is simply $1/(1-p)$ due to the *memoryless* property of the geometric distribution. The above arguments explain the term $(m+\frac{1}{1-p})T_{col}$ in Equation (35). Now we explain the term $(\frac{B(m)_{max}}{2}+\frac{CW_{max}}{2}\frac{1}{1-p})T_{avg}$. Specifically, $B(m)_{max}/2$ represents the average number of back-off slots in the first m stages. Then, at every additional stage, on an average $CW_{max}/2$ slots are involved, and the average number of additional stages is simply $1/(1-p)$. In the above discussion, we have assumed that once the CW reaches its maximum value (CW_{max}), it does not increase any more in the remaining stages.

From equations (34) and (35), we obtain,

$$\overline{D_{infinite}} = (1-P_{drop})\overline{D_{succ}} + T_2 \tag{36}$$

We can also express the average delay as follows,

$$\begin{aligned}\overline{D_{infinite}} &= (1-P_{drop})\overline{D_{succ}} + P_{drop}\overline{D_{drop}} + T_3 \\ &= \overline{D_{notify}} + T_3 \end{aligned} \tag{37}$$

where

$$T_3 = P_{drop}[T_{succ} + \frac{p}{1-p}T_{col} + \frac{CW_{max}}{2}\frac{1}{1-p}T_{avg}] \tag{38}$$

We notice that $\overline{D_{infinite}}$ is always larger than $\overline{D_{notify}}$ as well as $\overline{D_{succ}}$ (since $\overline{D_{notify}}$ is always larger than $\overline{D_{succ}}$). Therefore, in general $\overline{D_{infinite}}$ overestimates the delay. In fact, as shown later by the numerical results, the results of $\overline{D_{infinite}}$ is very misleading whenever p (or P_{drop}) is large.

From equations (32) and (33), we can also numerically obtain the standard deviation of $D_{infinite}$.

D. Fairness Analysis

IEEE 802.11 is fair in the *long-term* (e.g., equal average delay or throughput) in a *single-hop* network[2]. However, due to the randomness involved in the binary exponential back-off (BEB) algorithm, substantial *short-term* unfairness remains even in a single-hop scenario as clearly shown in [7].

In general, unfairness can be measured in terms of the throughput or the packet delay. To measure the short-term unfairness through the packet delay, an ideal way is to first get the probability distribution of the delay experienced at all the stations, and then see how variable the delay is. Clearly, higher variability implies a more unfair system. However, to get such a distribution, we need to model the system state in a way that there is at least one separate parameter used to represent the state of each station, making the state space very large if the number of stations is large. On the other hand, if we assume that the probability distribution of the delay experienced at a given station is similar to the distribution of the delay experienced at all the stations, the problem becomes much easier as we only need to consider the states at a typical station. We adopt this approach as it is much simpler but still able to reveal the general trend of the unfairness, though it may not give a very precise measurement of the unfairness.

In the previous subsection we have already obtained the probability distribution of the delay (e.g., D_{succ}) experienced at a typical station. Therefore, we now only need to develop a metric to quantify the variability of the delay and thus reflect the unfairness. The standard deviation is a good candidate, however, it is not independent of scale (i.e., the unit of measurement) [5] as shown by the following example. Considering there are two different schemes to distribute a common resource to three users, the delays experienced by the users under the two schemes are as follows:

Scheme I=$\{1, 2, 3\}$

Scheme II=$\{2, 4, 6\}$

Clearly, the two schemes deliver the same fairness as far as the delay is concerned. However, the standard deviation of the delay under the two schemes are $\sqrt{2}$ and $2\sqrt{2}$, respectively. Therefore, if we use the standard deviation as a metric to reflect unfairness, we will arrive at a misleading conclusion: the second scheme is two times more unfair than the first one. It is clear that the standard deviation is not a good metric to reflect the fairness as it is not scale independent. Now we look at the coefficient of variation (COV) of the delay, which can also reflect the variability of the delay. The COV is defined as,

$$COV = \frac{\text{Standard Deviation}}{\text{Average}} \tag{39}$$

Clearly, the COV under the above two schemes are the same, i.e., $\frac{\sqrt{2}}{2}$, implying that they deliver the same fairness. In fact, as mentioned in [5], COV in many senses is a good metric to reflect fairness. In general, the larger the COV is, the broader the delay spreads, and thus the more unfair the scheme is. Numerically, we can easily find the value of the COV of the delay from the results in the previous subsection.

Since the Jain's index [5] is a more popular metric for the unfairness, now we discuss how to relate the COV to the Jain's index. If there are n number of users sharing the common resource, and user i gets a proportion of x_i, the Jain's index is defined as follows,

$$\text{Jain's Index} = \frac{[\sum_i x_i]^2}{n\sum_i x_i^2} \tag{40}$$

On the other hand, the square of the COV is as follows,

$$COV^2 = \frac{\text{Variance}}{\text{Average}^2} = \frac{\overline{x^2}-(\overline{x})^2}{(\overline{x})^2} \tag{41}$$

Now we can get the relationship between the COV and the Jain' index as follows,

$$\begin{aligned}\text{Jain's Index} &= \frac{[\sum_i x_i]^2}{n\sum_i x_i^2} = \frac{[\frac{1}{n}\sum_i x_i]^2}{\frac{1}{n}\sum_i x_i^2} = \frac{(\overline{x})^2}{\overline{x^2}} \\ &= \frac{(\overline{x})^2}{(\overline{x})^2+\overline{x^2}-(\overline{x})^2} = \frac{1}{1+\frac{\overline{x^2}-(\overline{x})^2}{(\overline{x})^2}} = \frac{1}{1+COV^2}\end{aligned} \tag{42}$$

[2]Note that in a *multi-hop* scenario, IEEE 802.11 cannot even maintain the long-term fairness among the contending stations [8].

Clearly, while the COV is boundless, the Jain's index is in the range of $[0, 1]$. However, note that the Jain's index obtained from the COV in our case is not able to tell exactly how unfair the system is since we obtain it through the delay distribution at a typical station rather than through a distribution of the delay experienced at all the contending stations.

IV. Performance Evaluation of IEEE 802.11

In this section, we present the performance results for the four-way handshake in IEEE 802.11 DCF under the Direct Sequence Spread Spectrum (DSSS) physical layer [4]. Table I lists the values of the parameters used in the calculation. Rather than using the same transmission rate (i.e., 1 Mbps) for both the Data and control frames as in [1], to closely follow the standards, in our calculation the transmission rate for the Data frame is 2 Mbps while that for the control frames is 1 Mbps. Using the parameters listed in the above table, we can calculate $E(P)$, T_{succ}, and T_{col}, which are 4096 μs, 5440 μs, and 716 μs, respectively.

TABLE I
IEEE 802.11 Parameters used in the Calculation

Payload of data packet	1024 bytes
Data	1024 bytes + MAC header + PHY header
RTS	20 bytes + PHY header
CTS	14 bytes + PHY header
ACK	14 bytes + PHY header
PHY Header	192 us
MAC Header	28 bytes
Basic rate	1 Mbps
Data rate	2 Mbps
Slot time	20 us
SIFS	10 us
DIFS	50 us
EIFS	364 us

As shown in the previous section, the performance depends on four parameters: n, CW_{min}, CW_{max}, and m. In all the following performance graphs, the horizontal axis represents the number of stations in the network (i.e., n). The vertical axis represents one of the four performance metrics (throughput, packet drop probability, delay, or fairness) while in a graph one of the three system parameters (i.e., CW_{min}, CW_{max}, or m) is varied and the remaining two parameters are fixed at their standard values. The standard values [4] are as follows: $CW_{min} = 31$, $CW_{max} = 1023$, and $m = 6$.

A. Throughput Results

Figures 3-5 present the throughput results. From the figures, we can make three main observations: (i) Compared to the throughput presented in [1], the throughput obtained here is much smaller; (ii) In general, the larger the CW_{min}, CW_{max}, and m are, the larger the throughput is. On the other hand, when the network size (i.e., n) increases, the throughput normally decreases; (iii) The throughput is quite sensitive to the system parameters (i.e., CW_{min}, CW_{max}, and m) as well as to the network size, which is contrary to the observation made in [1].

There are two reasons behind the first observation. Firstly, since the model in [1] has not considered the retry limit (i.e., it assumes infinite retrials), it overestimates the throughput as mentioned in [12]. In fact, this can also be easily verified from Figure 5, which shows that the throughput increases with the increase of the retry limit m. The second reason is as follows. As mentioned in the discussion for Equation (12), T_{col} in our calculation is larger than the one in [1]. On the other hand, T_{succ} is smaller than that in [1] since we use a higher transmission rate for the Data frame. As a result, in our calculation, the time spent on collisions increases while the time spent on the payload transmission decreases. The above arguments explain why the throughput obtained here is smaller than that in [1].

To explain the second observation (i.e., how the throughput changes with the system parameters), we first need to look at the values of T_{col}/T_{succ} and σ/T_{succ} as discussed in the context of Equation (11). In our calculation, σ/T_{succ} is very small (i.e., 20/5440) compared to the unity, and thus the throughput S is insensitive to the value of P_{idle}/P_{succ}. On the contrary, T_{col}/T_{succ} is relatively large (i.e., 716/5440), and thus S is sensitive to the value of P_{col}/P_{succ}. Therefore, to explain the second observation, we only need to look at the value of P_{col}/P_{succ} under different system parameters. Clearly, the larger P_{col}/P_{succ} is, the smaller the throughput is. To exemplify this, Figure 6 presents the value of P_{col}/P_{succ} when the CW_{min} is varied. We notice that the ratio increases as the CW_{min} decreases or as n increases, explaining why the throughput in Figure 3 decreases in these two cases.

Now let us explain the third observation. As mentioned, T_{col}/T_{succ} is quite large in our case, and thus the throughput S is very sensitive to the system parameters. In contrast, the T_{col}/T_{succ} is very small (i.e., 8/191.36) in [1], and thus the throughput S presented there is insensitive to the system parameters.

Here, we would like to point out that in the emerging IEEE 802.11 standards (e.g., IEEE 802.11-b, -a and -g), the T_{col}/T_{succ} is becoming larger because the transmission rate of the Data frames (which determines T_{succ}) is increasing (e.g., can be as high as 54 Mbps) while the transmission rate of the control frames (which determines T_{col}) remains small. Therefore, we expect the throughput in the emerging standards to be more sensitive to the system parameters, and thus the dynamic tuning of the system parameters will be of great importance in these standards.

B. Packet Drop Probability Results

Figures 7-9 present the packet drop probability. From the figures, we find that, the drop probability increases when CW_{min}, CW_{max}, or m decrease. This is due to the fact that when CW_{min}, CW_{max}, or m decrease, the conditional collision probability p increases. We also notice that the drop probability is quite large (e.g., larger than 0.5) when the network size is large. This observation will be useful when we explain the large difference between the delay D_{notify} and $D_{infinite}$ later.

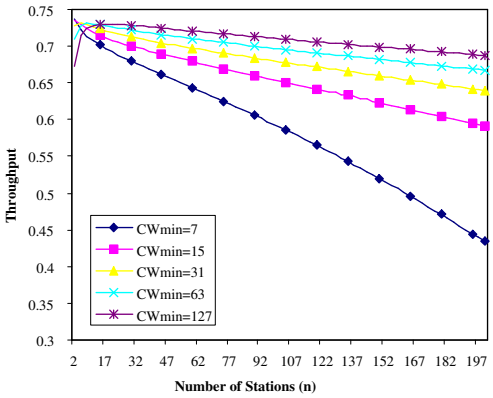

Fig. 3. Throughput under variable CW_{min}

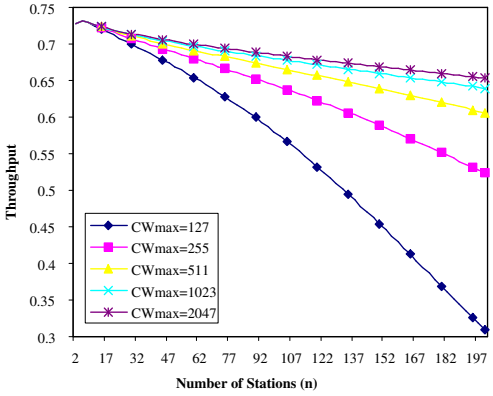

Fig. 4. Throughput under variable CW_{max}

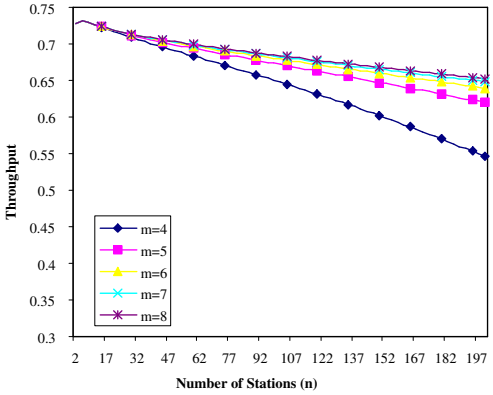

Fig. 5. Throughput under variable retry limit m

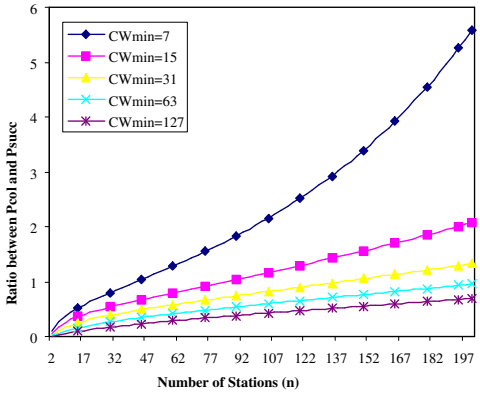

Fig. 6. Ratio P_{col}/P_{succ} under variable CW_{min}

C. Delay Results

In figures 10-14, we first present the five kinds of delay by varying CW_{min}. From the results, we find that: (i) In general, in the first three kinds of delay (i.e., $\overline{D_{succ}}$, $\overline{D_{drop}}$, and $\overline{D_{notify}}$), the larger the CW_{min} is, the larger is the delay; (ii) In contrast, $\overline{D_{intersucc}}$ and $\overline{D_{infinite}}$ decrease when CW_{min} increases; (iii) $\overline{D_{drop}}$ is much higher than $\overline{D_{succ}}$ (note that the vertical axes have different scales), while $\overline{D_{notify}}$ stands in between the above two; (iv) $\overline{D_{infinite}}$ is much larger than $\overline{D_{succ}}$ and $\overline{D_{notify}}$, showing the error introduced by the assumption of the infinite retrials.

The first observation is quite intuitive, and can be explained by the fact that a larger CW_{min} should result in a larger average number of back-off slots. Also, the trend of $D_{intersucc}$ with respect to CW_{min} coincides with the trend of throughput with respect to CW_{min}. In fact, using Equation (31), we can obtain similar throughput results as those presented in Figure 3. The trend of $D_{infinite}$ with respect to CW_{min} is similar to that obtained in [3], and gives the same misleading conclusion: a larger CW_{min} results in a smaller delay. Now we explain why the trend corresponding to $D_{infinite}$ is contrary to that of D_{notify} (as well as of D_{succ} and D_{drop}). As known from Equation (38), the difference between $D_{infinite}$ and D_{notify} becomes larger when the conditional collision probability p (or P_{drop}) becomes larger. Clearly, when CW_{min} is smaller, p is larger, resulting in a larger error in the calculation of $D_{infinite}$. On the contrary, the error is much smaller when CW_{min} is larger as p becomes smaller. Therefore, $D_{infinite}$ becomes larger under smaller CW_{min}. In conclusion, whenever the conditional collision probability (or the packet drop probability P_{drop}) is large, the assumption of infinite retrials will result in a large error in the calculation of the delay.

Figures 15-17 present the standard deviation of the first three kinds of delay (i.e., D_{succ}, D_{drop}, and D_{notify}) for different values of CW_{min}. In general, they follow the similar trend as the average delay. D_{notify} has a larger standard deviation than the other two since it considers the delay of all the packets (i.e., dropped as well as successfully transmitted), and since there is a large difference between the average delay of the packets being dropped and those being transmitted successfully.

In the above, we have presented the delay results when CW_{min} is variable. Similarly, we have also obtained the results when CW_{max} or m varies. However, here we only present the average of D_{succ} in figures 18-19. In general, the larger the CW_{max} and m are, the larger is the average delay. Under variable CW_{max} or m, the average of other kinds of delay and their standard deviations, in general, have the trend similar to those under variable CW_{min}.

D. Fairness Results

Figures 20-22 present the Coefficient of Variation (COV) of D_{succ} for different values of CW_{min}, CW_{max}, and m, respectively. To recall, a larger COV implies a more unfair system. From the figures, we can clearly make two observations: (i) In general, the larger the CW_{min} is and the smaller the CW_{max}

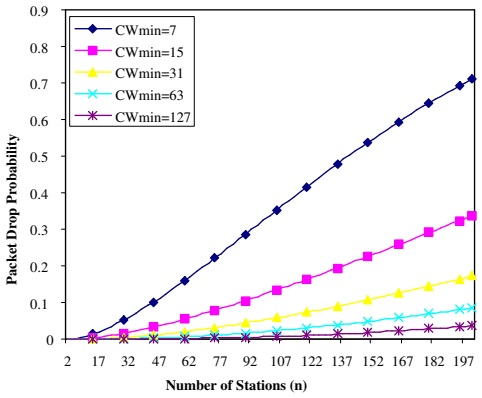

Fig. 7. Drop probability under variable CW_{min}

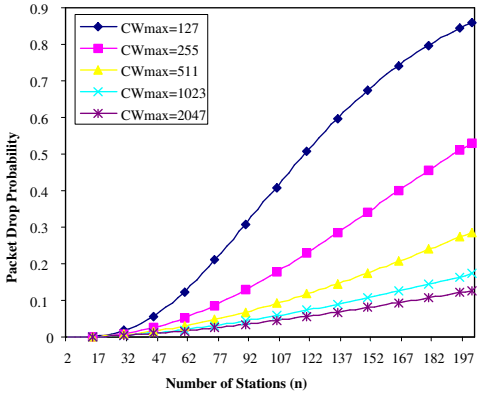

Fig. 8. Drop probability under variable CW_{max}

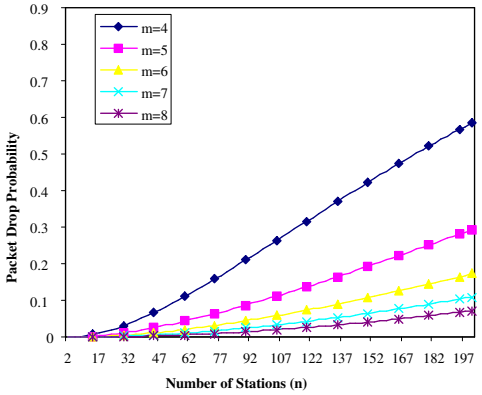

Fig. 9. Drop probability under variable retry limit m

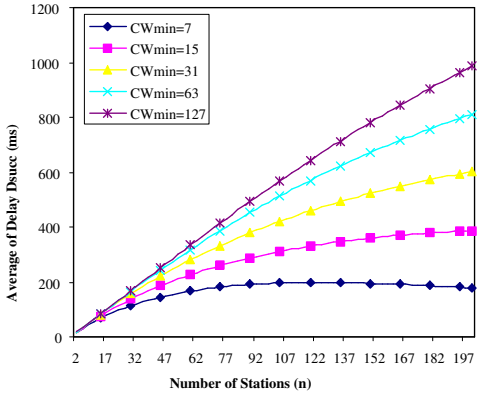

Fig. 10. Average delay D_{succ} under variable CW_{min}

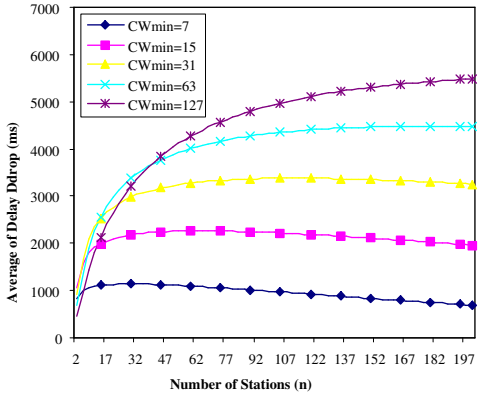

Fig. 11. Average delay D_{drop} under variable CW_{min}

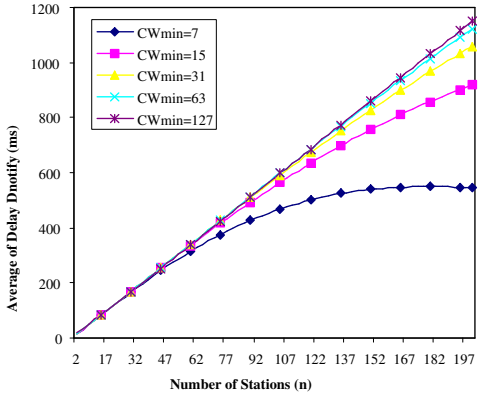

Fig. 12. Average delay D_{notify} under variable CW_{min}

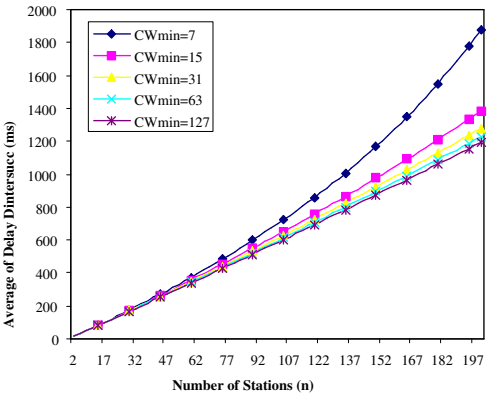

Fig. 13. Average delay $D_{intersucc}$ under variable CW_{min}

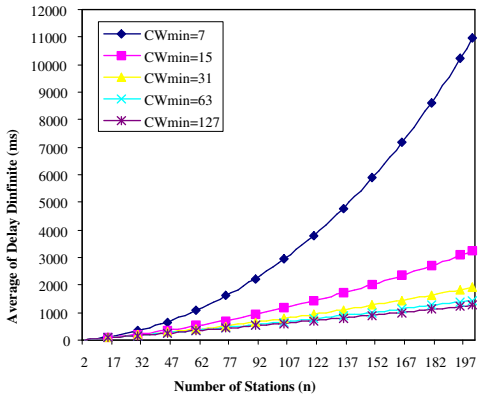

Fig. 14. Average delay $D_{infinite}$ under variable CW_{min}

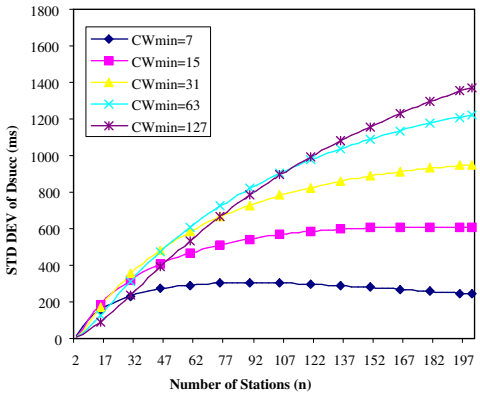

Fig. 15. Standard deviation of D_{succ} under variable CW_{min}

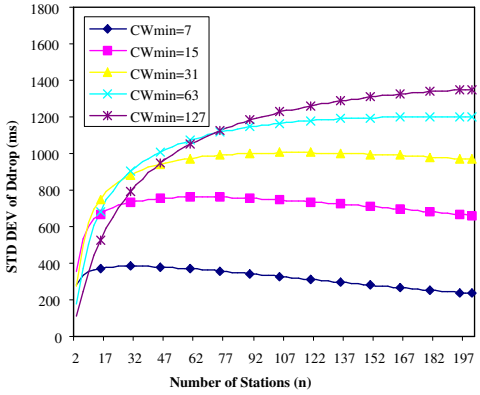

Fig. 16. Standard deviation of D_{drop} under variable CW_{min}

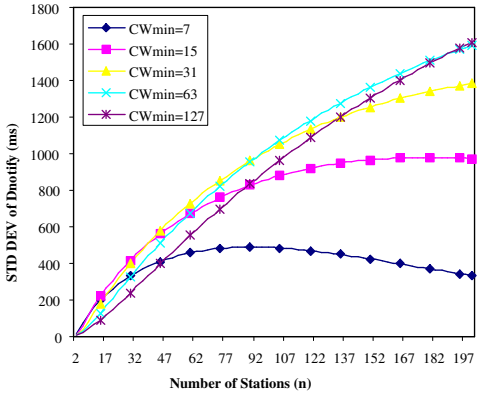

Fig. 17. Standard deviation of D_{notify} under variable CW_{min}

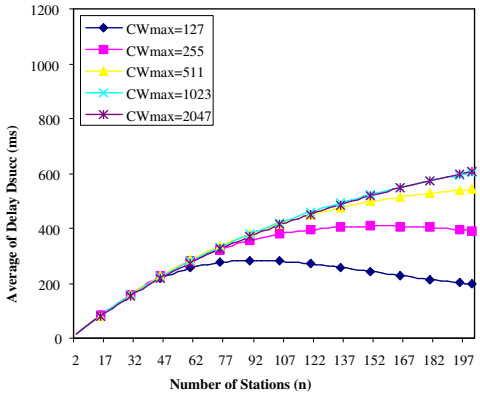

Fig. 18. Average delay D_{succ} under variable CW_{max}

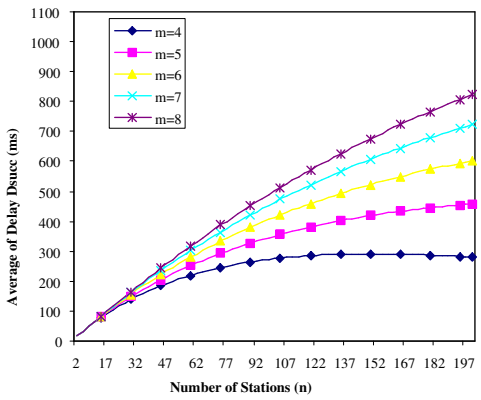

Fig. 19. Average delay D_{succ} under variable m

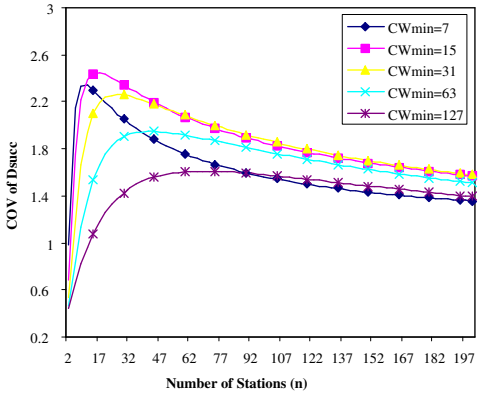

Fig. 20. Fairness under variable CW_{min}

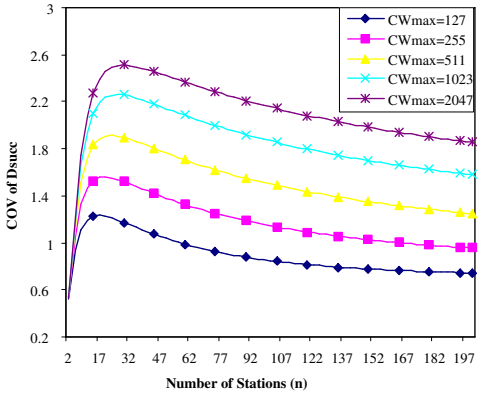

Fig. 21. Fairness under variable CW_{max}

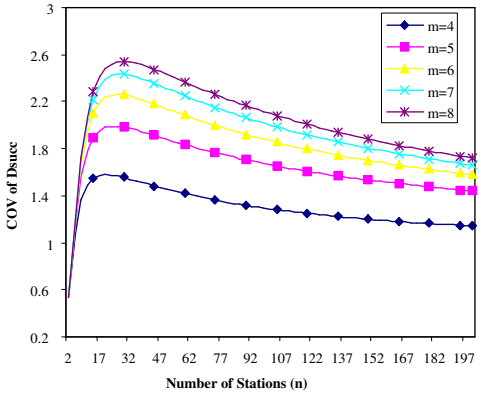

Fig. 22. Fairness under variable retry limit m

and m are, the more fair the system is; (ii) As the network size (i.e., n) increases, first the system becomes more unfair, and then after a certain value of n it becomes more fair.

We now explain the first observation. In IEEE 802.11, whenever a station transmits a packet successfully, it resets its contention window to CW_{min}. Clearly, if CW_{min} is small, the station that has just transmitted a packet successfully is more likely to get control of the medium again. On the other hand, if CW_{min} is large, it gives other stations more opportunity to transmit on the medium, explaining why the system becomes more fair when CW_{min} is larger. Now we consider the fairness with respect to CW_{max}. Clearly, the contention window at a station will become CW_{max} only when the station has experienced collisions successively. Therefore, if CW_{max} is large, the colliding stations, though they have already suffered a lot, will be more unfairly treated, explaining why the system is more unfair when the CW_{max} is larger. As for the retry limit m, if it is small, a station that has experienced collision(s) will drop the packet early and will reset its CW to CW_{min}, and then will schedule the transmission of a new packet. On the contrary, if m is large, the station may have to schedule the retransmission of a packet more number of times, which leads to a larger contention window, and in turn, short-term unfairness. Therefore, in this manner, the early dropping of a colliding packet contributes to the fairness, though it certainly results in more number of collisions as the stations become more aggressive. This explains why the system is more fair when m is small. From observation (i), we may reach an interesting conclusion regarding the back-off policy: to achieve better fairness, we should always use a medium-size CW without any retrial for the unsuccessful transmissions.

For the second observation, we first explain the reason why the system becomes more unfair when we begin to increase n (e.g., from 2 to 17). This is due to the randomness involved in the binary exponential back-off (BEB) algorithm. In particular, some stations may experience collisions consecutively while other stations can transmit packets without experiencing any collision. With the increase of n, collisions are more likely to happen, explaining the increasing amount of unfairness. In fact, we have also observed this phenomena in the simulation results presented in [7]. However, when the collision probability increases beyond a certain value along with the increase of n, most of the stations will experience collisions before they can transmit any packet successfully, and therefore, it tends to become more fair for all the stations. In the extreme case, when the number of stations becomes infinite, the collision probability tends to unity, and no station can transmit any packet successfully. Therefore, every station has zero throughput (or infinite delay), implying absolute fairness among the contending stations!

E. Summary of the Performance Evaluation

In this subsection, we give a brief summary of the performance of IEEE 802.11 in single-hop networks. With respect to the throughput, we find that it is quite sensitive to the chosen system parameters. This would be particularly true in the emerging IEEE 802.11 standards. Therefore, dynamic tuning of the system parameters is of great interest to achieve optimal performance. We also find that the throughput calculated in the model of [1] is higher than that in our model, which is more precise. As for the delay, we show that the first four kinds of delay under the finite retrials condition have very distinct values, showing the importance of distinguishing among them. We also find that the $D_{infinite}$ (i.e., delay under the infinite retrials assumption) has a large error, and thus leads to misleading observations. For the fairness performance, we get an interesting conclusion about the back-off policy: to achieve better fairness, a medium-sized contention window should be used without any retrial of the unsuccessful transmissions.

Table II summarizes how the system parameters should be selected (in terms of "large" or "small") to achieve the desired performance (i.e., higher throughput, smaller delay, and better fairness). By "smaller delay", we mean a smaller value of the average as well as the standard deviation of D_{succ}, D_{drop}, and D_{notify}. As shown before, when D_{succ}, D_{drop}, and D_{notify} are smaller, $D_{intersucc}$ and $D_{infinite}$ are larger. Since $D_{intersucc}$ is directly reflected by the throughput and $D_{infinite}$ is misleading, we do not need to consider these delays when optimizing the delay performance. Also, since a given set of system parameters, which yields a higher throughput, results in a smaller value of the packet drop probability, we do not need to present the desirable system parameters for a smaller packet drop probability.

From the table, we find that a set of system parameters delivering a good performance for one metric, may deliver a bad performance in terms of another metric. For example, if we aim to achieve a higher throughput, the table shows that we should use a large value of CW_{max} and m. However, the table also tells us that we should use a small value of CW_{max} and m if we aim to achieve better fairness. In other words, a trade-off among the metrics must be made in the dynamic tuning of the system parameters.

TABLE II
INTERDEPENDENCE BETWEEN PERFORMANCE AND PARAMETERS

Parameter \ Metric	Higher Throuhgput	Smaller Delay	Better Fairness
CW_{min}	Large	Small	Large
CW_{max}	Large	Small	Small
m	Large	Small	Small

V. RELATED AND FUTURE WORK

A. Related work

In the literature, many analytical models have focused on the calculation of the throughput (or capacity) of IEEE 802.11. For example, paper [2] derives the theoretical throughput of IEEE 802.11 by relating IEEE 802.11 to a p-persistent CSMA protocol. Paper [9] derives the throughput of IEEE 802.11 by using average analysis technique. However, the above models have oversimplified the calculation of the transmission probability τ, which is determined by the BEB algorithm. In

contrast, papers [1] and [12] derive a more precise value of τ by modelling the stochastic process representing the back-off time counter as a discrete-time Markov chain.

While most of the published models focus on the throughput of IEEE 802.11, two recent work [3], [10] concentrate on the delay analysis. However, neither of them have considered the retry limit, resulting in a large error as shown in this paper. Also, they do not distinguish the five kinds of delay defined in this paper, which have very different values.

Though many papers (e.g., [8], [11]) have focused on achieving fairness in IEEE 802.11, there are very few analytical models to quantify the unfairness in IEEE 802.11. We are only aware of one model [6] that analytically shows the short-term unfairness in IEEE 802.11 due to the hidden-terminal problem. However, the above model is very specific to the topology being given and thus it is not clear whether the model can be extended into a general single-hop network with a large number of stations.

B. Future Work

Motivated from the results obtained in this work, one main future work is to develop a dynamic tuning mechanism, which can achieve a reasonable tradeoff among the performance metrics. One possible method is as follows. We should first define a function, which characterizes the weights of the performance metrics according to the user preference. Using such a function, under any given network size (i.e., n), we can calculate the corresponding optimal system parameters (i.e., CW_{min}, CW_{max}, and m) that maximize the preferred function. This can be easily achieved using a table, which stores the optimum values of the systems parameters for different network sizes and for different weights that could be used in defining the function. The table can be prepared *off-line*. The estimation of the network size can be achieved by using the methods in [2], [7].

Another important future work is to extend the analysis by relaxing the assumptions made at the beginning of Section III. While the analysis should be easily extended when the assumptions (2)-(5) are relaxed as discussed in [10], [1], it may not be trivial to extend the model for a multi-hop network.

VI. Conclusions

In this paper, we have proposed a simple analytical model to evaluate the throughput, delay, and fairness performance in single-hop IEEE 802.11 networks. Using the proposed model, we have carried out an extensive performance evaluation of IEEE 802.11 under three different system parameters (i.e., CW_{min}, CW_{max}, and retry limit m). Our results show that the throughput, delay, and fairness are quite sensitive to these system parameters, and thus dynamic tuning of the parameters is of great interest to achieve the optimal performance. Moreover, the optimal values of the parameters may not be consistent with each other for different performance metrics, implying that some tradeoff among the metrics must be made in the dynamic tuning. Finally, our results also show that the published models have overestimated the throughput as well as the packet delay.

References

[1] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," IEEE Journal on Selected Area in Communications, March 2000, pp.535-547.

[2] F. Cali, M. Conti, E. Gregori, "Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit," IEEE/ACM Transactions on Networking, December 2000, pp.785-799.

[3] M. M. Carvalho, J. J. Garcia-Luna-Aceves, "Delay Analysis of IEEE 802.11 in Single-hop Networks," in IEEE ICNP, 2003.

[4] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," IEEE 802.11 standards, June 1999.

[5] R. Jain, D. Chiu, W. Hawe, "A Quantiative Measure of Fairness and Discrimination for Resouce Allocation in Shared Computer System," DEC Technical Report, 1984.

[6] Z.F. Li, S. Nandi, A.K. Gupta, "Modeling the Short-term Unfairness of IEEE 802.11 in Presence of Hidden Terminals," in IFIP Networking, 2004.

[7] Z.F. Li, S. Nandi, A.K. Gupta, "Achieving MAC Fairness in Wireless Ad-hoc Networks using Adaptive Transmission Control," in IEEE ISCC, 2004.

[8] T. Nandagopal, T. Kim, X. Gao, V. Bharghavan, "Achieving MAC Layer Fairness in Wireless Packet Networks," in ACM MOBICOM, 2000.

[9] Y. C. Tay, K. C. Chua, "A capacity analysis for the IEEE 802.11 MAC protocol," ACM Wireless Networks, 2, 2001, pp.159-171.

[10] O. Tickoo, B. Sikdar, "Queueing Analysis and Delay Mitigation in IEEE 802.11 Random Access MAC based Wireless Networks," in IEEE INFOCOM, 2004.

[11] N.H. Vaidya, P. Bahl, S. Gupta, "Distributed fair scheduling in a wireless LAN," in ACM MOBICOM, 2000.

[12] H.T. Wu, Y. Peng, K.P. Long, S.D. Cheng, J. Ma, "Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement," in IEEE INFOCOM, 2002.