Semi-Markov Process Formulation of IEEE 802.11 DCF Analytic Performance Models

Paolo Pileggi[1] and Pieter Kritzinger[2]

[1]Networking Group, Department of Telecommunications and Microelectronics, University of Rome (Tor Vergata), Italy
[2]Communications Research Group, Department of Electrical Engineering, University of Cape Town, South Africa
Email: psk@cs.uct.ac.za

Probably no protocol has been analyzed as much as the various versions of the IEEE 802.11 Distributed Coordination Function (DCF). In this paper, we show how to describe a typical DCF analytic model as a semi-Markov process (SMP). We do this at the hand of an example IEEE 802.11 network in Basic Access mode under both saturated and unsaturated traffic conditions. We show how to solve the corresponding SMP numerically. The accuracy of the technique is illustrated by comparing the results with that from the analytic model that is most frequently cited in the research literature.

Keywords: IEEE 802.11; Distributed Coordination Function; Performance Modeling; Semi-Markov process

Received 26 November 2010; revised 26 November 2010

1. INTRODUCTION

The IEEE 802.11 Distributed Coordination Function (DCF) has been analyzed in countless papers. Evidently the most-cited paper[3] in the literature of DCF modeling is that by Bianchi [1]. A reading of the 6 most-cited papers by Ziouva and Antonakopoulos [2], Zheng *et al.* [3], Dong and Varaiya [4], Ni *et al.* [5] and Daneshgaran *et al.* [6] reveals that the models in all of these papers are based upon a Markov chain (MC) that is some derivation of the original model by Bianchi.

In this paper, we show that the models mentioned above, and others, can be viewed as an instance of a much more general semi-Markov process (SMP). More significantly even, the solutions of the earlier Markov chain models invariably involve a complicated stochastic analysis with non-trivial mathematical formulation. We present an SMP model on the analysis of an example IEEE 802.11 DCF network and an efficient algorithmic solution of the model.

It is not the objective of this paper to present another DCF model but rather to show how previous MC-based analytic models can be re-interpreted as SMPs, thus allowing a much broader and more general perspective on the modeling of versions of the IEEE 802.11 standard, workloads and channel conditions.

In the next section, we describe the behavior of the example network we have chosen to illustrate the SMP modeling methodology. In Sect. 5, we present an algorithmic solution of the SMP of the network. Such an algorithm can be written for the solution of any SMP of a system in a stable state, which is what all MC models assume. In Sect. 4, we derive the basic normalized network throughput for our example network and, in Sect. 6, we compare the results with those of the original Markov model presented by Bianchi. Moreover, we show that it is straightforward to account for unsaturated traffic conditions and present results for 2 unsaturated traffic intensities.

2. SYSTEM BEHAVIOR

We describe an IEEE 802.11 system operating under unsaturated traffic conditions to illustrate the proposed methodology. As is usually the case, we make the assumption that the system is a fully-connected point-to-multipoint IEEE 802.11 network of homogeneous nodes. Each node n, of a total of N nodes, transmits data frames to a central access point (AP) according to the Basic Access (BA) mode of the DCF access mechanism.

As is often done [7, 8, 9, 10], we describe the system behavior from the perspective of an observer node, denoted n, which may be any one of the N nodes in the network.

We define the state of a node (and thus of the SMP,

[3]With 3917 citations at the time of writing

see Sect. 3.2), as the number of successive transmissions of the same frame. An observation point is set at the instant when a change of state, possibly a return to the same state, occurs.

At each observation point $1, 2, \ldots$, the observer records whether the channel was idle or busy during the previous observation period. As shown in Fig. 1, the wireless channel is thus considered to be alternating between consecutive, interleaved idle and busy periods, corresponding to the collective behavior of the N nodes. A frame is transmitted, either successfully or not, during a busy period. Failure would result if 2 or more nodes transmit simultaneously, referred to as a collision. In order not to complicate the SMP, we do not take failure as a result of an error on the wireless channel into consideration, although such an extension is straightforward.

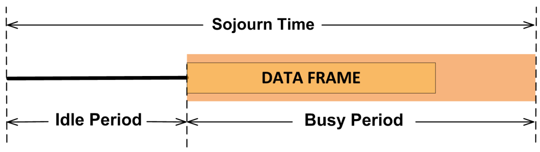

FIGURE 1. An instance of the channel alternating between idle and busy periods as one or more nodes transmit data

The DCF dictates that an individual node, ready to transmit a frame, first monitors the channel for a Distributed Interframe Space (DIFS) period, T_{DIFS}. If the node has a frame to transmit, the node decrements the back-off counter for a random number of slots. If the node has no frames to transmit, it remains idle for an empty time period.

The duration of the back-off period, BoV_n, is picked uniformly from an integer-valued interval called the *contention window*. The length of the contention window begins at CW_{min} and doubles in length after every failed transmission, up to a maximum length CW_{max}. This specific action attempts to avoid collisions, since other nodes may have been monitoring the channel for the same purpose.

If the channel is sensed busy during back-off, the back-off counter suspends and is reactivated after a successful transmission and after the channel has been sensed idle for a further DIFS period. When the back-off counter reaches zero, the node transmits the frame. If two or more nodes transmit simultaneously, a collision occurs. To resolve a collision, all nodes involved in the collision reset their back-off counters and restart back-off immediately. Counters of the other non-colliding nodes in the neighborhood suspend transmission for a period T_{EIFS} called the Extended Interframe Space (EIFS). Thus, colliding nodes have a greater probability of medium access for this post-collision period.

All frames are acknowledged within a Short Interframe Space (SIFS) period T_{SIFS}. If the sender does not receive an acknowledgment within the required time, it either reschedules transmission or drops the frame, depending on its retransmission (retry) count. The microsecond values for the slot duration σ and the Interframe Spaces are determined by the modulation scheme employed in the Physical Layer (PHY).

3. DCF BEHAVIOR AS A SEMI-MARKOV PROCESS

Although the following material about SMPs can be found in several textbooks (e.g., Bause [11]), we believe that a short introduction is useful for those readers not already familiar with the theory.

3.1. General semi-Markov theory

An SMP can be thought of as a process whose successive state occupancies are governed by the transition probabilities of a Markov process that spends time (the holding or sojourn time) in any state described by an integer-valued random variable that depends on the state currently occupied and on the state to which the next transition will be made. The state variable in our case is the number of consecutive transmissions of the same frame at any one node.

At the transition instants (the observation points described in Sect. 2), the SMP behaves just like a Markov process. We call this process the *embedded Markov process* and denote the single-step state transitional probabilities by $p_{ij},\ i, j = 0, 1, \ldots, R$. For simplicity, we assume that the process has a finite state space of size R that is in fact the maximum number of collisions allowed for any one frame. We also define τ_{ij} as the sojourn time in state i before the transition to the next state j.

The sojourn times with expected value $\overline{\tau_{ij}}$ are positive, integer-valued random variables (with unit time σ, the IEEE 802.11 slot time), each governed by a probability mass function $s_{ij}(m),\ \ m = 0, 1, \ldots$, called the sojourn time mass function for a transition from state i to state j. That is,

$$P[\tau_{ij} = m] = s_{ij}(m), \quad m = 0, 1, \ldots; \ i, j = 0, 1, \ldots, R \tag{1}$$

$$\overline{\tau_{ij}} = \sum_{m=0}^{\infty} m s_{ij}(m) \tag{2}$$

In other words, to fully describe a discrete-time SMP, we must specify R^2 holding time mass functions, in addition to the single-step transition probabilities.

We next define the *waiting time* τ_i with expected value $\overline{\tau_i}$ as the time spent in state i, $i = 0, 1, \ldots, R$, irrespective of the successor state and we define the probability mass function of this waiting time as

$$P[\tau_i = m] = \sum_{j=0}^{R} p_{ij} s_{ij}(m) \tag{3}$$

and

$$\overline{\tau_i} = \sum_{j=0}^{R} p_{ij}\overline{\tau_{ij}} \quad (4)$$

That is, the probability that the system will spend m time units in state i if we do not know its successor state, is the probability that it will spend m time units in state i if its successor state is j, multiplied by by the probability that its successor state will indeed be j and summed over all possible successor states.

Finally, let vector $\varphi = (\phi_0, \phi_1, \ldots, \phi_R)$ denote the steady-state probabilities ϕ_r of the SMP in state r and let $\Pi = \{\pi_0, \pi_1, \ldots, \pi_R\}$ be the steady-state probability vector of the equivalent embedded MC.

One can show (e.g., Howard [12]) that

$$\phi_r = \frac{\pi_r \overline{\tau}_r}{\sum_{i=0}^{R} \pi_i \overline{\tau}_i} \quad (5)$$

or

$$\varphi = \frac{1}{\overline{\tau}} \Pi M \quad (6)$$

where we have written

$$\overline{\tau} = \sum_{r=0}^{R} \pi_r \overline{\tau}_r \quad (7)$$

and M is the diagonal matrix $[\overline{\tau}_r]$ of mean waiting times.

Intuitively, the probability ϕ_r of finding the SMP in state r is the product of the average waiting time $\overline{\tau}_r$ in that state, multiplied by the probability π_r of being in that state in the embedded MC and normalized to the mean total time $\overline{\tau}$ spent in all of the R possible states.

In Sect. 4, we show that the values of $\overline{\tau}_r, \phi_r$ and the transition probabilities are sufficient to derive common performance metrics for IEEE 802.11 networks.

3.2. SMP model for the example network

Assuming homogeneous node behavior and a perfect channel, the number of possible events in the network and the resultant states and state changes are the same for each node. Fig. 2 illustrates the SMP corresponding to the system behavior, described in Sect. 2. In the figure, the state variable r, $r = 0, 1, \ldots, R$, represents the number of transmissions of the same frame at a particular node, where $R \in \mathbb{R}^+$ is the installation specific retry-limit, i.e., the maximum number of consecutive frame retransmissions allowed, after which the frame is discarded, i.e., dropped. When in state R, the system will revert to state 0 in the case of either a failure or success of the last transmission.

While in state r, node n can experience the events and resultant transitions listed in Table 1. These transition definitions are similar to the definition of slot states by Malone *et al.* [13]. Different from those authors, we do not define an explicit idle state, rather the *idle* time is included in the sojourn time of a state.

4. CALCULATING COMMON PERFORMANCE METRICS

It is important to be reminded that IEEE 802.11 performance can be viewed from the perspective of a single node (sometimes referred to as a "de-coupled" node), or from the perspective of the transmission medium experiencing the collective behavior of all nodes on the network. To be specific, we refer to a *user performance metric* and a *channel performance metric*, respectively, in the sequel.

The most popular performance metric of a node in IEEE 802.11 DCF is the normalized throughput defined as the amount of useful data transmitted divided by the capacity of the medium. Referring to Fig. 2, this normalized mean throughput, denoted $\overline{\psi_n}$ of a single user n, is given by the probability of the SMP being in a state r, $r = 0, \ldots, R$, returning to state 0 after a successful transmission of the mean time to transmit the payload information $\overline{\ell_r}$, all as a ratio of the mean sojourn time $\overline{\tau_r}$ in that state:

$$\overline{\psi_n} = \sum_{r=0}^{R} \phi_r \times p_{r,0} \times \frac{\overline{\ell_r}}{\overline{\tau_r}}. \quad (8)$$

The normalized mean throughput $\overline{\psi}$ of the whole network of N nodes, each with the same behavior, is then given by

$$\overline{\psi} = N \times \overline{\psi_n}. \quad (9)$$

Another metric is the mean network (or channel) efficiency $\overline{\nu}$. That is, the percentage of time the channel is being used productively. From the SMP model in Fig. 2 and Table 1, we have

$$\overline{\nu} = \phi_0 \times p_{0,0}^{'s} + \sum_{i=0}^{R} \phi_i \times (p_{i,0} + p_{i,i}^{s}) \quad (10)$$

As is the intention of the paper, the metrics mentioned are not intended to be an exhaustive list but rather as typical examples of those that can be computed using the statistics obtained through the SMP and parameters of the model that can be computed, as we shall show in the following section.

5. STEADY STATE ANALYSIS OF THE SMP

In order to compute the metrics mentioned in the previous section, we need to determine the values of the basic variables $p_{i,j}$ and $\overline{\tau_{i,j}}$ $\forall i,j = 0,\ldots,R$. The probability mass functions $s_{ij}(m)$ of Eq. 2 are not known but we can determine the values of $\overline{\tau ij}$ empirically. Algorithm 1 is used to do this, assuring that the system reaches steady-state. In other words, the algorithm is executed until it converges and then the appropriate $p_{i,j}$ and $\overline{\tau_{i,j}}$ statistics are computed as shown in Table 2.

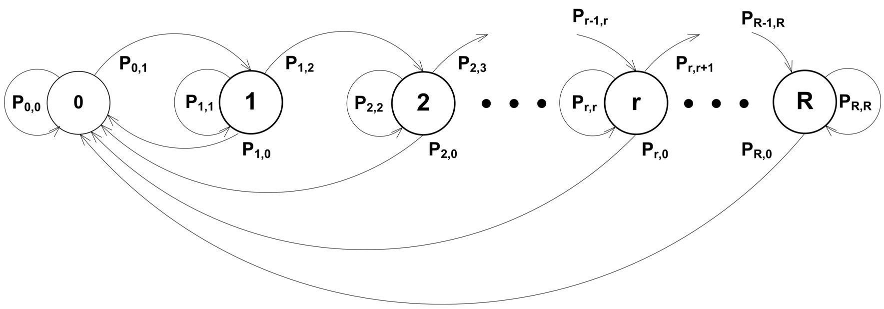

FIGURE 2. Semi-Markov chain for a typical node

TABLE 1. State transition descriptions

Transition	**Description**
$p_{r,0}$ where $r = 0, \ldots, R$	The probability that the current frame at node n is transmitted successfully, in which case the number of collisions returns to 0.
$p_{r,r+1}$ where $r = 0, \ldots, R-1$	The probability that the transmitted frame at n is involved in a collision, in which case node n must back-off by an adjusted random time and the number of transmissions increases by 1.
$p^s_{r,r}$ where $r = 0, \ldots, R$	The probability that the back-off process at node n is suspended due to a successful transmission by another node.
$p^c_{r,r}$ where $r = 0, \ldots, R$	The probability that the back-off process at node n is suspended due to a *collision* at nodes other than itself.
$p'^s_{0,0}$	The probability that there was no frame to send by the node during an observation interval while another node transmitted successfully.
$p'^c_{0,0}$	The probability that there was no frame to send by the node during an observation interval while two or more nodes suffered a collision.
$p'_{R,0}$	The probability that a frame at node n suffers a collision in state R, i.e., the drop probability.

The algorithm requires that one nominates a node to act as the wireless channel observer, called the *reference* node. Lines 16, 21, 36, 40, 56 and 58 show where the convergence statistics are recorded for the reference node. $p_{i,j}$ statistics, as defined in Table 1, are recorded with their corresponding $\tau_{i,j}$ values.

Lines 7 through 9 determine after how many slots the next transmission will occur and which nodes get to transmit next. Referring to Fig. 1 and Table 1, depending on the transition, the idle time is made up of one or more of the empty time, DIFS, EIFS and the back-off time (annotated as BoV).

The number of slots required to transmit during the next busy period t_{busy} is shown to be computed at line 10. Depending on whether the transmission was a success or not, t_{busy} takes on one of two values. These values are different for either BA and RTS/CTS modes.

At line 29, a new BoV is generated for a node that transmitted, whether successful or not. However, at line 33, the BoV is only updated to the difference between the amount of time it would have waited to transmit and the amount of time passed before another node transmitted during the observation period.

A new ET is generated for a node at lines 18 and 24, which is the case when the node enters state 0, i.e., a frame was transmitted successfully or dropped. The node is thus also reverted at these lines to state 0. In lines 43 through 47 and lines 49 through 53, ET is updated by the time elapsed since the last observation point, which includes t_{busy}.

When the node is in state R and it is involved in a collision, the probability of a node dropping a frame is recorded in the same way as for any other collision.

In order to verify that the values of the respective computed metrics converge, we computed the auxiliary variables defined below. The results shown in Table 2 are for a network operating in BA mode with an arbitrary number $N = 10$ contending nodes and where infinite retries are allowed, i.e., $R \to \infty$.

First, we write $\overline{\psi(i)}$ as the normalized mean throughput observed after each of i consecutive iterations of 10^6 steps of the algorithm, with $i = 1, \ldots, 20$. Furthermore,

$$\begin{aligned} \delta(i) &= \overline{\psi(i)} - \overline{\psi(i-1)} \\ \overline{\delta(i)} &= \frac{\delta(i)}{\overline{\psi(i)}} \end{aligned}$$

Next we form the cumulative sum $\zeta(i)$ of $\overline{\delta(i)}$ for $i,\ \ i = 1, \ldots, 20$, i.e.,

$$\begin{aligned} \zeta(i) &= \sum_{j=1}^{i} \overline{\delta(j)} \\ \overline{\zeta(i)} &= \zeta(i) - \zeta(i-1) \end{aligned}$$

Algorithm 1: Algorithm to derive values for the SMP parameters

```
begin
    Input: N, number of contending nodes; R, retry-limit;
    activityCount ← 0;
    ET_n ← compute(RANDOM) ∀ n ∈ {1, ..., N}; BoV_n ← compute(RANDOM) ∀ n ∈ {1, ..., N};
    previousWasSuccess ← true; dropOccurred ← false; involvedInPreviousTx_n ← false ∀ n;
    repeat
        slotsToNextTX_n ← compute(ET_n, DIFS, EIFS, BoV_n) ∀ n ∈ {1, ..., N};
        minSlotsToNextTX, minCount ← compute(slotsToNextTX_n);
        update(effective idle time accumulator);
        compute t_busy for current observation activity;
        previousWasSuccess ← (minCount == 1)?true : false;
        for n ∈ N do
            if slotsToNextTX_n == minSlotsToNextTX then                    /* node transmits */
                if minCount == 1 then                                     /* node transmits successfully */
                    if node is reference node then
                        record successful transmission by node: p_{r,0} and τ_{r,0};
                    end
                    move node_n to state 0; generate next ET_n;
                else                                                      /* node collides */
                    if node is reference node then
                        record collision by node: p_{r,r+1} and τ_{r,r+1};
                    end
                    if present_n = R then                                 /* drop occurred */
                        move node to state 0; generate next ET_n;
                    else                                                  /* node collides */
                        move node to state r + 1;
                    end
                end
                generate next BoV_n;
            else                                                          /* node does not transmit */
                set involvedInPreviousTx_n ← false;
                if ET_n ≤ minSlotsToNextTX + t_busy then                  /* node suspends */
                    update BoV_n to (slotsToNextTX_n − minSlotsToNextTX);
                    if minCount == 1 then                                 /* suspends due to successful TX */
                        if node is reference node then
                            record suspension by node due to another node transmitting successfully: p^s_{r,r} and τ^s_{r,r};
                        end
                    else                                                  /* suspends due to collision */
                        if node is reference node then
                            record suspension by node due to collision between other nodes: p^c_{r,r} and τ^c_{r,r};
                        end
                    end
                    if node is non-empty then
                        set ET_n ← 0;
                    else                                                  /* node is empty */
                        reduce ET_n by (minSlotsToNextTX + busy slots for successful or failed TX);
                    end
                else                                  /* node buffers were empty during entire observation interval */
                    if node is non-empty then
                        set ET_n ← 0;
                    else                                                  /* node is empty */
                        reduce ET_n by (minSlotsToNextTX + busy slots for successful or failed TX);
                    end
                    if node is reference node then
                        if successful transmission occurred by another node then
                            record empty time observation by node due to successful transmission by another node: p'^s_{0,0} and τ'^s_{0,0};
                        else                                              /* collision occurred between other nodes */
                            record empty time observation by node due to collision between other nodes: p'^c_{0,0} and τ'^c_{0,0};
                        end
                    end
                end
            end
        end
        activityCount ← activityCount + 1;
    until solution converges;
end
```

The values for these variables for i, $i = 1, \ldots, 20$, derived from executing the algorithm for 20×10^6 cycles are shown in Table 2.

Note that, in particular $\zeta(i)$, the cumulative sum of the relative difference in successive values of $\overline{\psi(i)}$, remains almost constant after $i = 9$. Since it is a numerical computation, one expects the final value to fluctuate around some steady state value as is indicated in the Table. The computations were done on a 2.5GHz Intel I5 processor and takes less than 10 seconds to converge to the values shown in the table. The Java code of the algorithm, comprising just over 450 lines of code, is available from the authors.

6. VERIFICATION OF THE SMP ANALYSIS

From the observed convergence behavior of the solver illustrated in Table 2, it is clear that the SMP is already in steady state after $2 \times 10^7 \times N$ state changes, where N nodes are contending to access the channel.

For a verification of our technique, we decided that comparing our results with those obtained by Bianchi and published in his seminal paper in 2000 [1] would suffice as verification of the accuracy of the SMP modeling approach. Not only is that approach by Bianchi widely accepted, but we were fortunate to be given access by the author to the numerical tool used to compute the results in his original paper.

Fig. 3 shows the plots for the BA case of saturated traffic for both the SMP approach and the results from the Bianchi paper mentioned. For interest, we added 2 more plots for various mean frame inter-arrival rates. The 2 traffic intensities, intensity 1 and 2, are characterized by a geometric distribution of the empty time with mean 15 and 30 slots, respectively. As one would expect, due to fewer collisions, for a larger

TABLE 2. Results showing consecutive iterations through Algorithm 1

Iteration i	$\overline{\psi(i)}$	$\overline{\delta(i)}$	$\zeta(i)$	$\overline{\zeta(i)}$
1	0.83573307	-	-	-
2	0.835337667	4.73E-04	4.73E-04	-
3	0.835479861	-1.70E-04	3.03E-04	1.70E-04
4	0.835579425	-1.19E-04	1.84E-04	1.19E-04
5	0.835498504	9.68E-05	2.81E-04	-9.68E-05
6	0.835476546	2.63E-05	3.07E-04	-2.63E-05
7	0.835479476	-3.51E-06	3.03E-04	3.51E-06
8	0.835469503	1.19E-05	3.15E-04	-1.19E-05
9	0.835390717	9.43E-05	4.10E-04	-9.43E-05
10	0.835337454	6.38E-05	4.73E-04	-6.38E-05
11	0.835351775	-1.71E-05	4.56E-04	1.71E-05
12	0.835379917	-3.37E-05	4.23E-04	3.37E-05
13	0.835348441	3.77E-05	4.60E-04	-3.77E-05
14	0.835324045	2.92E-05	4.89E-04	-2.92E-05
15	0.835328289	-5.08E-06	4.84E-04	5.08E-06
16	0.835327233	1.26E-06	4.86E-04	-1.26E-06
17	0.835312823	1.73E-05	5.03E-04	-1.73E-05
18	0.835322277	-1.13E-05	4.92E-04	1.13E-05
19	0.835323287	-1.21E-06	4.90E-04	1.21E-06
20	0.835327283	-4.78E-06	4.86E-04	4.78E-06

number of competing nodes, less traffic actually leads to better throughput when compared with the saturated case.

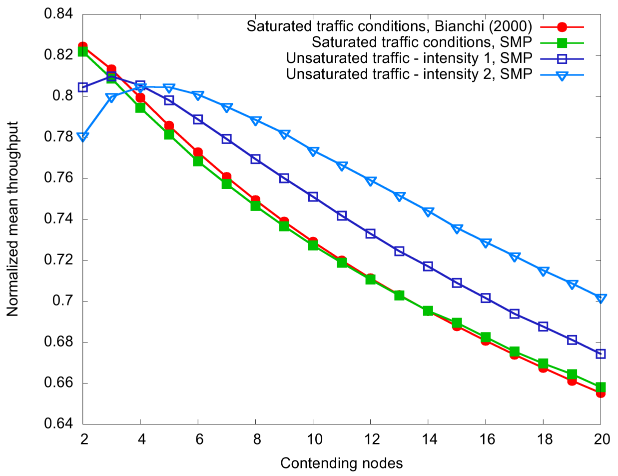

FIGURE 3. Normalized throughput $\overline{\psi}$ of the network in BA mode

The fact that increased collisions leads to lower throughput is further confirmed by the plots in Fig. 4 for the operation of the network in RTS/CTS mode.

In addition to the throughput results already presented, Fig. 5 shows the results for mean channel efficiency $\bar{\nu}$, as given by Eq. 10 where $\bar{\nu}$ is the of course the same for both operational modes. As expected, the probability of a transmission over the channel resulting in a success decreases from 1 (in the case of 1 contending node, i.e., no contention) as the number of nodes increases, since the probability of a collision increases. Comparing the results of saturated and unsaturated

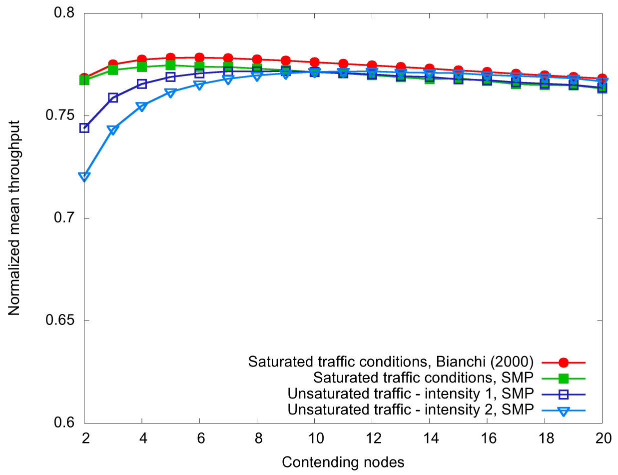

FIGURE 4. Normalized throughput $\overline{\psi}$ of the network in RTS/CTS mode

traffic conditions, it is evident that, when nodes are less likely to have data ready to send, collisions are less likely.

Other metrics that can be computed for the IEEE 802.11 DCF are not discussed since, as mentioned before, the intention of the paper is to present a methodology that coalesces earlier analytic models into a generalized stochastic model for modeling such systems, rather than to discuss the merits of the network protocol. Similarly, we give neither the SMP nor the algorithm for the results of the RTS/CTS operation, since both are easily determined from what we present for the BA case. Extending the SMP and solving it in the case of RTS/CTS and in fact, several other possible variations of an IEEE 802.11 network,

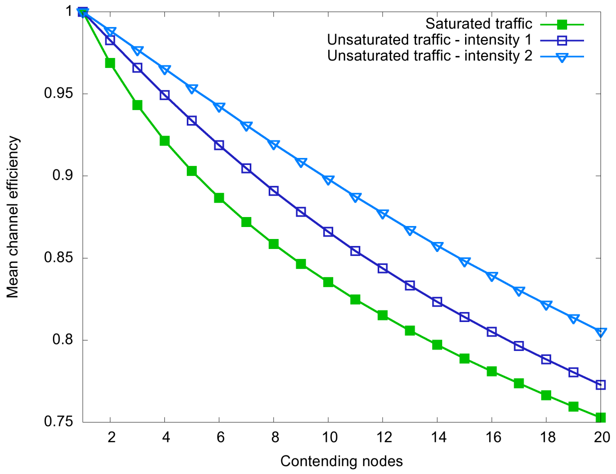

FIGURE 5. Channel efficiency $\overline{\nu}$ of the network

such as imperfect channel operation, are not difficult to do.

7. CONCLUSION

Each of the many analyses of the IEEE 802.11 DCF that exist in the literature is very specific about the DCF operation, the workload environment, the channel behavior, and so on. Most are based on a Markov chain model and use complex and often restrictive stochastic theory to determine an analytic solution.

In this paper, we generalize earlier analytic models into a general semi-Markov process structure that we then solve numerically. We verify the accuracy of this approach by comparing the numerical results with those of the first Markov chain model that was proposed and that has been imitated by many variations with their individual stochastic analyses.

In addition to its accuracy and simplicity, the semi-Markov process can be modified easily to cater for instances where the channel is imperfect, retries are limited, etc.

REFERENCES

[1] Bianchi, G. (2000) Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, **18**, 535–547.

[2] (2002) CSMA/CA performance under high traffic conditions: throughput and delay analysis. *Computer Communications*, **25**, 313–321.

[3] Zheng, Y., Lu, K., Wu, D., and Fang, Y. (2006) Performance Analysis of IEEE 802.11 DCF in Imperfect Channels. *Vehicular Technology, IEEE Transactions on*, **55**, 1648 –1656.

[4] Dong, X. and Varaiya, P. (2005) Saturation throughput analysis of IEEE 802.11 wireless LANs for a lossy channel. *Communications Letters, IEEE*, **9**, 100–102.

[5] Ni, Q., Li, T., Turletti, T., and Xiao, Y. (2005) Saturation Throughput Analysis of Error-Prone 802.11 Wireless Networks. *Journal of Wireless Communications and Mobile Computing, Wiley*, **5**, 945–956.

[6] Daneshgaran, F., Laddomada, M., Mesiti, F., and Mondin, M. (2008) Unsaturated Throughput Analysis of IEEE 802.11 in Presence of Non Ideal Transmission Channel and Capture Effects. *IEEE Transactions on Wireless Communications*, **7**, 1276–1286.

[7] Giustiniano, D., Malone, D., Leith, D. J., and Papagiannaki, K. (2007) Experimental assessment of 802.11 MAC layer channel estimators. *IEEE Communications Letters*, **11**, 961–963.

[8] Li, F., Li, M., Lu, R., Wu, H., Claypool, M., and Kinicki, R. (2006) Tools and Techniques for Measurement of IEEE 802.11 Wireless Networks. *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pp. 1–8. IEEE.

[9] Malone, D., Clifford, P., and Leith, D. J. (2007) MAC layer channel quality measurement in 802.11. *IEEE Communications Letters*, **11**, 143–145.

[10] Tinnirello, I. and Bianchi, G. (2009) Rethinking the IEEE 802.11e EDCA Performance Modeling Methodology. *IEEE/ACM Transactions on Networking*, **18**, 540–553.

[11] Bause, F. and Kritzinger, P. S. (2002) *An Introduction to the theory of Stochastic Petrinets*. Vieweg Verlag, Germany.

[12] Howard, R. (1971) *Dynamic Probabilistic Systems. Volume I: Markov Models. Volume II: Semi-Markov and Decision Processes*. John Wiley & Sons, Inc.

[13] Malone, D., Duffy, K., and Leith, D. (2007) Modeling the 802.11 distributed coordination function in nonsaturated heterogeneous conditions. *IEEE/ACM Trans. Netw.*, **15**, 159–172.