Timestepped Stochastic Simulation of 802.11 WLANs*

Arunchandar Vasan and A. Udaya Shankar
Department of Computer Science and UMIACS
University of Maryland
College Park, MD 20742

CS-TR-4866 and UMIACS-TR-2007-19
May 1st 2007

Abstract

We present Timestepped Stochastic Simulation (TSS) for 802.11 WLANs. TSS overcomes scalability problems of packet-level simulation by generating a sample path of the system state $\mathbf{S}(t)$ at time $t = \delta, 2\delta, \cdots$, rather than at each packet transmission. In each timestep $[t, t+\delta]$, the distribution $\mathsf{Pr}(\, \mathbf{S}(t+\delta) \,|\, \mathbf{S}(t)\,)$ is obtained analytically and $\mathbf{S}(t+\delta)$ is sampled from it.

Our method computes sample paths of instantaneous goodput $N_i(t)$ for all stations i in a WLAN over timesteps of length δ. For accurate modeling of higher layer protocols, δ should be lesser than their control timescales (e.g., TCP's RTT). At typical values of δ (e.g, 50ms), $N_i(t)$'s are correlated across both timesteps (e.g., a station with high contention window has low goodput for several timesteps) and stations (since they share the same media). To model these correlations, we obtain, jointly with the $N_i(t)$'s, sample paths of the WLAN's DCF state, which consists of a contention window and a backoff counter at each station.

Comparisons with packet level simulations show that TSS is accurate and provides up to two orders of magnitude improvement in simulation runtime. Our transient analysis of 802.11 complements prior literature and also yields: (1) the distribution of the instantaneous aggregate goodput; (2) the distribution of instantaneous goodput of a tagged station conditioned on its MAC state; (3) quantification of short-term goodput unfairness conditioned on the DCF state; (4) efficient accurate approximation for the n-fold convolution of the distribution of the total backoff duration experienced by a tagged packet; and (5) a simple closed form expression and its logarithmic approximation for the collision probability as a function of the number of active stations.

I. Introduction

Performance evaluation of computer networks has been based primarily on packet-level simulation models because analytical methods have typically not been able to capture state-dependent control mechanisms

* Supported in part by the Laboratory for Telecommunications Sciences under the UMIACS contract.

adequately. However, packet-level simulation becomes prohibitively expensive as link speeds, workloads, and network size increase. Timestepped Stochastic Simulation (TSS) is a method to achieve the modeling accuracy of simulation at a fraction of the computational cost. TSS generates **sample paths** of the network state, just as in packet-level simulation, but only at increments of discrete timesteps rather than at every packet transmission. If $\mathbf{S}(t)$ represents the network state at time t, TSS generates $\mathbf{S}(t)$ for $t = \delta, 2\delta, \cdots$, given $\mathbf{S}(0)$. In each timestep t, the distribution $\mathsf{Pr}(\, \mathbf{S}(t+\delta) \,|\, \mathbf{S}(t) \,)$ is obtained analytically assuming that all stochastic inputs are time-invariant in $[t, t+\delta]$, and $\mathbf{S}(t+\delta)$ is sampled from it. This time-invariance assumption holds for δ less than the feedback time-scale of the end-to-end control mechanisms employed, in which case TSS generates sample paths which are very good approximations of those generated by packet-level simulation. The time-invariance requirement means that δ cannot be too large; we use $\delta = 50$ms.

TSS has been developed for networks of point-to-point links [13], [12]. We present a method to perform TSS for 802.11 wireless networks (WLANs). The basic MAC protocol in all WLANs is the 802.11 Distributed Coordination Function (DCF) [16]. 802.11 DCF is a variant of Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA). It uses ACKs and performs retransmissions probabilistically to ensure successful delivery. Each station uses a time-based procedure to adapt its MAC state, and hence, its transmission attempts, to the current level of contention. The MAC state of station i is the tuple $\langle C_i(t), B_i(t) \rangle$, where $C_i(t)$ is the **contention window** and $B_i(t)$ is the **backoff counter** at time t. Station i continuously decrements $B_i(t)$ at the rate of one unit per slot, pausing only when the channel is sensed to be busy. The station transmits when $B_i(t)$ reaches zero. If the transmission is unsuccessful (i.e., ACK not received) $C_i(t)$ is doubled, otherwise $C_i(t)$ is reset to a specified initial value. In either case, a new value of $B_i(t)$ is chosen uniformly at random from $[0..C_i(t)–1]$. (An overview of the protocol can be found in Section II.)

Consider an 802.11 WLAN with α stations, where each station i is either active or inactive over time, with transitions occurring only at timestep boundaries. A station that is active (inactive) at time t has (no) packets to send in its output queue throughout $[t, t+\delta]$. (The output queue is fed, in general, by state-dependent data sources, e.g., TCP.) Let $\mathbf{M}(t)$ be the vector of length α whose ith entry is 1 if the station i is active and 0 if not, and $M(t)$ be the number of active stations. Let $N_i(t)$ be the *goodput* of station i in timestep $[t, t+\delta]$, defined as the number of packets successfully transmitted by station i in the timestep. $N_i(t)$ is zero for a station i inactive at t. For a station i active at t, $N_i(t)$ depends on all the active stations at t, as determined by the 802.11 MAC protocol.

Given $\mathbf{M}(t)$, our method computes evolutions of the goodput vector $\langle N_i(t){:}1{\leq}i{\leq}\alpha \rangle$ for $t = 0, \delta, 2\delta, \cdots$. For the timestep size δ of interest (i.e., $\delta = 50$ms), the DCF protocol introduces strong dependencies in the $N_i(t)$'s, specifically, positive correlation in $N_i(t)$ across timesteps and the negative correlation between $N_i(t)$'s across stations i in the same timestep. It is essential to capture these dependencies, otherwise the evolutions of the $N_i(t)$'s would not be an adequate foundation for simulating upper-level protocols (e.g., TCP) in a timestepped manner. Thus the key issue is the *short-term* behavior of the DCF protocol. Our method computes evolutions of the goodput vector and DCF state jointly: at each timestep, the goodput

and DCF state at the end of the timestep is obtained in terms of the goodput and DCF state at the previous timestep. We validate against PLS by comparing the resulting marginal distributions, the crosscorrelations (across stations), and the autocorrelations (across timesteps) of the per-station instantaneous goodput and DCF state. We find that TSS is quite accurate and yields runtime speed-up of up to two orders of magnitude.

To compute sample path evolutions of the goodputs and the MAC states, we need to probabilistically obtain $\{C_i(t+\delta), B_i(t+\delta), N_i(t)\}$ given $\{C_i(t), B_i(t)\}$ accounting for correlations both across stations and time. It turns out, however, that $B_i(t)$ can be approximated in terms of $C_i(t)$ as we explain later in Section IV. So we need to probabilistically obtain $C_i(t+\delta), N_i(t)$ given $C_i(t)$ for all i. Our method obtains this in the following steps:

- **Step 1:** Obtain the distribution $\mathsf{Pr}(\, N_A(t)\,)$ of the **aggregate goodput** $N_A(t) = \sum N_i(t)$, and sample $N_A(t)$ from it.

  We extend the analysis in reference [2] (which obtains the longterm average aggregate goodput) to show that $\mathsf{Pr}(\, N_A(t)\,)$ can be approximated by a normal distribution dependent on δ. We first characterize both the mean and the variance of the renewal period between two successes (prior works focus on the mean alone). The distribution follows from the central limit theorem for renewal processes. (Analysis in Section III.)
- **Step 2:** Obtain the distribution $\mathsf{Pr}(\, X\,)$ of the **total backoff duration** of a tagged packet, which is the total time spent in backoff by the packet's station during the packet's lifetime (from the start of the first transmit attempt until successful transmission or abort). (Analysis in Section V-A.)
- **Step 3:** For each station i, obtain $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ by abstracting the interaction with the rest of the stations by an average collision probability.

  We use the fact that $C_i(t)$ (stochastically) determines the instant t_s when the first successful transmission of station i occurs in $[t, t+\delta]$. Conditioned on t_s, the distribution of the number of successful packet transmissions in the interval $[t_s, t+\delta]$ is obtained by seeing how many total backoff durations (one for each successful packet transmission; aborts are explained later) can "fit" within this interval. Unconditioning on t_s gives $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$. (Analysis in Section IV. An efficient algorithm to obtain n-fold convolution of the total backoff duration is given in Section V.)
- **Step 4:** Dependently sample $N_i(t)$ from $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ for all i such that the sampled $N_i(t)$'s are correlated and $\sum N_i(t) = N_A(t)$.

  This step uses a randomized algorithm that enforces a negative correlation constraint among any subset of stations, in addition to the constraint that the samples add up to the sampled $N_A(t)$. (Explained in Section VI.)
- **Step 5:** For each station i, obtain $\mathsf{Pr}(\, C_i(t+\delta)|N_i(t), C_i(t)\,)$ and sample $C_i(t+\delta)$ from it.

  This distribution is obtained by accounting for the total backoff durations spent in the $N_i(t)$ successful transmissions in $[t, t+\delta]$. (Analysis in Section VII.)

Because all probability distributions involved can be parametrized in terms of the collision probability and the timestep length δ, they can be precomputed or cached across simulation runs.

A. Contributions

To the best of our knowledge, there has been no prior work on timestepped stochastic simulation of WLANs. There has also been no transient analysis of the 802.11 DCF performance. Specific contributions of our work are as follows:

- We present a transient analysis of 802.11 performance, yielding a method to generate sample paths of instantaneous metrics. Prior performance analyses (e.g., [3], [6], [9], [21], [19]) obtain the average aggregate steady-state goodput over a sufficiently long interval of time.
- We obtain the distribution of the instantaneous aggregate goodput $N_A(t)$ by obtaining both the mean and variance of the aggregate goodput renewal period.
- We obtain the distribution of the instantaneous goodput of a tagged station conditioned on its MAC state. This explains the short-term unfairness in instantaneous goodputs due to the 802.11 DCF backoff mechanism.
- We present an efficient algorithm to obtain the n-fold convolution of the distribution of total backoff involved in a packet's successful transmission or abort. Our results show how this seemingly long-tailed convolution can, in fact, be modeled well as a weighted combination of gaussian distributions.
- We obtain a closed form expression for the collision probability as a function of the number of stations for finite number of transmission attempts (extending the results in reference [2] which considers infinite number of retries) and present a simple logarithmic approximation for this function.

B. Roadmap

Section II introduces the notation, explains the operation of the DCF protocol, and states the modeling approximations. Section III obtains the distribution of the idle interval and from this, the distribution of instantaneous aggregate goodput. Section IV obtains the distribution of the instantaneous goodput of a tagged station in a timestep conditioned on its MAC state at the beginning of the timestep. Section V obtains the distribution of the total backoff duration of a packet and presents an algorithm to obtain its n-fold convolutions; this is used in Section IV. Section VI explains how we obtain correlated samples of $N_i(t)$. Section VII explains how we obtain the new MAC state given the old MAC state and goodputs from the previous timestep. Section VIII puts all the pieces of analysis together into the TSS simulator. Section X surveys related literature. Section XI discusses proposed extensions and Section XII concludes.

II. Overview of 802.11 DCF

An 802.11 network evolves in slotted time (of $9\mu s$ slots for 802.11a). Each evolution of basic 802.11 DCF (i.e., no RTS/CTS) consists of a sequence of successful or unsuccessful (collision) transmission intervals

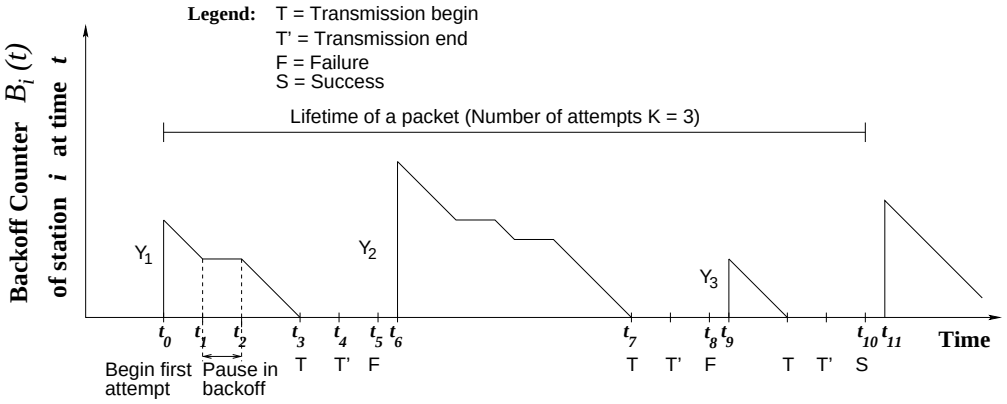


Fig. 1. Evolution of $B_i(t)$ during a packet's lifetime at station i. $C_i(t)$ changes at t_5 and t_8 to 2γ and 4γ respectively.

Symbol	Stands for
α	number of stations
β	retry limit
γ	initial contention window size
PKT	time to transmit a packet
ACK	time to transmit an ACK
$SIFS$	Short Inter-frame Spacing
$DIFS$	DCF Inter-frame Spacing
τ	transmission interval;
	equals $PKT + SIFS + DIFS + ACK$
For per-station attempt process:	
$C_i(t)$	contention window size of station i at time t
$B_i(t)$	backoff counter of station i at time t
$\langle C_i(t), B_i(t)\rangle$	MAC state of station i at time t
p	per-station collision probability
For a tagged packet of a station:	
K	number of transmission attempts
Y_j	backoff duration for j th attempt
X	total backoff duration; equals $Y_1 + Y_2 + \cdots + Y_K$
For aggregate attempt process:	
I	idle interval;
	variable interval between successive packet transmissions
	when all stations decrement their backoff counters
L	number of attempts for one successful transmission
p_A	aggregate collision probability

TABLE I
NOTATION

separated by variable idle intervals. A successful packet transmission has a **transmission interval** τ that consists of:

- The time to put the packet on the air (equals packet size divided by bitrate for data),
- The SIFS duration, which is the period separating a packet from its ACK transmission
- The time to put the ACK on the air (equals ACK size divided by bitrate for ACK), and
- The DIFS duration, which is the minimum period separating an ACK from the next data frame.

An unsuccessful transmission also has the same transmission interval τ. Specifically, stations respond to a collision as follows [16]:

- If a receiving station's physical layer deciphers an 802.11 packet with a checksum error from a signal resulting from the collision, then the station waits for an EIFS (Extended Inter-Frame Spacing defined to be SIFS + ACK + DIFS) after the end of the colliding transmissions before resuming its backoff.
- If a receiving station's physical layer cannot decipher any 802.11 frame (even with a checksum error) from the collision, then it waits only for DIFS after the end of the colliding transmissions before resuming backoff.
- A transmitting station always starts backing off only after DIFS + ACK Timeout (specified to be ACK + SIFS in the Systems Description Language appendix of [16]) irrespective of whether the transmission succeeded or failed.

In the case of receiving stations, we believe that the common case is the reception of a frame in error rather than the non-reception of any frame. So we choose the same transmission interval for both collision and success. References [9], [10], [4], [11] do the same. Some prior works (e.g., [3]) do not include the ACK time following a collision. Note that the use of RTS/CTS implies different transmission intervals for successful and unsuccessful transmissions.

In addition to the DIFS duration, the ACK of each transmission is separated from the next frame by a variable **idle interval** that is determined by the protocol operation as explained next.

A. Protocol operation

Figure 1 shows one possible evolution of the backoff counter $B_i(t)$ of a tagged station i that gets a packet to transmit at time instant t_0. The following steps occur:

- The MAC state $\langle C_i(t_0^-), B_i(t_0^-)\rangle$ just before t_0 (denoted by t_0^-) is the **idle state** $\langle 0, 0\rangle$.
- At t_0, the station chooses an initial backoff counter value Y_1 from $Uniform[0..\gamma-1]$. Thus the MAC state $\langle C_i(t_0^+), B_i(t_0^+)\rangle$ just after t_0 is $\langle \gamma, Y_1\rangle$.
- The station senses the medium. As long as the channel is idle, B_i is decremented at the rate of one per slot (in the figure, the decrease is shown as continuous). Whenever the medium is busy (due to another station transmitting), the decrementing is paused as shown between t_1 and t_2.
- At time $t = t_3$, $B_i(t)$ becomes zero and the station starts the transmission of the packet and finishes it at $t_4 = t_3 + PKT$

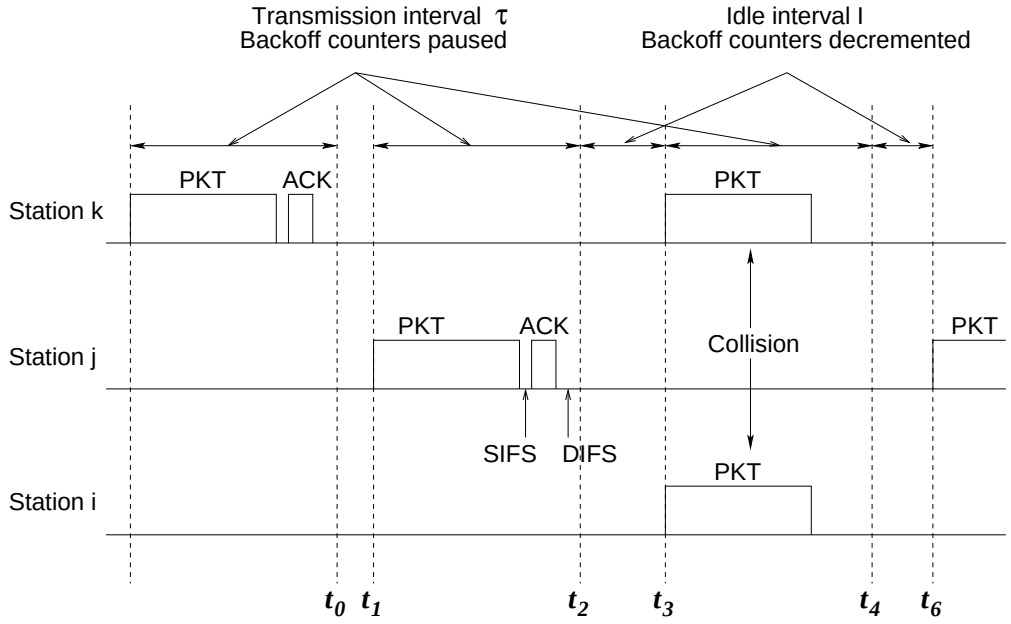


Fig. 2. Evolution of WLAN-wide transmissions, including a transmission by a tagged station i.

- No ACK is received within the standard timeout duration of *SIFS + ACK*. So at time $t_5 = t_4 +$ *SIFS + ACK*, the station doubles $C_i(t)$ to 2γ and chooses a new random backoff counter value Y_2 from *Uniform*$[0..2\gamma-1]$. This is the so-called Binary Exponential Backoff (BEB).
- The second attempt to transmit begins at time $t_6 = t_5 +$ *DIFS*.
- The second transmission starts at t_7 and is decided a failure at time t_8.
- The third attempt is successful at time instant t_{10} when it receives an ACK. At this point, the MAC state is reset to $\langle 0, 0\rangle$ if there are no packets to transmit. If there is a packet to transmit, C_i becomes γ and a new value for B_i is chosen from *Uniform*$[0..\gamma-1]$.

If successful transmission does not occur within β attempts, then the packet is **abort**ed and the MAC state is reset to $\langle 0, 0\rangle$. Thus, for evolution of the MAC state, an abort is equivalent to a success. The **lifetime** of a packet refers to the time elapsed from the start of the first transmit attempt to the end of either its successful transmission or abort.

We refer to the sequence of transmission attempts of a station as its attempt process. Each station in the system executes the same protocol. So each station has its **per-station attempt process**. The superposition of the per-station attempt processes results in the WLAN-wide **aggregate attempt process** as shown in Figure 2. A collision occurs if two or more stations start transmission in the same slot. Because collisions waste the channel, DCF tries to minimize collisions by performing BEB.

Figure 2 shows the the WLAN-wide transmissions and the associated timing details during the interval $[t_0, t_6]$ of Figure 1. At t_0 station i gets a packet to transmit and starts the backoff procedure. From time t_1

Symbol	Stands for	Type within timestep
δ	timestep of TSS	constant
$\mathbf{M}$	bit-vector of active stations	constant
M	# of stations with packets to transmit, i.e., $\|\mathbf{M}\|$	constant
p	per-station collision probability	constant
p_A	aggregate collision probability	constant
I	generic idle interval	stationary random variable
η	equals $E[I]/(E[I]+\tau)$	constant
N_i	goodput of station i	random variable
$\mathbf{N}$	vector of N_i for all i	random variable
N_A	aggregate goodput, $\sum_i N_i$	random variable

TABLE II

NOTATION FOR TSS QUANTITIES DEFINED IN TIME INTERVAL $[t, t+\delta]$. ALL QUANTITIES TERMED CONSTANT CAN VARY ONLY AT THE BOUNDARIES OF TIMESTEPS. ALL QUANTITIES MEASURING TIME (δ, I, τ) ARE IN 802.11 SLOTS.

through t_2, station j transmits a packet, so backoff counters of all stations remain unchanged in the interval $[t_1, t_2]$. Finally station i makes the first attempt at time t_3 resulting in a collision. Station j transmits the next packet after t_6.

Now consider the variable period preceding a packet transmission (for instance, the period between t_2 and t_3) during which backoff counters of all stations are decremented. This variable period is called the **idle interval** and is denoted by I. The value of I before a transmission is determined by the minimum of the backoff counters of all stations at the end of the preceding transmission. Note that I does not include the fixed overhead *SIFS + ACK + DIFS* after each packet transmission.

B. Modeling assumptions

Notation used in TSS is shown in Table II. We assume the following within a given timestep:

- The number of stations with packets to transmit is constant and denoted by M.
- Each attempt by a tagged station is a collision with per-station probability p dependent only on M ("per-station" distinguishes this from the aggregate collision probability explained in Section III).
- The transmission interval of a collision is the same as that of a successful packet transmission.
- The per-station collision probability p is constant within a timestep and can be obtained as a function of M (either by our empirical model in Section IX-I or a fixed point iteration as in reference [2]).
- There are no aborts. For standard values of protocol parameters, the probability of an abort is p^β is negligible (e.g., < 0.007 for $p < 0.5$ and $\beta = 7$).
- Each idle interval is an IID copy of a stationary random variable I.

All quantities that are assumed constant within a timestep can change over the course of a TSS run at timestep boundaries. We assume the following across all timesteps for the entire run of a TSS simulation:

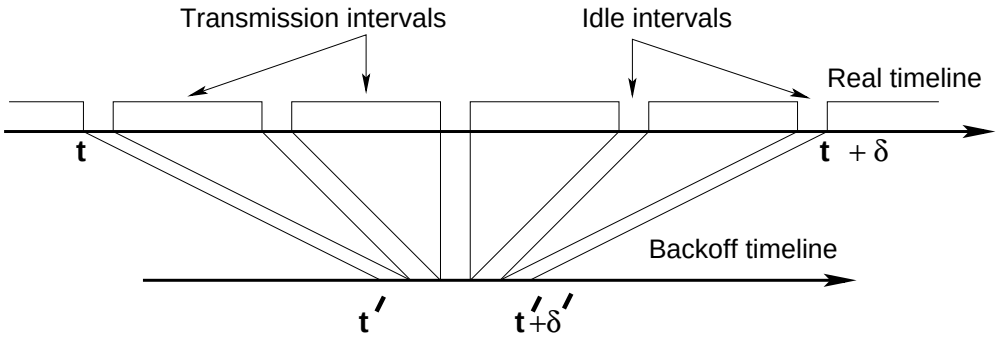


Fig. 3. Real-to-backoff-timeline contraction approximation. An interval $[t', t' + \delta']$ in the backoff timeline corresponds to an interval $[t, t + \delta]$ in the real timeline, where $\delta' = \eta\delta$ and $\eta \triangleq E[I]/(E[I] + \tau)$.

- RTS/CTS exchanges are not used.
- Every successful transmission is received at all stations (i.e., no hidden or exposed terminals) and all packets involved in a collision result in checksum errors at receivers (i.e., no physical layer capture).
- The transmission bitrates are constant.
- Packet size is constant.

Define the **backoff timeline** to be the sequence of all idle intervals ordered by their occurrence time. In other words, the transmission intervals in the real timeline are collapsed to points to obtain the backoff timeline. Figure 3 illustrates this real-to-backoff-timeline contraction approximation. Note that an interval of δ slots in the real timeline would on average have $\delta E[I]/(E[I] + \tau)$ idle interval slots. So the δ interval would on average correspond to an interval $\eta\delta$ in the backoff timeline, where $\eta \triangleq E[I]/(E[I]+\tau)$. To simplify the analysis, we assume that the variablity from the average is negligible. That is

- Any interval of length δ slots in the real timeline contracts (corresponds) to an interval of length $\eta\delta$ slots in the backoff timeline. (Section III-D justifies this in detail.)

III. The distribution $\mathsf{Pr}(\, N_A(t)\,)$

Recall that the per-station collision probability p is available as a function of M. Given this relationship, we obtain the distribution of the instantaneous aggregate goodput $N_A(t)$ in $[t, t+\delta]$ in terms of the number of active stations M, the transmission interval τ, and the timestep δ. In Sections III-A and III-B, we leverage results from reference [2]. Section III-A analyzes the per-station attempt process in the backoff timeline to obtain a tagged station's attempt rate λ in terms of p. This is done by considering the number of the transmission attempts K by the station for successful transmission of a tagged packet and the total backoff duration X in those attempts. Section III-B analyzes of the aggregate attempt process as the superposition of the per-station attempt processes to obtain the distribution of the idle interval I and the aggregate collision probability p_A.

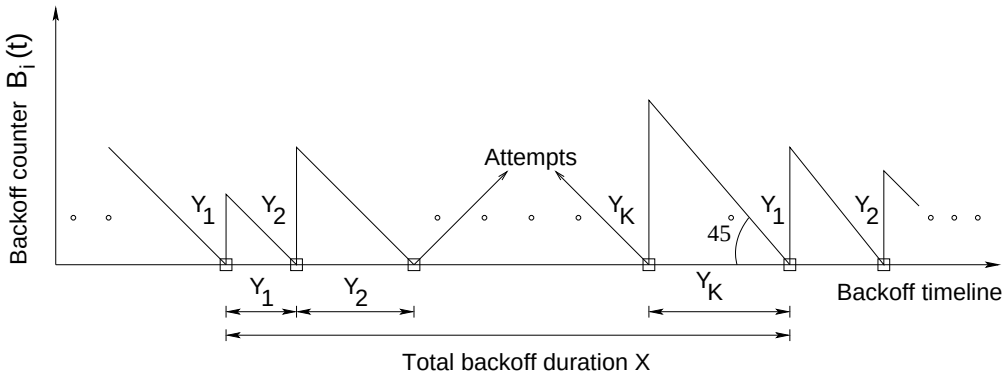


Fig. 4. Per-station attempt process of station i in backoff timeline driven by backoff counter $B_i(t)$. Attempts are made when $B_i(t)$ hits zero, and $B_i(t)$ is renewed according to the backoff process. The angle 45° indicates that $B_i(t)$ decreases with slope -1 everywhere except at the attempt points.

Then, in Section III-C, we present our analysis in the real timeline for the distributions of the instantaneous aggregate throughput and the instantaneous aggregate goodput $N_A(t)$. This is done by obtaining the moments of the throughput and goodput renewal periods and applying the central limit theorem for renewal processes. (Reference [2] obtains the mean of $N_A(t)$ alone.) Section III-D justifies the real-to-backoff-timeline contraction approximation for the aggregate idle interval in a timestep.

All analysis is within a timestep $[t, t+\delta]$. For sake of brevity, we omit the suffix "(t)" for time-dependent quantities henceforth unless essential for the discussion.

A. Analysis of per-station attempt process in backoff timeline

The per-station attempt process of a station i is driven by its backoff counter $B_i(t)$; transmissions occur whenever $B_i(t)$ reaches zero. Figure 4 shows the evolution in the backoff timeline of $B_i(t)$ of a station i attempting to transmit a tagged packet. On reaching zero, $B_i(t)$ is renewed according to the backoff process under our modeling assumption that each transmission results in a collision with probability p.

Thus the attempt process of a station i is a sequence of intervals with the pattern $\langle Block\rangle\langle Block\rangle\cdots$. Each $\langle Block\rangle$ is of the form $Y_1, Y_2 \cdots Y_K$ where

- there is a transmission after each Y_i;
- the transmission after Y_K alone is successful; and
- the total backoff duration X for a successful transmission is $Y_1 + Y_2 + \cdots + Y_K$.

In the backoff timeline, $B_i(t)$ is a markovian renewal process with average overall cycle (renewal) period $E[X]$ and average number of attempts in a cycle $E[K]$. By the renewal reward theorem [18], the attempt rate λ is given by $E[K]/E[X]$. In other words, the probability that a station transmits at the start of a given slot in the backoff timeline is λ.

$E[K]$ and $E[X]$ are calculated as follows. Each Y_i is chosen from $Uniform[0..\gamma 2^{i-1}-1]$, and so $E[Y_i] =$

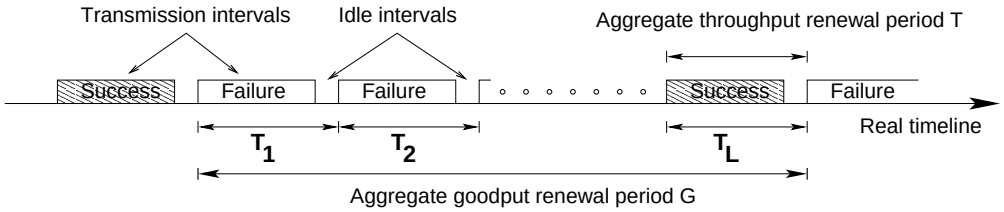


Fig. 5. Aggregate attempt process in the real timeline. Aggregate goodput renewal period G is the time between two successful transmissions.

$\gamma 2^{i-2} - 1/2$. We have $E[X] = \sum_{i=1}^{i=\beta} Pr(K = i).E[Y_1 + Y_2 + \cdots + Y_i]$. Because each attempt is Bernoulli with failure probability p, K is a truncated geometric random variable with the distribution

$$\begin{aligned} Pr(K = i) &= (1-p)p^{i-1} \qquad \text{for } \ 1 \le i < \beta \\ &= p^{\beta-1} \qquad\quad \text{for } \ i = \beta \end{aligned}$$

B. Analysis of aggregate attempt process in backoff timeline

The aggregate attempt process is the superposition of the per-station attempt processes. In the backoff timeline, the aggregate attempt process is a sequence of intervals with the pattern $\langle Block\rangle\langle Block\rangle\cdots$. Each $\langle Block\rangle$ is of the form $I_1, I_2 \cdots I_L$ where

- each I_i is an IID copy of the idle interval I;
- a single station transmits successfully after I_L; and
- two or more stations transmit unsuccessfully after each I_i for $i \neq L$.

We assume that the per-station attempt processes evolve independently in the backoff timeline though they evolve with the same p. Then the probability that there is a transmission in at least one of the M superposed per-station processes with attempt rate λ each is $1-(1-\lambda)^M$. Therefore the idle interval I is a geometric random variable with success probability $1-(1-\lambda)^M$.

An I_i is followed by a collision with aggregate collision probability p_A, which is the probability that two or more transmissions start in a slot given that there is at least one transmission. That is

$$p_A = \frac{1-(1-\lambda)^M - M\lambda(1-\lambda)^{M-1}}{1-(1-\lambda)^M}$$

Assuming that collisions are independent, L is geometric with success probability $1-p_A$.

Note that $p_A \neq p$, the per-station collision probability, because two or more frames collide in any collision. [To illustrate, suppose there are two active stations. Over some time interval, let s_1 and s_2 denote the number of packets transmitted successfully by the two stations and f, the number of collisions. Then $p \approx \dfrac{f}{s_1+f}$ and $p_A \approx \dfrac{f}{s_1+s_2+f}$. If $s_1 \approx s_2$ and $f \ll s_1$, then $p_A \approx p/2$.]

C. Analysis of aggregate attempt process in real timeline

The aggregate attempt process in the real timeline is a sequence of intervals with the pattern $\langle Block\rangle\langle Block\rangle\cdots$. Each $\langle Block\rangle$ is of the form $T_1, T_2, \cdots, T_L$, where each T_i is an IID copy of $T = I+\tau$ and only T_L is successful (as before). Thus the aggregate throughput process is a renewal process with period T. With $E[T] = E[I]+\tau$ and $Var[T] = Var[I]$, by the central limit theorem for renewal processes [18], we have

Theorem 1 *The aggregate throughput (i.e., number of throughput renewals including both successes and failures) in $[t, t+\delta]$ is normally distributed with mean $\delta/E[T]$ and variance $\delta\, Var[T]/E[T]^3$.*

The aggregate goodput process is a renewal process with period G corresponding to each $\langle Block\rangle$ of $T_1, T_2, \cdots, T_L$. Figure 5 shows the renewal period of the aggregate goodput and its relation to the aggregate throughput renewal periods. Observe that G is a compound random variable, i.e., a sum of a random number (L) of random variables (T_i). All prior work (e.g., [2], [3], [9]) compute $E[G]$ and thereby compute the mean goodput in $[t, t+\delta]$ as $\delta/E[G]$ from the renewal reward theorem. However, to obtain the distribution of $N_A(t)$, we need both $E[G]$ and $Var[G]$, which we obtain as follows:

$$\begin{aligned}
E[G] &= E[E[G|L]] \\
&= E[E[\sum_{i=1}^{i=L} T_i]] \\
&= E[L.E[T]] \qquad \text{(by independence of } T_i\text{'s and } L) \\
&= E[T].E[L] \\
E[G^2] &= E[E[(\sum_{i=1}^{i=L} T_i)^2]] \\
&= E[\,Var(\sum_{i=1}^{i=L} T_i) + E[\sum_{i=1}^{i=L} T_i]^2] \\
&= E[L\,Var[T] + (L.E[T])^2] \\
&= E[L].\,Var[T] + E[L^2]E[T]^2 \\
&= E[L]\,Var[T] + (\,Var[L] + E[L]^2)E[T]^2 \\
&= E[L].\,Var[T] + \,Var[L].E[T]^2 + E[G]^2
\end{aligned}$$

Therefore, we have $Var[G] = E[L]\,Var[T] + Var[L]E[T]^2$, which appeals to intuition in accounting for the variance in both L and T. Because L is a geometric random variable with success probability $1-p_A$, we have $E[L] = 1/(1-p_A)$ and $Var[L] = p_A/(1-p_A)^2$.

Theorem 2 *The random variable $N_A(t)$ is normally distributed with mean $\delta/E[G]$ and variance $\delta\, Var[G]/E[G]^3$.*

D. Real-to-backoff-timeline contraction approximation

Because each throughput renewal has an idle interval I that is geometrically distributed, the aggregate idle interval in $[t, t+\delta]$ would actually be the sum of a (normally distributed) random number of geometric random variables. One can compute the mean and variance of the total backoff compound random variable and approximate this by a normal distribution.

However, we approximate this random variable by a constant that is equal to its mean, namely, $\delta E[I]/(E[I]+\tau) \triangleq \delta\eta$. This contraction approximation greatly simplifies the presentation of the analysis for the per-station instantaneous goodput $N_i(t)$ (which would have otherwise needed conditioning and unconditioning on the aggregate idle interval in $[t, t+\delta]$). This assumption is justified because the deviation of the aggregate idle interval is very small relative to the mean (<8% for $\delta = 50$ms), which is because the deviation in the number of throughput renewals is not high relative to its mean. As can be seen from the results in Section IX, this approximation does not significantly compromise the accuracy.

IV. The distribution $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$

To obtain the distribution of $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$, we consider the per-station attempt process for station i in the backoff timeline; and obtain the required distribution in terms of the distribution of the total backoff duration X in a packet's lifetime. For the case $\langle C_i(t) = 0, B_i(t) = 0\rangle$, i.e., the station transmitted successfully just before t and starts attempts for a new packet just after t, we obtain $Pr(N_i(t) = n)$ as the probability of fitting n copies of X within a total backoff of $\eta\delta$ in the timestep. For an arbitrary starting state, we first obtain the distribution of X_f^*, the time to the first successful transmission in the interval conditioned on $\langle C_i(t), B_i(t)\rangle$. Conditioned on X_f^*, the distribution of $N_i(t)$ can be obtained by fitting copies of X in $\eta\delta - X_f^*$, as in the previous case. Finally, we uncondition on $B_i(t)$ to obtain distribution of $N_i(t)$ conditioned on $C_i(t)$ alone.

A. Obtaining the distribution $\mathsf{Pr}(\, N_i(t)|C_i(t) = 0, B_i(t) = 0\,)$

Figure 6 shows successful transmissions of the tagged station i in the corresponding interval in the backoff timeline given by $[t', t'+\delta\eta]$, which is the contraction of $[t, t+\delta]$. The backoff duration between two successful transmissions is X_i, an IID copy of the total backoff duration X in a packet's lifetime. There are n successful transmissions in the interval $[t', t'+\delta\eta]$ iff n IID copies of X when added is less than the total backoff duration $\delta\eta$ in the interval and the $n+1$th successful transmission occurs outside the interval.

Let E_1 denote the event $X_1 + \cdots + X_n \leq \eta\delta$ and E_2, the event $X_1 + \cdots + X_{n+1} \leq \eta\delta$. Clearly $E_1 \subset E_2$. Denoting the probability of n successful transmissions in a backoff timeline interval of length $\eta\delta$ as $h(n, \eta\delta)$,

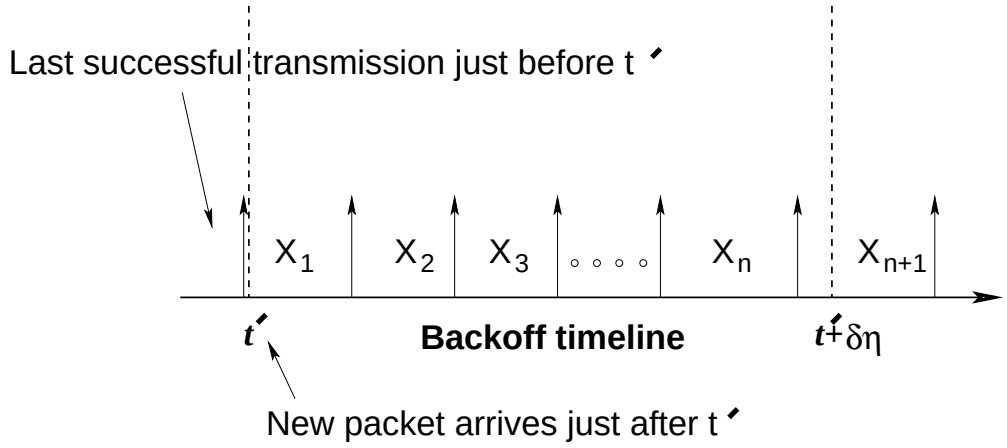


Fig. 6. Successful transmissions of a tagged station in the interval $[t', t' + \delta\eta]$ in the backoff timeline. The timestep $[t, t + \delta]$ in the real timeline is contracted to $[t', t' + \eta\delta]$ in the backoff timeline.

we have

$$\begin{aligned} h(n, \eta\delta) &= Pr(N_i(t) = n | C_i(t) = 0, B_i(t) = 0) \\ &= Pr(X_1 + \cdots + X_n \leq \eta\delta \wedge \\ &\quad X_1 + \cdots + X_{n+1} > \eta\delta) \\ &= Pr(E_1 \wedge \overline{E}_2) \\ &= Pr(E_1) - Pr(E_2) \quad (\text{since } E_1 \subset E_2) \\ &= F_X^n(\eta\delta) - F_X^{n+1}(\eta\delta) \end{aligned}$$

where the pdf f_X^n of the distribution $\mathsf{Pr}(\, X_1 + X_2 + \cdots + X_n \,)$ is the n-fold convolution of f_X with itself, and F_X^n denotes the corresponding cdf. Once f_X^n has been obtained (as described in section V), the pdf $h(n, \eta\delta)$ can be obtained as above. Note that $h(n, \eta\delta)$ depends solely on δ and η.

B. Obtaining the distribution $\mathsf{Pr}(\, N_i(t) | C_i(t) = \gamma 2^{c-1}, B_i(t) = b \,)$

We now obtain the distribution of $N_i(t)$ given an arbitrary starting state $B_i(t), C_i(t)$. Figure 7 shows the interval $[t', t' + \eta\delta]$ in the backoff timeline corresponding to the interval $[t, t + \delta]$ in the real timeline. At time t', the state is not $\langle 0, 0 \rangle$ and the first successful transmission occurs at t'_f. Define X_f^* to be the time to first success in the backoff timeline given $C_i(t), B_i(t)$, i.e., $X_f^* \triangleq t'_f - t'$. Conditioned on X_f^*, $Pr(N_i(t) = n)$ is given by $h(n-1, \eta\delta - X_f^*)$, the probability of $n-1$ successes in the backoff timeline interval $\eta\delta - X_f^*$ starting from the neutral state at t'_f. This is because the first successful transmission occurs at $t' + X_f^*$ and $n - 1$ more occur in the interval of length $\eta\delta - X_f^*$ with probability $h(n - 1, \eta\delta - X_f^*)$. So we want to obtain the pdf of backoff time to first success X_f^*.

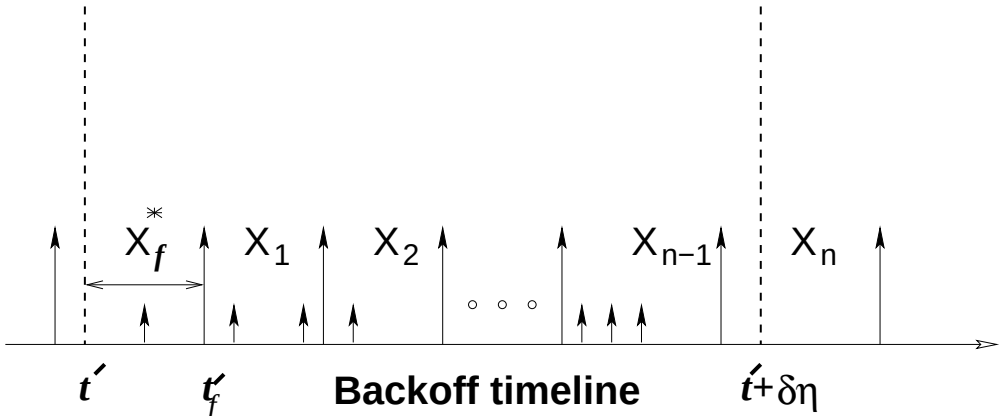


Fig. 7. Transmissions of a tagged station in the backoff timeline interval $[t', t' + \eta\delta]$ corresponding to real timeline interval $[t, t + \delta]$ when $\langle B_i(t), C_i(t)\rangle \neq \langle 0, 0\rangle$. A shorter arrow indicates a failure, a longer arrow success. The first successful transmission occurs at t'_f and X^*_f is the backoff duration $t'_f - t'$.

Given $C_i(t) = \gamma 2^{c-1}$ and $B_i(t) = b$, the first transmission occurs at $t' + b$ in the backoff timeline. The number of further attempts K (which can be zero) before a successful transmission at t'_f is distributed according to a geometric distribution with

$$Pr(K = i) = \begin{cases} (1-p)p^i & \text{for } i = 0 \leq i < \beta - c \\ p^{\beta - c} & \text{for } i = \beta - c \end{cases}$$

Therefore, the total backoff duration till t'_f is $b + Y_{c+1} + Y_{c+2} + \cdots Y_{c+K}$. Each of the Y_i's is uniformly distributed in increasing intervals and the number of attempts K is bounded by β and so this pdf can be obtained by straightforward convolution. In sum,

$$\begin{aligned} Pr(X^*_f = l | B_i(t) = b, C_i(t) = \gamma 2^{c-1}) &= \sum_{i=1}^{i=\beta-c} Pr(b + Y_c + \cdots + Y_{c+i} = l) Pr(K = i) \\ Pr(N_i(t) = n | X^*_f) &= h(n-1, \eta\delta - X^*_f) \\ Pr(N_i(t) = n | B_i(t) = b, C_i(t) = \gamma 2^{c-1}) &= \sum_{l=0}^{l=\eta\delta} Pr(X^*_f = l | B_i(t) = b, C_i(t) = \gamma 2^{c-1}) \times h(n-1, \eta\delta - l) \end{aligned}$$

C. Obtaining $Pr(N_i(t)|C_i(t) = \gamma 2^{c-1})$

If $C_i(t) = \gamma 2^{c-1}$, $B_i(t)$ was chosen from the $Uniform[0..C_i(t)-1]$ when it was renewed. Therefore at a given t, the distribution of $B_i(t)$ is distributed according to the forward recurrence time (or the remaining time) of the distribution $Uniform[0..C_i(t)-1]$. For a random variable $U \sim Uniform[0..a]$, the forward recurrence

time is a random variable U^+ whose distribution is given by

$$\begin{aligned} Pr(U^+ = k) &= Pr(U > k)/E[U] \\ &= \frac{(a-x)/(a+1)}{a/2} \\ &= \frac{2(a-x)}{a(a+1)} \end{aligned}$$

Thus we have $Pr(B_i(t) = b|C_i(t) = \gamma 2^{c-1}) = \frac{2(C_i(t)-b-1)}{C_i(t)(C_i(t)-1)}$ for $b \in [0, C_i(t)-1]$. We have obtained $P(N_i(t)|C_i(t), B_i(t))$ and $Pr(B_i(t)|C_i(t))$. Unconditioning on $B_i(t)$ gives $Pr(N_i(t)|C_i(t))$

D. Short-term unfairness in 802.11

Short-term unfairness in 802.11 has been the subject of much research [7], [8], [20], [22]. Reference [22] examines short-term unfairness for hidden terminals while references [8], [7] claim 802.11 is fair over intervals that are defined in terms of the number of inter-transmissions that other hosts may perform between two transmissions of a given station. Our analysis naturally yields a quantification of the short-term unfairness over arbitrary fixed intervals (δ here) even with no hidden terminals.

Consider a pair of tagged stations i and j among M active stations. Note that a difference between $C_i(t)$ and $C_j(t)$ automatically results in a difference in the means of $N_i(t), N_j(t)$. To quantify the extent of short-term unfairness in goodputs, we use Jain's fairness index J_F [17]. For two stations, $J_F(N_i, N_j)$ is defined to be $\frac{(N_i+N_j)^2}{2(N_i^2+N_j^2)}$ and ranges in $[1/2, 1]$, where $1/2$ corresponds to lowest fairness (one station gets all the goodput while the other gets nothing) and 1 corresponds to highest fairness (both get equal goodput). Specifically, we compute $E[J_F(N_i, N_j)]$ in two different ways: 1) by approximating the jdf of $\langle N_i(t), N_j(t)\rangle$ the product of the pdf's of $N_i(t)$ and $N_j(t)$, which are identical when unconditioned; and 2) by packet level simulations (PLS) described later in Section IX. Likewise, we compute $E[J_F(N_i, N_j)|C_i, C_j]$ analytically by approximating the jdf of $\langle N_i(t), N_j(t)\rangle$ given $\langle C_i(t), C_j(t)\rangle$ as the product of the pdf's of $N_i(t)|C_i(t)$ and $N_j(t)|C_j(t)$ and verify the analysis by simulations.

Values of $E[J_F(N_i, N_j)]$ are shown for varying M in Table III for 1) N_i, N_j unconditioned on C_i, C_j; and 2) conditioned on a fixed value of $C_i(t) = 16$ for varying $C_j(t)$. The value predicted by the analysis matches that obtained from PLS for the unconditioned Jain's index almost exactly. For the conditioned case, PLS results match the analysis almost exactly for small values of $C_j(t)$ (16, 256). However, for large values of $C_j(t)$ (512) the analysis overestimates the fairness. This is because the analysis allows N_j to be high (with some probability) jointly with high values of N_i due to the independence assumption. However, in reality, when N_i is high (which is likely due to low C_i), N_j is less likely to be high (due to negative correlation).

V. Obtaining f_X^n

To evaluate the pdf obtained in the previous section, we need the n-fold convolution f_X^n of the pdf f_X of the total backoff duration X in a tagged packet's lifetime (resulting in success or abort). We first obtain

M	C_i	C_j	$E[J_F(N_i,N_j)\|C_i,C_j]$	
			PLS	Analysis
4	Unconditioned		0.94	0.95
4	16	16	0.96	0.97
4	16	256	0.89	0.90
4	16	512	0.68	0.83
8	Unconditioned		0.83	0.84
8	16	16	0.91	0.92
8	16	256	0.83	0.82
8	16	512	0.60	0.74
16	Unconditioned		0.73	0.74
16	16	16	0.88	0.88
16	16	256	0.77	0.75
16	16	512	0.56	0.68

TABLE III

SHORT-TERM UNFAIRNESS ILLUSTRATED BY $E[J_F(N_i,N_j)]$ AND $E[J_F(N_i,N_j)|C_i,C_j]$ AS OBTAINED BY PLS AND ANALYSIS FOR VARIOUS VALUES OF M. FOR TWO STATIONS, JAIN'S FAIRNESS INDEX RANGES IN $[1/2,1]$ WHERE THE VALUE OF $1/2$ CORRESPONDS TO LOWEST FAIRNESS WHILE THE VALUE OF 1 CORRESPONDS TO HIGHEST FAIRNESS. THE EXTENT OF FAIRNESS VARIES DEPENDING ON THE CONTENTION WINDOW FOR CONDITIONED GOODPUTS.

f_X and explain why the structure of this pdf precludes a normal approximation to f_X^n. Then we present a simple and efficient convolution algorithm that exploits the structure of f_X to obtain f_X^n. The basic idea behind the convolution algorithm is to first approximate f_X as a weighted mixture of gaussians and then obtain f_X^n as a weighted mixture of gaussians efficiently using heuristics; the result is discretized to obtain the discrete pdf f_X^n.

A. The distribution of total backoff duration in a packet's lifetime $\mathsf{Pr}(\,X\,)$

Recall that for a tagged packet, K denotes the number of transmission attempts to success, and $Y_1, Y_2, \cdots, Y_K$ denote the backoff values chosen for those attempts. As seen in Section III, K is a truncated geometric variable with parameter p. Let $Z_i \triangleq Y_1 + Y_2 + \cdots + Y_i$ denote the total backoff duration if $K = i$. Then $f_X = \sum_{i=1}^{i=\beta} Pr(K=i).f_{Z_i}$. To obtain f_{Z_i}, we proceed as follows. Y_i is sampled from $Uniform[0..\gamma 2^{i-1}–1]$. Because Z_i is the sum of such uniformly distributed random variables, we approximate f_{Z_i} by the pdf of a normal distribution with mean m_i given by $\sum_{j=0}^{j=i} E[Y_j]$ and variance s_i^2 given by $\sum_{j=0}^{j=i} Var[Y_j]$. Because Y_1, Y_2, etc. have smooth uniform distributions, the normal approximation to Z_i works very well for $i > 1$ though the number i of random variables being added is small. Thus f_X can be written as $\sum w_i g_i(m_i, s_i)$, i.e., a weighted combination of gaussian pdf functions. Here each weight w_i is $Pr(K = i)$ and g_i is the pdf of a gaussian with mean m_i and s_i described before.

Figure 8 illustrates the accuracy of the approximation. It compares the pdf f_X obtained by the analytical approximation with that obtained by packet level simulation for a collision probability of 0.4. For the lowest

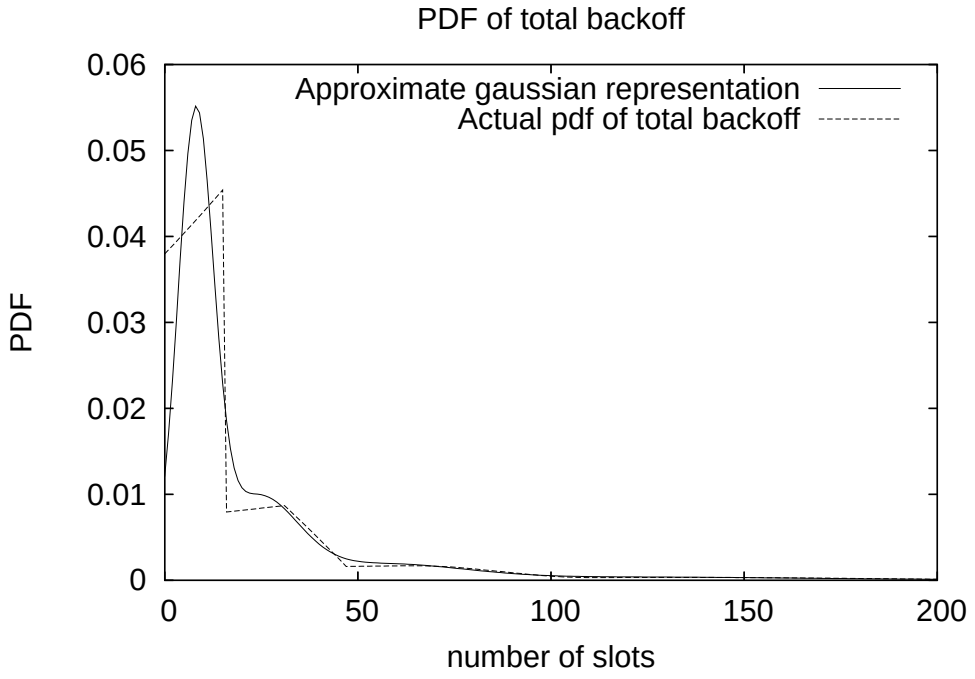


Fig. 8. Illustrating the accuracy of the weighted gaussian approximation to the pdf of the total backoff duration in a packet's lifetime X.

lobe, i.e., for $K = 1$ the approximation is not very accurate. However, this approximation suffices in practice in computing n-fold convolutions of f_X.

B. The impracticality of a simple normal approximation to f_X^n

A natural approach would be to use a normal approximation to the n-fold convolution of f_X. This would be similar to approximating the aggregate goodput $N_A(t)$ using the central limit theorem for renewal processes. Because X has finite support, both $E[X]$ and $E[X^2]$ are finite and therefore a normal approximation is theoretically feasible. However, *the convergence to normal is very slow for f_X^n* because of the "cascading" tail of the distribution. (For examples, see results in Section IX-J. The modes of f_X are approximately the $E[Z_i]$'s. $E[Z_i]$ grows exponentially with increasing i while the associated weight w_i shrinks exponentially, implying a power-law dependence between $E[Z_i]$ and w_i. Thus, the envelope of f_X at its modes can be thought of as a truncated Pareto distribution, and the sum of Pareto random variables can be approximated only by a Levy distribution [14], [15].) Therefore, we need another method of approximating f_X^n for our purposes.

In the case of $N_A(t)$ however, the aggregate goodput renewal period G, neglecting the contribution from the idle intervals is distributed as a well-behaved geometric random variable, which is why the normal approximation works well for $N_A(t)$.

C. Algorithm for obtaining convolution

Recall that the convolution of a normal distribution with mean m_1 and deviation s_1 with another of mean m_2 and deviation s_2 results in a normal distribution with mean $m_1 + m_2$ and deviation $\sqrt{s_1^2 + s_2^2}$. Because $f_X = \sum_{i=0}^{i=\beta} w_i g_i$, we have $f_X^2 = \sum_{i=0}^{i=\beta} \sum_{j=0}^{i=\beta} w_i w_j g(m_i + m_j, \sqrt{s_i^2 + s_j^2})$ to have β^2 normal terms. Likewise, f_X^n will have β^n terms in general. We need a way to evaluate this distribution efficiently. We observe the following:

- The weights w_i, being the probability of i consecutive losses, decrease exponentially with increasing i; Therefore, not all w_i are equally significant.
- The term-by-term convolution yields several gaussian terms whose means and deviations are close enough to be approximated by a single term which absorbs the weights of such close terms.

These two observations yield an efficient and accurate approximation of f_X^n as per Algorithm CONVOLVE.

The loop in lines 3 through 25 iterates to compute to f_X^{i+1} from f_X^i in two phases. In the first phase (lines 5-9), the convolution of terms f_X^i (stored as *curr-list*) with f_X (stored in *init-list*) is computed and stored in *new-list*. In the second phase (lines 11 through 25), *new-list* is shrunk. The shrinking process first sorts *new-list* in lexicographically increasing order according to the tuple (m_k, s_k) in line 11. The algorithm maintains $\langle w, (m, s)\rangle$ as the candidate entry to be added to *shrunk-list*. For each entry $\langle w_k, (m_k, s_k)\rangle$ in the sorted *new-list*, the following heuristics are used:

- If w_k is small compared to a threshold ϵ (typically, 0.001), the entry $\langle w_k, (m_k, s_k)\rangle$ is ignored by simply adding w_k to current candidate weight w. This is done in line 16.
- If w_k is significant and m_k and s_k are comparable to the current m and s values, then m and s are combined with m_k and s_k respectively after weighting by w and w_k. The value m is deemed comparable to m_k if $|m_k - m| < \theta m$, where $\theta < 1$ (typically, 0.1) is a small number. This is done in lines 18 through 20.
- If w_k is significant and $\langle w_k, (m_k, s_k)\rangle$ cannot be combined with $\langle w, (m, s)\rangle$ then $\langle w, (m, s)\rangle$ is added to *shrunk-list* and the shrinking continues with $\langle w_k, (m_k, s_k)\rangle$ becoming the new candidate *shrunk-list* entry. This is done in lines 22 and 23.

D. Runtime

We assume that n–1 fold convolutions have been computed and want to obtain the runtime of the n-th convolution. Recall that we start with f_X having β terms. In the worst case, the shrinking algorithm (depending on the tunable threshold θ) may not reduce any terms at all from the partial convolutions. However, in practice, we see that the shrinking algorithm keeps the number of terms in any partial convolution to be within $O(\beta)$. Under this assumption, the run-time for the n-th convolution is $O(\beta^2 \log \beta)$. If a is the number of discrete support points in f_X, f_X^n will have $n(a-1)+1$ points, which is $O(na)$. Discretizing the gaussian mixture approximation of f_X^n with worst case $O(\beta^2 \log \beta)$ gaussian terms over $O(na)$ points takes

CONVOLVE(f_X, n)

1 *init-list* $\leftarrow$ list of $< w_i, (m_i, s_i) >$ in f_X
2 *curr-list* $\leftarrow$ *init-list*, *count* $\leftarrow 0$
3 **while** (*count* $< n$)
4 *count* $\leftarrow$ *count* $+ 1$
5 *new-list* $\leftarrow \{\}$
▷ Phase-1: Obtain convolution of weighted
▷ gaussian sums by term-by-term convolution
6 **for** each $\langle w_i, (m_i, s_i)\rangle$ in *init-list*
7 **for** each $\langle w_j, (m_j, s_j)\rangle$ in *curr-list*
8 $m \leftarrow m_i + m_j$; $s \leftarrow \sqrt{s_i^2 + s_j^2}$; $w \leftarrow w_i \times w_j$
9 Add $\langle w, (m, s)\rangle$ to *new-list*
▷ Phase-2: Shrink the obtained result
10 *shrunk-list* $\leftarrow \{\}$
11 Sort *new-list* according to increasing (m, s)
12 $\langle w_0, (m_0, s_0)\rangle \leftarrow$ first(*new-list*)
13 $w \leftarrow 0$, $m \leftarrow m_0$, $s \leftarrow s_0$
14 **for** each successive $\langle w_k, (m_k, s_k)\rangle$ in *new-list*
▷ ϵ is a threshold
15 **if** $(w_k < \epsilon)$
16 $w \leftarrow w + w_k$
17 **elseif** $|m_k - m| \leq \theta m$
and $|s_k - s| \leq \theta s$
18 $p_1 \leftarrow w/(w_k + w)$; $p_2 \leftarrow w_k/(w_k + w)$
19 $m \leftarrow p_1 m + p_2 m_k$; $s \leftarrow \sqrt{p_1 s^2 + p_2 s_k^2 + p_1 p_2 (m - m_k)^2}$
20 $w \leftarrow w + w_k$
21 **else**
22 Add $\langle w, (m, s)\rangle$ to *shrunk-list*
23 $w \leftarrow w_k$, $m \leftarrow m_k$, $s \leftarrow s_k$
24 Add $\langle w, (m, s)\rangle$ to *shrunk-list*
25 *curr-list* $\leftarrow$ *shrunk-list*
26 **return** *curr-list*

$O(na\beta^2 \log \beta)$. The use of an FFT based convolution, which starts with a discrete representation of f_X over a points and computes the partial convolutions proceeding with a similar strategy would take $O(na \log na)$ time for the n-th convolution. If $\beta^2 \log \beta$ is $O(1)$ w.r.t. input size $O(na)$, then our approach reduces $O(na \log na)$ to $O(na)$.

E. Optimizations

Our optimizations are based on the observation that we are interested in the pdf's of the n-fold convolutions for support points lesser than $\delta\eta$, the total backoff in a timestep of length δ.

Suppose $F_X^{n^*}(\delta\eta) \approx 0$ for some n^*, i.e., the probability that $X_1 + X_2 + \cdots + X_{n^*}$ takes a value lesser than $\eta\delta$ is negligible, then the algorithm for computation of the convolution can be halted at n^* because for any $n > n^*$, $F_X^n(\eta\delta) \approx 0$ and does not give any more information required for obtaining $P(N_i|C_i)$ and the other pdf's for timesteps of length δ.

Another related optimization is to to represent the a n-fold partial convolution only up to an interval length of $\delta\eta$ rather than over the entire support range of $O(na)$. Thus this would help optimize both the gaussian approximation method as well as the FFT-based approach specifically for our case.

VI. Dependent sampling of $N_i(t)$'s

The goodput of an active station i in $[t, t+\delta]$ is determined by two factors: 1) its initial state $C_i(t)$; and 2) its interaction with all other active stations in $[t, t+\delta]$. If $C_i(t)$ is too high, with high probability, i will not attempt often enough to possibly get a high goodput. Likewise, if the goodputs obtained by other stations are high, then $N_i(t)$ will necessarily go down since there is only so much channel capacity in $[t, t+\delta]$. So far we have obtained $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$, which captures the effect of the first factor by approximating the interaction with all other stations by a constant per-attempt collision probability in $[t, t+\delta]$. If the $N_i(t)$ were independent of each other, all that needs to be done is to sample each $N_i(t)$ from the distribution $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$. However, in reality, the interactions within stations in $[t, t+\delta]$ ensures that $N_i(t)$ is correlated, even if very weakly, with every $N_j(t)$ for $i \neq j$. Further, because $C_i(t+\delta)$ depends on $N_i(t)$, the states of all stations are also weakly correlated. Thus we want a method that will sample $N_i(t)$'s from their conditional distributions in a manner that reflects their negative correlation.

A. The aggregate goodput constraint

We obtained the distribution of the aggregate goodput $N_A(t)$ independent of any constraint (even from $N_A(t-\delta)$). Therefore, in each timestep, we sample $N_A(t)$ from its distribution and require that any sampling of $N_i(t)$ should be such that they add up to the sampled $N_A(t)$. Note that if the $N_i(t)$ were chosen independent of each other, the variance of the sum would be cumulative and not be as low as $Var[N_A(t)]$, which is a result of the negative correlation. Clearly, the constraint $N_A(t) = \sum N_i(t)$ requires that the $N_i(t)$ be sampled in a way reflecting the negative correlation.

B. Preliminary approaches

We want to obtain $\mathbf{N}(t)$ where:

- each $N_i(t)$ is sampled from $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$;
- $\sum N_i(t) = N_A(t)$; and
- $N_i(t)$ are negatively correlated.

One option is to first obtain tentative samples $N'_i(t)$ from the respective distributions $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ independently and then obtain each $N_i(t)$ as $N'_i(t).N_A(t)/\sum N'_i(t)$. While this approach does handle negative correlation, the resulting distribution of $N_i(t)$ as obtained by TSS does not match the distribution of $N_i(t)$ obtained by PLS well. A second approach is to consider a random permutation π of the indicices of $\mathbf{M}$ that are 1's, i.e., the list of active stations. Suppose we sample $N_i(t)$ for each i in π from $Pr(N_i(t)|C_i(t)$ and assign the remainder to the station not seen so far. This ameliorates the bias in favor of stations with lower indices, but it makes the negative correlation between stations whose indices are considered last higher (because they have the lowest goodput to share). Further, it leaves open the possibility that all stations whose goodputs are chosen initially by the random permutation do not add up to a significant value thereby making the last station to be assigned have an arbitrarily large goodput.

C. Algorithm for sampling $N_i(t)$

The basic idea is to sample the goodput of a station from a "suitable" portion of its pdf depending on how much the aggregate of all previously allocated goodputs deviates from what could be expected for that aggregate. Algorithm SAMPLE-GOODPUTS shows our approach.

The variable *count* keeps track of the number of stations that have been allotted goodputs, and variables *allotted* and *expected* represent the actual and expected goodput allocated to *count* number of stations with allowable tolerance *upper-tolerance* and *lower-tolerance*. All variables are initialized as shown in lines 1 through 3. Note that the goodputs of all stations $N_i(t)$ are assigned zero initially. The algorithm generates a random permutation π of $1..M$ and a sample of $N_A(t)$ from $\mathsf{Pr}(\, N_A(t)\,)$ in lines 4 and 5 respectively. Each iteration of the **while** loop from lines 6 through 22 assigns the goodput of the station i chosen in the position *count* of the random permutation. If the actual *allotted* goodput for the $count-1$ stations is higher (lower) than *expected* subject to an *upper-tolerance* (*lower-tolerance*) as checked in line 8 (line 10) then a tentative sample s is obtained from the lower (upper) tail of distribution of $P(N_i|C_i)$ in line 9 (line 11). Let n^* be a goodput such that $Pr(N_i \leq n^*|C_i) = 1/2$. By sampling the lower (upper) tail of $\mathsf{Pr}(\, N_i|C_i\,)$, we mean sampling from the distribution $\mathsf{Pr}(\, N_i|C_i, N_i \leq n^*\,)$ (distribution $\mathsf{Pr}(\, N_i|C_i, N_i > n^*\,)$). If both tolerances are not exceeded, then $N_i(t)$ is sampled from the full distribution $\mathsf{Pr}(\, N_i|C_i\,)$ in line 13. As long as the tentative sample s taken with the goodput *allotted* so far does not exceed the sampled $N_A(t)$ as checked in 14, $N_i(t)$ is set to s in line 15 or is assigned the residual goodput in line 17 and the assignment stops. In lines 18 through 22 the variables *count*, *allotted*, *expected*, *upper-tolerance*, and *lower-tolerance* are updated. The last station in the random permutation π is assigned the residual goodput, if any, in line 23.

D. Runtime and optimization

Like mentioned before, all pdf's are precomputed or cached after computation during the simulation run. Because this has a one-time fixed cost, we analyze the algorithm assuming that all pdf's are precomputed. The random permutation can be generated in $O(M)$ time by a Knuth shuffle [5]. Each iteration of the **while** loop takes $O(1)$ time to sample a random variable from a distribution (independent of the pdf size by building and indexing a table of the inverse of the cdf) and update state variables. Because there are at most $M-1$ iterations of the loop, the runtime of Algorithm SAMPLE-GOODPUTS takes $O(M)$ deterministic time. Even the most efficient implementation of a packet level simulator would take $O(M\delta \times \text{bit-rate})$ because each packet-transmission by any station schedules events in the other $M-1$ stations. This is the reason why TSS scales much better with increasing bitrates.

VII. OBTAINING THE NEW STATE $C_i(t+\delta)$

We want to obtain the distribution $\mathsf{Pr}(\, C_i(t+\delta)\,)$ given the old state $C_i(t)$ and the goodput $N_i(t)$ that was obtained after accounting for correlation. We analyze the per-station attempt process in the backoff timeline and obtain the distribution of the time instant of the last successful packet transmission in the interval. Given the instant of the last successful transmission, the distribution of the new state can be obtained by Bayes theorem.

We first analyze the case $N_i(t) \neq 0$. Figure 9 shows the backoff timeline interval $[t', t'+\delta\eta]$. In this backoff timeline, X_f^* is the backoff time to the first success from the beginning of the interval. Likewise, X_l^* is the backoff time from the last success to the end of the interval. Because $N_i(t) \neq 0$, X_f^* and X_l^* are well defined. Recall that we have already seen how to obtain the distribution of the backoff time to the first success X_f^* given $C_i(t)$ in Section IV. Our goal is to obtain the distribution $Pr(X_l^*|C_i(t), N_i(t))$. Once this is done, we can obtain the distribution of $C_i(t+\delta)$ given that X_l^* slots have been spent in backing off since the last successful transmission. We can rewrite $Pr(N_i(t)=n|C_i(t))$ as follows:

$$Pr(N_i(t)=n|C_i(t)) = \sum_{r=0}^{r=\eta\delta} Pr(X_f^*=r|C_i(t)) \sum_{s=0}^{s=\eta\delta-r} Pr(X_1+\cdots+X_{n-1}=s)Pr(X_n > \eta\delta\text{–}r\text{–}s)$$

By Bayes' theorem we have:

$$Pr(X_l^*=s|N_i(t)=n, C_i(t)) = \frac{\sum_{r=0}^{r=\eta\delta-s} Pr(X_f^*=r|C_i(t))Pr(X_1+\cdots+X_{n-1}=\eta\delta\text{–}r\text{–}s)Pr(X_n>s)}{Pr(N_i(t)=n|C_i(t))}$$

Suppose $X_l^*=x$. This means the total backoff X_n of the $n+1$-th successful transmission is greater than x. Recall the notation that $Y_1, \cdots, Y_K$ are the the backoff counter values chosen in successive transmission attempts of a tagged packet, if there are successive transmission attempts at all. For $C_i(t+\delta) = 2^{c-1}\gamma$ to occur after spending a backoff duration x from a reset state $< C_i(t)=0, B_i(t)=0 >$, we want $c-1$

SAMPLE-GOODPUTS($\mathbf{N}, \mathbf{M}$)

▷ $\mathsf{Pr}(\ N_i|C_i\ ), \mathsf{Pr}(\ N_A\ )$ are global to this routine

1 *count* $\leftarrow 1$
2 *allotted*, *expected*, *upper-tolerance*, *lower-tolerance* $\leftarrow 0$
3 $\forall i \quad N_i(t) \leftarrow 0$, $M \leftarrow$ number of active stations
4 $\pi \leftarrow$ random permutation of indices of $\mathbf{M}$ with 1's
5 $N_A(t) \leftarrow$ sample from $\mathsf{Pr}(\ N_A(t)\ )$
6 **while** *count* $< M$ and *allotted* $< N_A(t)$
7 $\quad i \leftarrow \pi(count)$
8 $\quad$ **if** *allotted* $>$ *expected* + *upper-tolerance*
9 $\quad\quad s \leftarrow$ sample lower tail of $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$
10 $\quad$ **elseif** *allotted* $<$ *expected* - *lower-tolerance*
11 $\quad\quad s \leftarrow$ sample upper tail of $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$
12 $\quad$ **else**
13 $\quad\quad s \leftarrow$ sample from full distribution $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$
14 $\quad$ **if** *allotted* $+s \leq N_A(t)$
15 $\quad\quad N_i(t) \leftarrow s$
16 $\quad$ **else**
17 $\quad\quad N_i(t) \leftarrow N_A(t) -$ *allotted*
18 $\quad$ *allotted* $\leftarrow$ *allotted* $+N_i(t)$
19 $\quad$ *count* $\leftarrow$ *count* $+1$
20 $\quad$ *expected* $\leftarrow$ *count* $\times \frac{N_A(t)}{M}$
21 $\quad$ *upper-tolerance* $\leftarrow \theta_1 \times$ *expected*
22 $\quad$ *lower-tolerance* $\leftarrow \theta_2 \times$ *expected*
23 **if** (*count* $= M$)
24 $\quad\quad N_{\pi(M)}(t) \leftarrow max(N_A(t) -$ *allotted*, 0)

unsuccessful transmissions, $Y_1 + \cdots + Y_c$ to just exceed x, and $Y_1 + \cdots + Y_{c-1}$ should be less than x. Therefore, we have

$$\begin{aligned} & Pr(C_i(t+\delta) = 2^{c-1}\gamma | X_l^* = x) \\ = \; & p^{c-1} \frac{Pr(Y_1 + \cdots + Y_{c-1} \leq x \wedge Y_1 + \cdots + Y_c > x)}{Pr(X_n > x)} \\ = \; & p^{c-1} \frac{Pr(Y_1 + \cdots + Y_{c-1} \leq x) - Pr(Y_1 + \cdots + Y_c \leq x)}{Pr(X_n > x)} \end{aligned}$$

Unconditioning on X_l^* yields $\mathsf{Pr}(\ C_i(t+\delta)|C_i(t), N_i(t)\ )$.

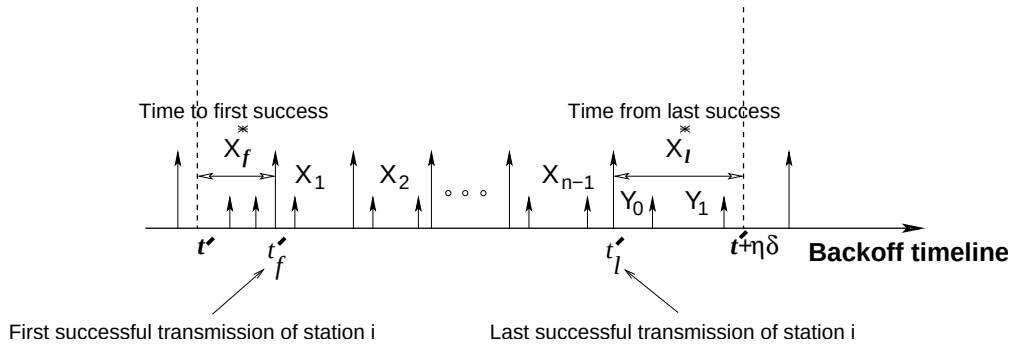


Fig. 9. Transmissions of a tagged station in the backoff timeline interval $[t', t' + \eta\delta]$ corresponding to the real timeline interval $[t, t + \delta]$. A longer arrows indicates a successful transmission, a shorter arrow failure.

When $N_i(t) = 0$, as in Section IV, we approximate $B_i(t)$ as the forward recurrence time. Using the forward recurrence time and a similar application of Y_i's, we obtain the distribution of $C_i(t + \delta)$ conditioned on X_f^*. Again, unconditioning on X_f^* yields the required distribution.

VIII. The Timestepped Simulator

We describe how the analysis fits together as the TSS generates a sample path. The simulator initializes the state of all stations at $t = 0$ in line 2. The **while** loop in line 4 iterates through *sim-duration* in timesteps of δ. The number of active stations M is obtained in line 5 from either the simulation input or from the outputs of higher layer protocols (e.g., TCP) making the WLAN output queues non-empty. The corresponding collision probability is computed/looked up from M in line 6. By looked up, we mean looked up from a cache that was populated either before the simulation began or during the simulation run itself. The precomputation phase is possible because all required probability distributions $(\mathsf{Pr}(\, N_A(t)\,), \mathsf{Pr}(\, N_i(t)|C_i(t)\,), \mathsf{Pr}(\, C_i(t+\delta)|N_i(t), C_i(t)\,)$ are parametrized easily by δ and M (and other fixed protocol parameters like the initial contention window γ and the maximum number of attempts β).

Within each timestep, the following steps occur:

- $N_A(t)$ is sampled from its distribution in line 8.
- For each station i, $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ is obtained in 9.
- Algorithm Sample-goodputs is used to sample the goodput of each station i in line 11.
- For each i, the distribution of $\langle C_i(t+\delta), B_i(t+\delta)\rangle$ is obtained given $C_i(t)$,$B_i(t)$, and $N_i(t)$ and sampled to obtain the new state. This is done in lines 12 through 13

As seen before, Algorithm Sample-goodputs takes $O(M)$ time. All further random sampling can be done in $O(M)$ time. Each iteration of the **while** loop in line 4 takes $O(M)$ time assuming that all pdf's are precomputed. Because the precomputation of the pdf's can be amortized over various runs of the simulation, we do not consider the runtime for it. Thus the runtime of Algorithm TSS-WLAN is $O(M.\, \textit{sin-duration})$, which is **independent of the bit-rate.**

TSS-WLAN(α, γ, δ)

▷ α : total stations

▷ γ : initial contention window

▷ δ : simulation timestep

1 $t \leftarrow 0$

2 **for** $i = 1$ **to** α

3 $\quad C_i(t) \leftarrow \gamma$

4 **while** ($t <$ *sim-duration*)

▷ All this is for interval $[t, t + \delta]$.

▷ We omit t everywhere for brevity except in line 13.

5 $\quad M \leftarrow$ number of active stations

6 $\quad$ compute/look up collision probability p for M stations

7 $\quad$ compute/look up $\mathsf{Pr}(\ N_A\ )$ using M, δ, p

8 $\quad N_A \leftarrow$ sample from $\mathsf{Pr}(\ N_A\ )$

9 $\quad$ **for** each station i

10 $\quad\quad$ compute/look up $\mathsf{Pr}(\ N_i|C_i\ )$ using M, δ, p

11 $\quad$ SAMPLE-GOODPUTS($\mathbf{N}, \mathbf{M}$)

12 $\quad$ **for** $i = 1$ **to** M

13 $\quad\quad$ compute/look up distribution of $\mathsf{Pr}(\ C_i(t + \delta)|N_i(t), C_i(t)\ )$ and sample

14 $\quad t \leftarrow t + \delta$

IX. Results

Our main results are broadly along two directions: quantifying speedup and validating accuracy. Because the pdf's required for TSS are precomputed using the transient analysis, we first quantify the cost for precomputation in (memory) space and time. Then we compare the runtime improvement offered by TSS over PLS. Next, we validate 1) the transient analysis of 802.11; and 2) the overall TSS technique for WLANs.

For validation of the transient analysis of 802.11, we compare the conditional pdf's, namely, $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$ and $\mathsf{Pr}(\ C_i(t + \delta)|N_i(t), C_i(t)\ )$. For validation of TSS, we compare an "internal" (to the method) metric, namely, $C_i(t)$ and an "external" metric, namely, $N_i(t)$. Specifically, we consider:

1) the pdf of $C_i(t)$;
2) the autocorrelation function of the timeseries $C_i(0), C_i(\delta), \cdots$ that captures correlations across time;
3) the crosscorrelation function between the series $C_i(0), C_i(\delta), \cdots$ and $C_j(0), C_j(\delta), \cdots$ that captures correlations across stations.

The same three points of comparison are considered for the metric $N_i(t)$ as well. Note that the average delay in a timestep can be obtained as the inverse of $N_i(t)$. In addition, we also consider the pdf of the aggregate goodput $N_A(t)$. Finally, we consider ensemble metrics for simulation scenarios with time-varying M.

Our secondary results include 1) validating the accuracy of our algorithm to obtain the convolution of the total backoff duration pdf and quantifying the speedup obtained by it; and 2) a closed form approximation and analysis of the per-station collision probability as a function of M.

A. Simulation setup

Because TSS models only the MAC layer, to insure a fair comparison of the time taken for a simulation, we have implemented a simple 802.11 MAC layer packet level simulator (PLS) instead of resorting to a full blown simulator such as ns-2 [1]. This avoids the overheads of upper layer (routing, transport) as well as lower layer (physical) events in ns-2 which TSS for 802.11 does not model. As an illustration of ns-2 overheads, a simulation run of 1000 seconds for a scenario of two constant bit rate (CBR) flows sharing one 802.11 channel takes about 4.5 seconds in our custom simulator with logging enabled, while ns-2 takes about 70 seconds with all logging disabled.

All simulations were carried on a machine with a 3.2GHz Pentium-4 processor and 1.5Gb RAM running Red Hat Enterprise Linux release 3. We use a fixed packet size of 1500 bytes including the MAC-layer overhead and the 802.11a parameters: slot size of $9\mu s$, SIFS of $16\mu s$, data bitrate of 54Mbps, ACK bitrate 6Mbps, PHY-layer overhead of $20\mu s$, and contention window ranging over the 7 values $[16, 32, \cdots, 1024]$ with 7 maximum attempts. Unless otherwise mentioned, all stations always have packets to transmit in their output queues, i.e., $M(t)$ is constant, during the entire run of the simulation.

B. Precomputation costs in space and time

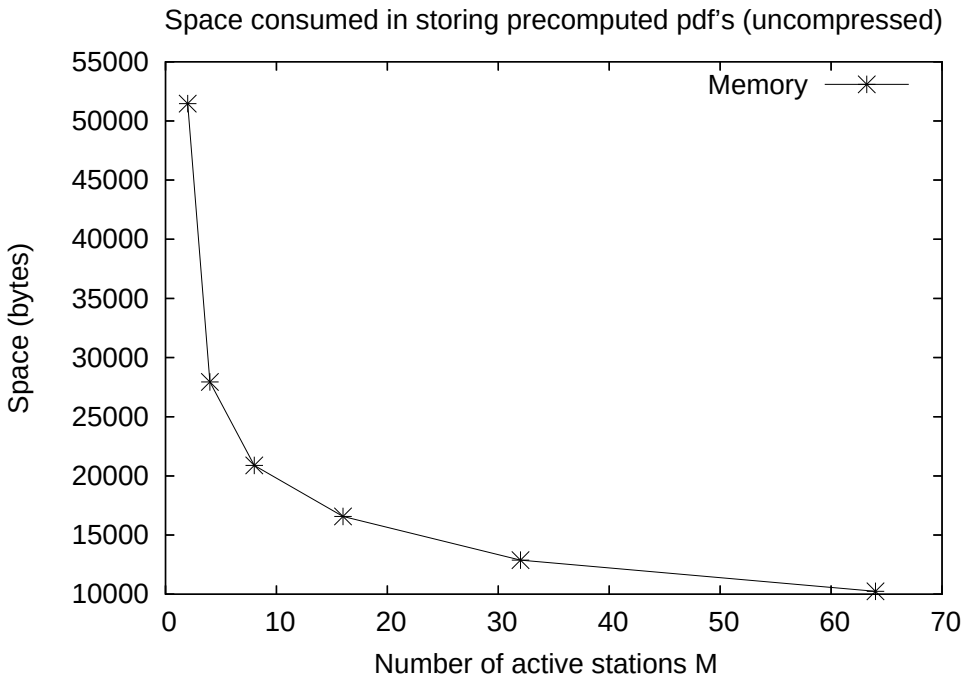
Using the transient analysis, for each tuple $\langle M, C_i(t)\rangle$, a table of tuples of the form $\langle N_i(t), \mathit{pdfval}\rangle$ is obtained for $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$. Likewise, for each tuple $\langle M, C_i(t), N_i(t)\rangle$, a table of tuples of the form $\langle C_i(t+\delta), \mathit{pdfval}\rangle$ is obtained for $\mathsf{Pr}(\, C_i(t+\delta)|N_i(t), C_i(t))\,)$. Because $C_i(t)$ ranges over the standard seven values $[16, \cdots, 1024]$, for a fixed M, the sizes of all tables are determined by the maximum value $N_i(t)$ can take.

For a fixed M, let n_m denote the maximum value of $N_i(t)$ for which entries of tables are computed. So the tables for $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ have $7n_m$ entries in all. Likewise, the tables for $\mathsf{Pr}(\, C_i(t+\delta)|C_i(t), N_i(t)\,)$ have $7 \times n_m \times 7 = 49n_m$ entries in all. Each entry in the table is stored as a `double` of size eight bytes. So the space required is $400n_m$ bytes. For $M = 2$, n_m is about 130, and this yields a space requirement of about 52000 bytes (in uncompressed form).

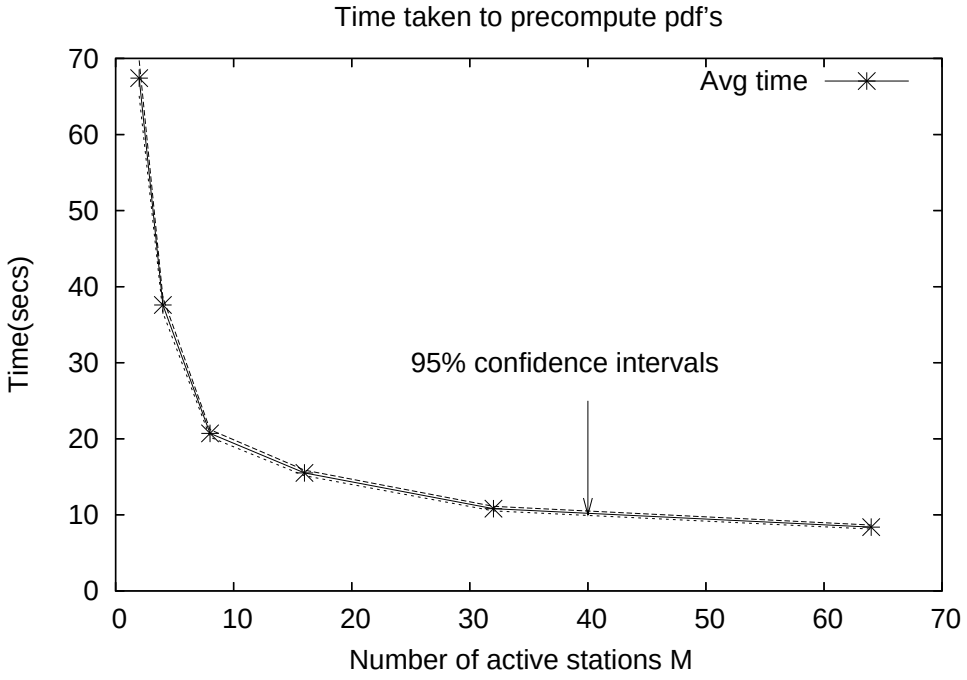
Figure 10(a) shows the space requirement for pdf's for $\delta = 50$ms with M varying in $[2, 4, 8, \cdots, 64]$. Because n_m decreases with increasing M, the space required decreases with increasing M. A similar trend can be seen in Figure 10(b), which shows the time taken to precompute the pdf's and store it to disk.

The space requirement is almost negligible compared to memory consumed in typical packet level simulators, and the time requirement is a one-time cost shared across all runs of a simulation scenario. Nevertheless, these costs can be reduced by interpolating the pdf's among the parameters M and N_i (C_i is likely not a suitable candidate for interpolation for large M).

We note that n_m increases with increasing δ, so the table sizes increase with increasing δ. Even though the memory required is low, the nature of TSS allows us to trade off space with time as follows: instead of precomputing tables for large δ, precompute for, say, $\delta/2$ and perform computation for two smaller sub-timesteps of $\delta/2$ before updating metrics for the required timestep of δ. This sort of trade-off is very difficult,



(a)



(b)

Fig. 10. The space and time costs of precomputation of $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$ and $\mathsf{Pr}(\ C_i(t+\delta)|C_i(t), N_i(t)\ )$ for $\delta = 50$ms with M varying in $[2, 4, 8, \cdots, 64]$.

if not impossible, to achieve in PLS.

C. Runtime comparison

TSS for WLANs provides an improvement up to two orders of magnitude in the runtime over PLS. Figure 11(a) shows the average time taken by both PLS and TSS for a 1000s simulation run with M in $[2, 4, 8, \cdots, 64]$. Figure 11(b) shows the ratio of the runtimes for PLS and TSS. Each point plotted in Figure 11(a) and its associated 95% confidence interval is obtained from 100 runs. For PLS, the curve is shown scaled down by a factor of 50 to enable visual comparison with TSS. For TSS, the runtime includes the time taken to load precomputed pdf's from disk, and the time taken for precomputation is amortized over 100 runs. The PLS curve shows a linear increase in the runtime as expected. The TSS curve shows a dip and then an increase. This is because the amortized time to calculate precomputed pdf's is significant compared to the actual simulation loading time and runtime for smaller M; once a threshold has been crossed in M, the computational costs predominate. The trend in the TSS runtime curve for smaller M is similar to the precomputation cost curves in Figures 10(b) and 10(a).

D. The instantaneous aggregate goodput distribution $\mathsf{Pr}(\, N_A(t)\,)$

We obtain the distribution of the instantaneous aggregate goodput for $\delta = 50$ms through simulations and analysis. In each run of the simulation, the system is "warmed up" for 5s from a "cold start" and then a sample of the instantaneous aggregate goodput is obtained. We obtain the pdf of the instantaneous aggregate goodput from the samples of 10000 such runs and the results comparing it with analytically predicted distribution are shown in Figure 11.

We make the following observations:

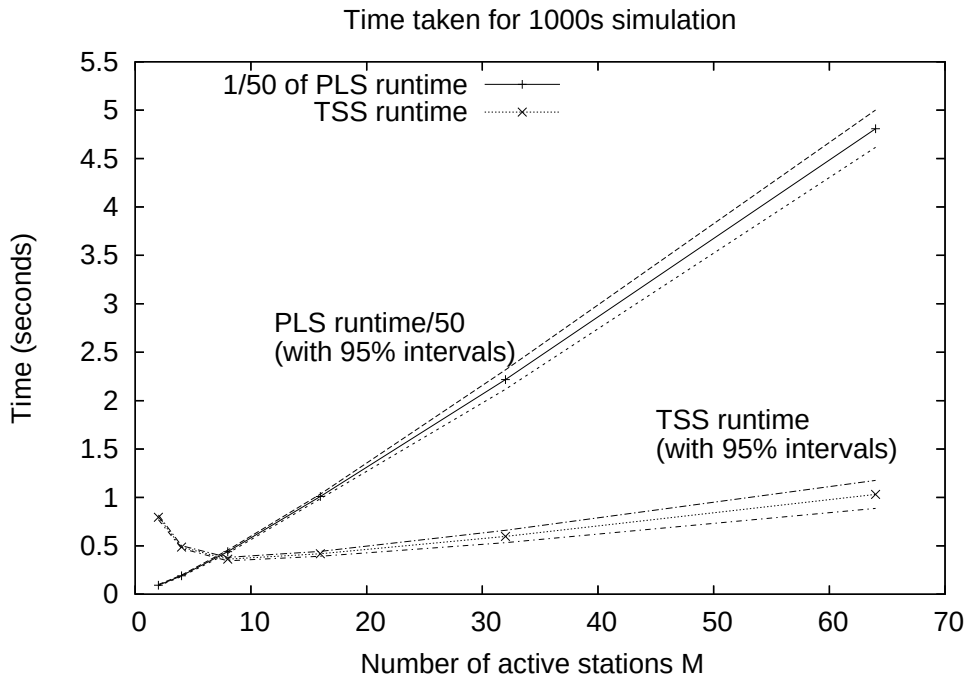
- The distribution of $N_A(t)$ can be well approximated by a gaussian as predicted by the analysis.
- The means of the distributions obtained by simulation coincide almost exactly with those obtained by analysis.
- The peaks (deviations) of the normal distributions obtained by simulations are higher (lower) than those obtained by analysis; for $M = 2$ the scenario is reversed.

The last observation can be explained as follows. For the analysis, we had assumed that each throughput renewal in the global timeline is a failure with a fixed probability independent of the past. In reality, there are two factors that affect the variance, namely:

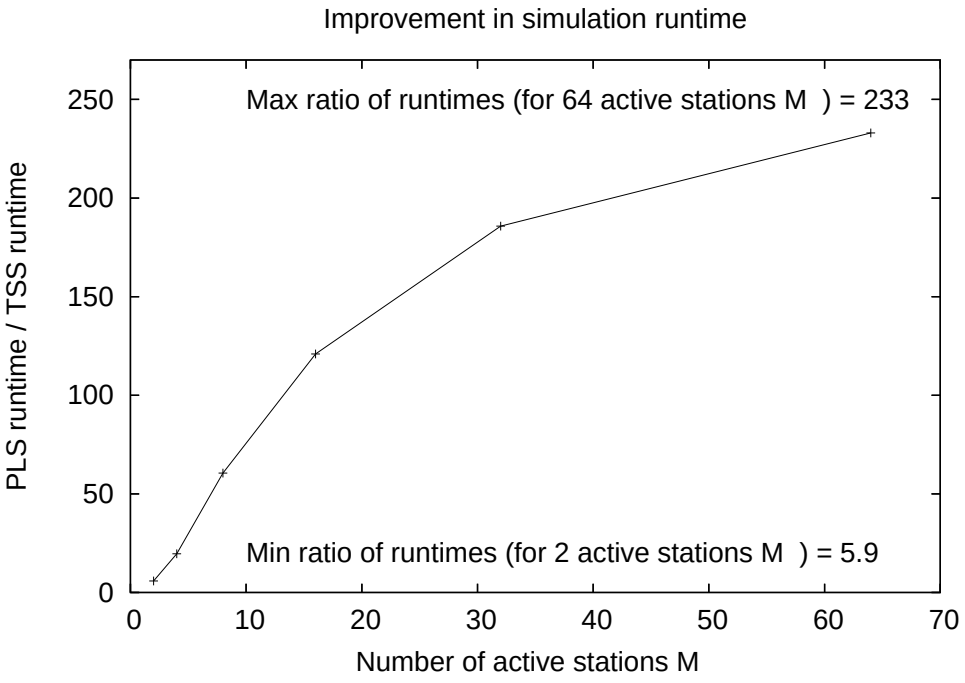
1 The size of an idle interval is positively correlated with the event that the preceding throughput renewal(s) is a collision.

2 The event that a throughput renewal is a failure is negatively correlated with the event that previous throughput renewal(s) is a failure.

Because a collision in a throughput renewal increases the contention windows of at least two stations, it increases the range of values over which a minimum is chosen for the next attempt thereby causing factor



(a) Runtime of TSS and PLS for a 1000s simulation with M varying in $[2, 4, 8, \cdots, 64]$. Each point is an average of 100 runs. For PLS, the runtime has been scaled down by a factor of 50 to enable visual comparison with TSS. For TSS, the runtime includes the time taken to load precomputed pdf's from disk, and the time taken for precomputation is amortized over 100 runs.



(b) Ratio of PLS runtime to TSS runtime

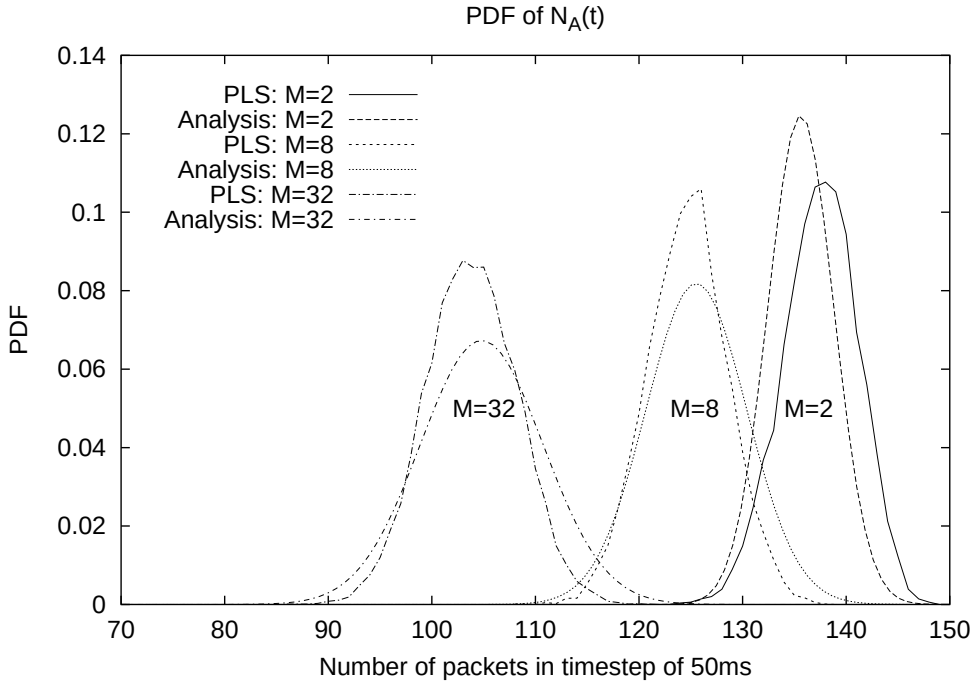


Fig. 11. Comparison between empirically obtained pdf of $N_A(t)$ for $t = 5$s and $\delta = 50$ms for varying M. The deviations of $N_A(t)$ predicted by the analysis are overestimates for $M > 2$ while for $M = 2$, it is an underestimate.

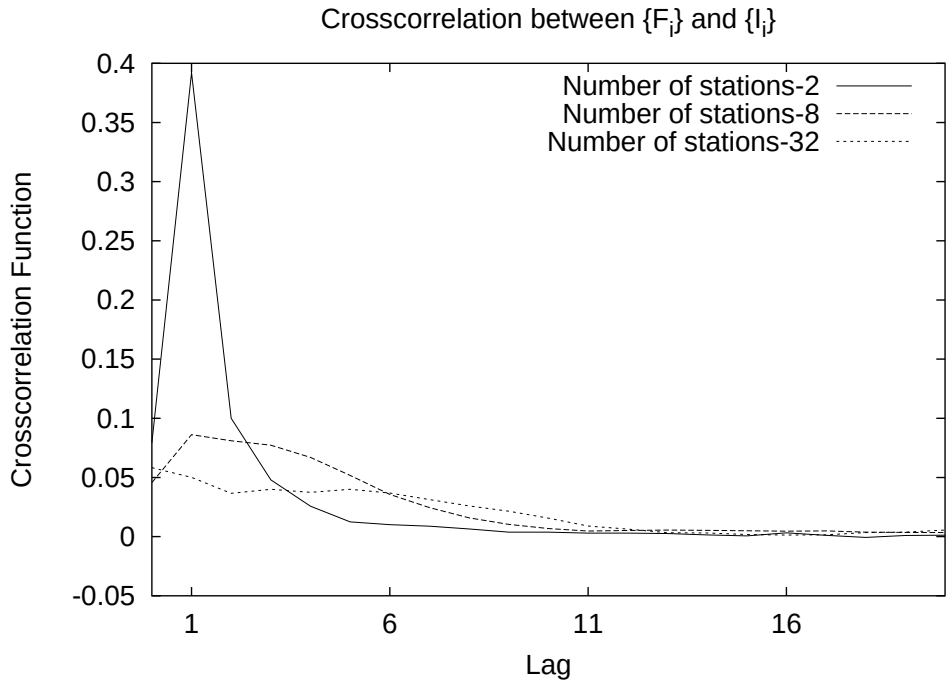


Fig. 12. Crosscorrelation function between sequences $\{I_i\}$ and $\{F_i\}$ obtained over 1000000 samples for each M

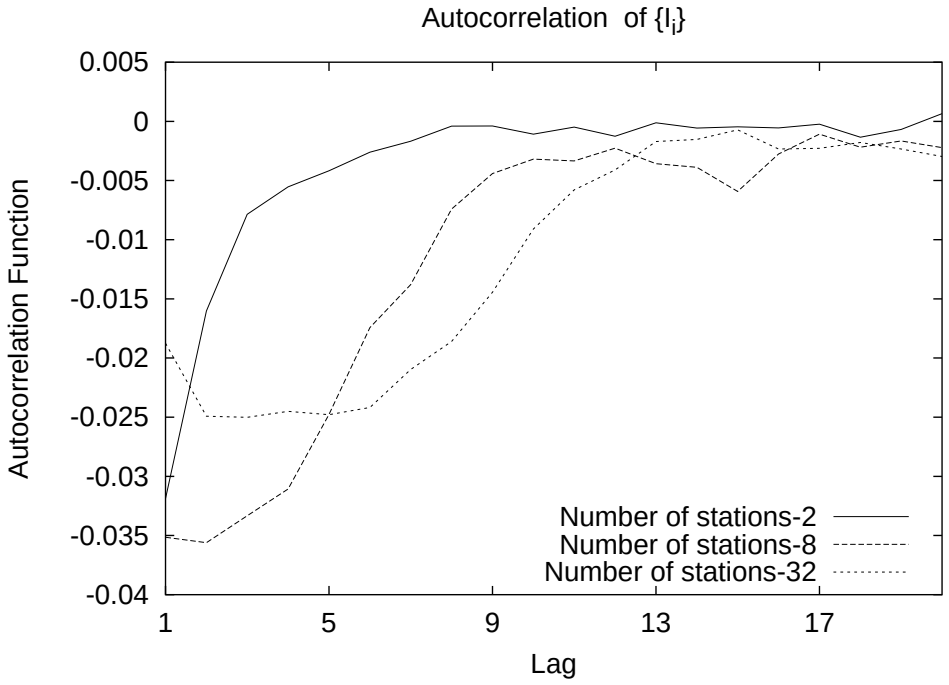


Fig. 13. Autocorrelation function of I_i sequence obtained over obtained over 1000000 samples for each M. At lag 0, the function is exactly 1 and not shown in the figure.

1. For exactly the same reason, a collision reduces the probability of future collisions, thereby causing factor 2. Factor 1 increases the variance of the goodput renewal period over that of completely independent idle intervals and transmission successes, whereas factor 2 decreases the variance of the goodput renewal period. For $M > 2$, factor 2 dominates over factor 1, thereby explaining why simulations yield lower deviation. For $M = 2$, factor 1 dominates because after any collision, there are no other stations whose backoff counters could be in a lower range.

We illustrate these correlation factors through simulations. On some sample path, let $I_1, I_2, \ldots$ denote the idle intervals in some sample path of the system, and let $F_1, F_2, \ldots$ be indicator random variables such that F_i is 1 iff the the transmission preceding I_i is a failure. Figure 12 shows the cross-correlation function between the sequences $\{I_i\}$ and $\{F_i\}$. The peak at lag 1 illustrates factor 1. Figure 13 shows the autocorrelation function of the sequence $\{F_i\}$ obtained over 1000000 samples for varying M. As can be seen, there is negative correlation over a significant lag, illustrating factor 2.

E. The distributions $\Pr(\, N_i(t)|C_i(t)\,)$ *and* $\Pr(\, C_i(t+\delta)|N_i(t), C_i(t)\,)$

For each value of M, we do 100000 simulation runs with M constant throughout the simulation runs. In each simulation run, at $t = 5$s and $\delta = 50$ms, a sample of $N_i(t), C_i(t)$, and $C_i(t+\delta)$ is obtained. From 100000 samples from 100000 such runs, a frequency distribution of N_i is obtained for each fixed C_i as an

estimate of the conditional probability distribution $\mathsf{Pr}(\ N_i|C_i\ )$. From this same set of samples, a conditional distribution of $C_i(t+\delta)$ given $C_i(t), N_i(t)$ is also obtained. This entire exercise is repeated for varying values of M.

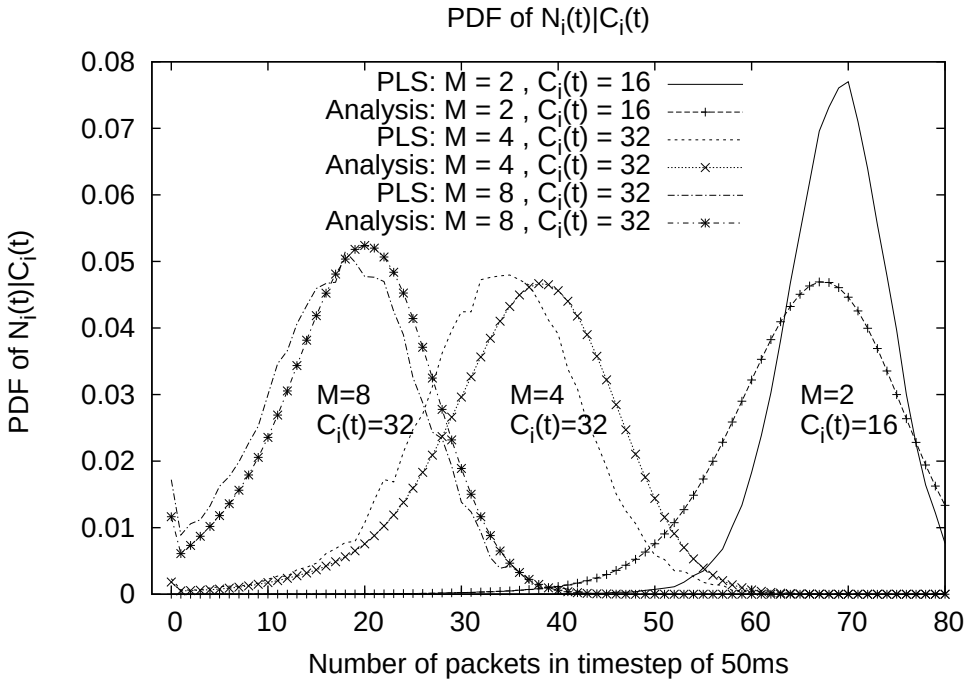
Figure 14(a) shows the PDF $\mathsf{Pr}(\ N_i|C_i\ )$ for smaller contention windows for varying M. The pdf's do not match exactly because of our approximation in obtaining the random total backoff in an interval $[t, t+\delta]$ by a constant $\eta\delta$. However, the accuracy improves with increasing M. For two stations, the distribution is almost normal. While the mean matches, the deviation doesn't quite match; this is due to the small lag correlations as explained before. As M and contention window size increase, the goodput starts deviating from normal significantly with an increased probability of zero instantaneous goodput. The analysis captures this trend as can be seen in Figure 14(b). We also note that $Pr(N_i(t) = 0|C_i(t) = 1024)$ for $M = 16$ is much higher (around 0.45) than $Pr(N_i(t) = 0|C_i(t) = 16)$ for $M = 32$ (around 0.075) illustrating the short-term unfairness; even though the number of active stations is doubled (i.e. $M = 32$), the probability of zero goodput is much lower (than for $M = 16$) because of a favorable contention window (in this case 16).

Figures 15(a) and 15(b) shows the distribution of $C_i(t+\delta)|N_i(t), C_i(t)$ for varying values of $M, N_i(t)$, and $C_i(t)$. Figure 15(a) covers low values of $C_i(t)$ while Figure 15(b) shows the same distribution for relatively higher values. The accuracy is quite good for both $N_i(t) = 0$ as well as $N_i(t) \neq 0$, thereby validating both cases of the analysis in Section VII.

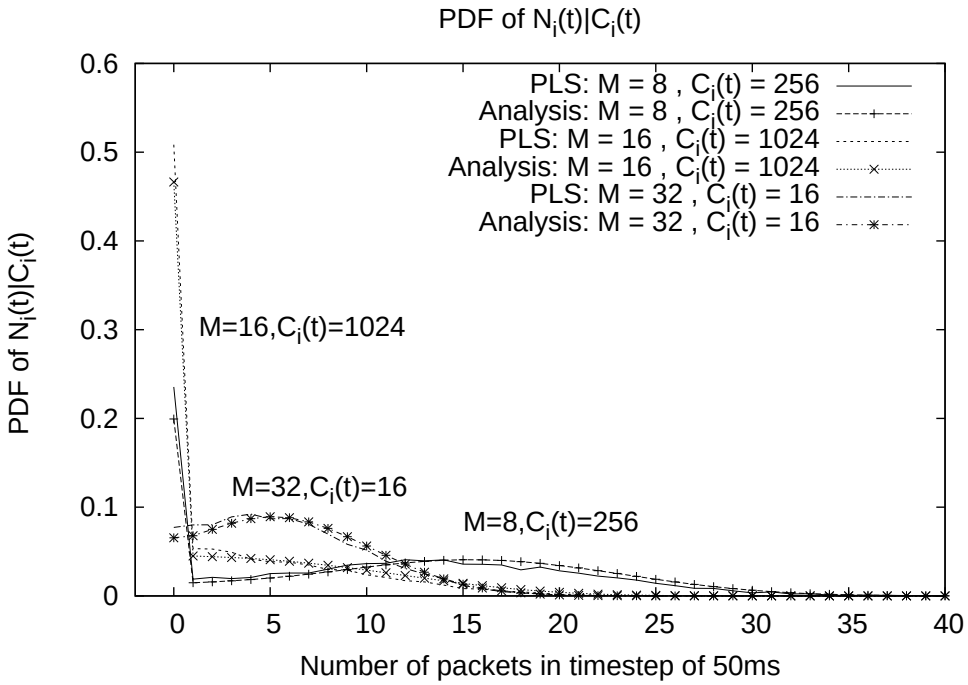
F. The distributions $\mathsf{Pr}(\ N_i(t)\ )$ *and* $\mathsf{Pr}(\ C_i(t)\ )$

We now compare the unconditional distributions of $N_i(t)$ and $C_i(t)$. As can be seen from Figures 16(b) and 16(a), the distribution of $N_i(t)$ as obtained from TSS is very close to that obtained from PLS except for a large M (e.g., 64) where it overestimates the time with zero goodput (and underestimates the others). This is because TSS overestimates the probability of $C_i(t)$ being high for large M; the reason for this is explained in the next paragraph.

Next, we compare the distribution of $C_i(t)$ obtained by both TSS and PLS in Figures 17(a) and 17(b). Note that this distribution so obtained is an approximation of the frequency distribution of the time spent by the tagged station in each possible value of the contention window. When M is low, TSS tracks the trend quite accurately. However, when M is very high (e.g., 64) TSS overestimates the time spent in high backoff states (e.g., $C_i = 1024$), which are more likely with more stations. This is because we track only $C_i(t)$ and approximate $B_i(t)$ by the forward recurrence time. Suppose $C_i(t) = 1024$ and that $N_i(t)$ was probabilistically chosen to be zero in some timestep $[t, t+\delta]$ according to the algorithm. With high probability $C_i(t+\delta) = C_i(t)$, i.e., there were no transmissions and the state is unchanged. Now note that $\mathsf{Pr}(\ N_i(t+\delta)|C_i(t+\delta)\ )$ is the same as $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$, i.e., there is no credit for the backoff duration of the timestep $[t, t+\delta]$. This would have been modeled if $B_i(t)$ was also tracked and used in obtaining $\mathsf{Pr}(\ N_i(t)|C_i(t), B_i(t)\ )$ instead of just being approximated as $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$. Thus TSS overestimates the frequency of $C_i(t)$ being high for high M, and therefore it also overestimates the frequency of a station obtaining zero goodput as observed in the

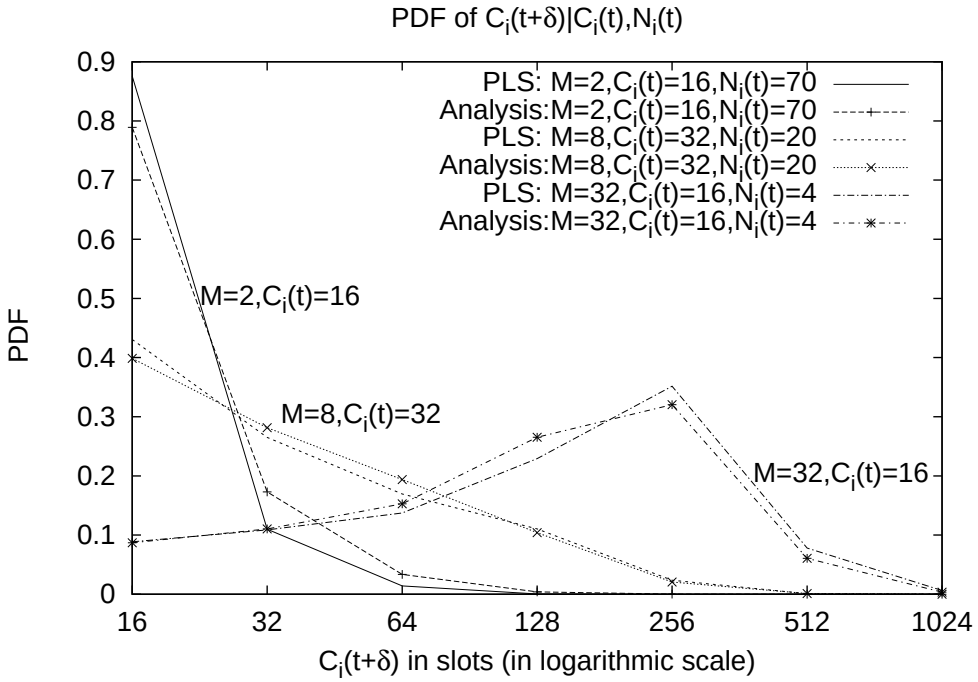


(a)

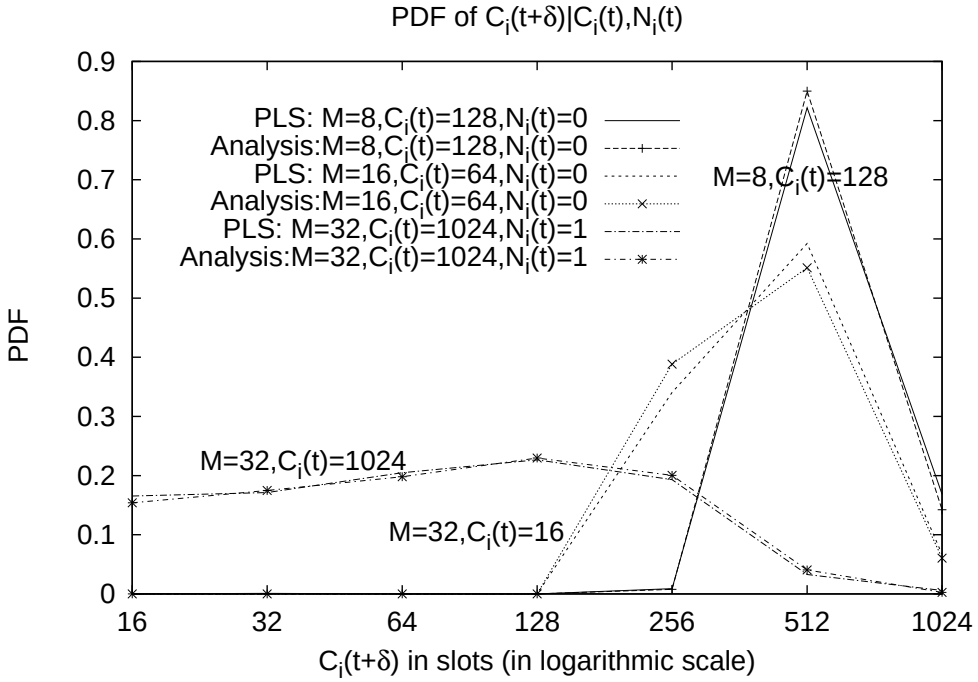


(b)

Fig. 14. PDF of $N_i(t)|C_i(t)$ for various values of $C_i(t)$ and varying M.

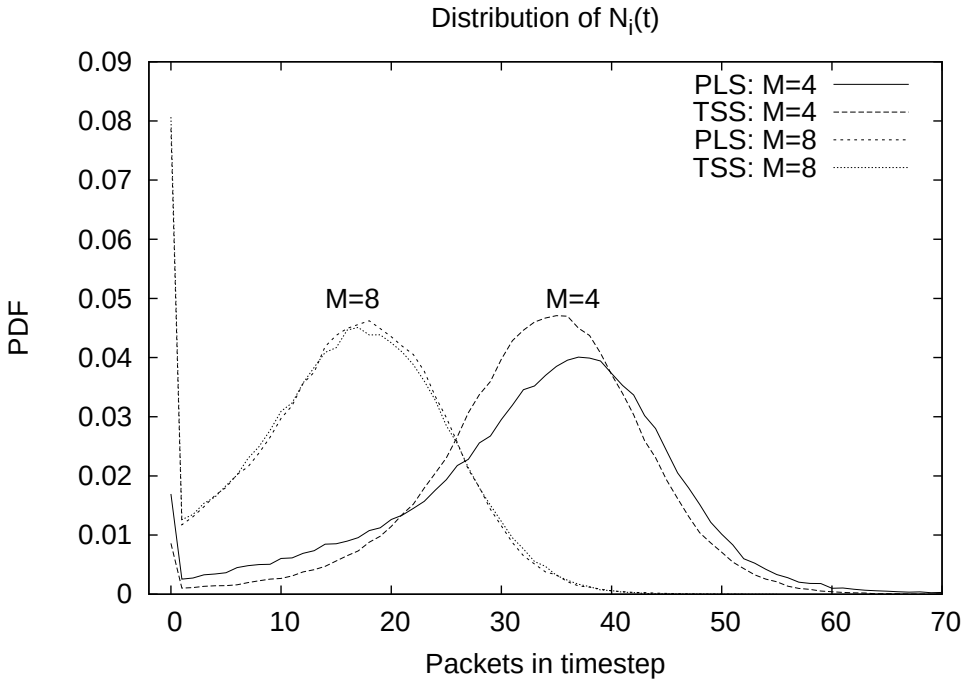


(a)

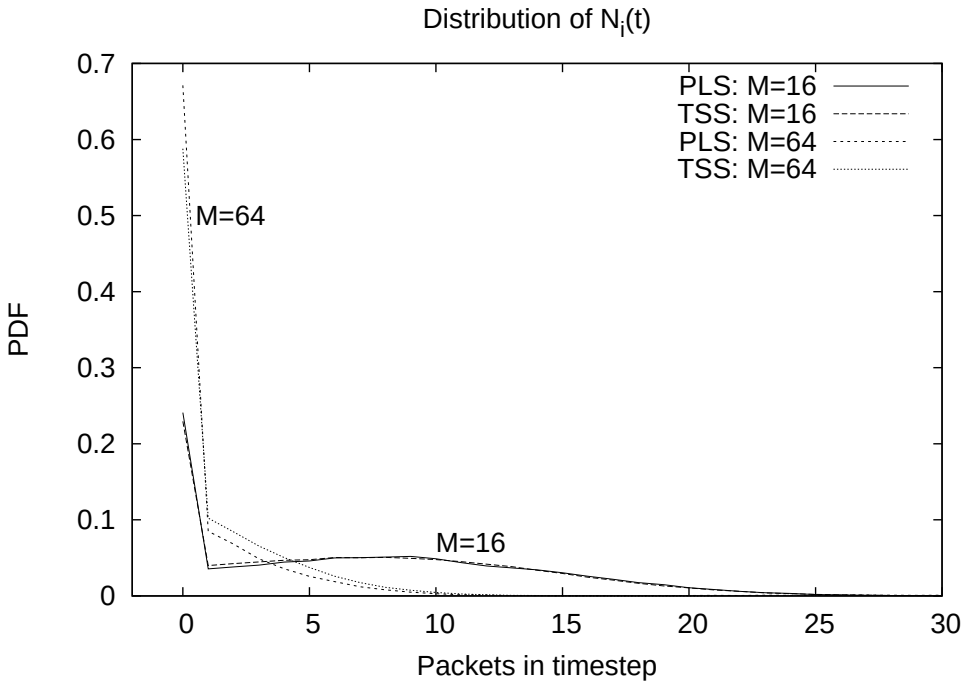


(b)

Fig. 15. PDF of $C_i(t+\delta)|N_i(t),C_i(t)$ for various values of $C_i(t)$,$N_i(t)$,and M.



(a)



(b)

Fig. 16. Distribution of unconditional $N_i(t)$

previous paragraph.

G. Comparing sample paths

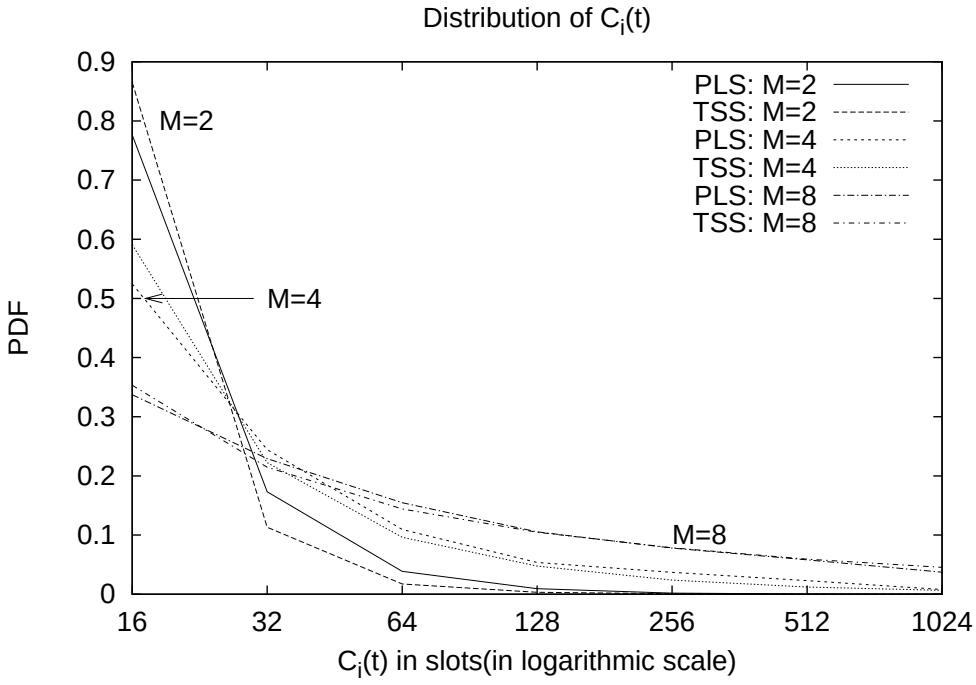
We now compare sample paths generated by TSS and PLS for statistical similarity. To do this, we obtain one single sample path of the system for a run of 10000 seconds both by TSS and PLS for a fixed M. Like before, all comparisons are repeated for varying values of M.

In a fixed sample path, we compare how TSS handles the correlations both across time for a tagged station as well across stations at a given timestep. To do this, we obtain the autocorrelation function for the per-station goodput time-series, i.e., $N_i(0), N_i(\delta), ...$ Figures 18(a) and 18(b) show the autocorrelation function obtained from TSS and PLS. TSS tracks the correlation in N_i across time for a tagged station well for small M. For higher M, while the exact values do not match as well, TSS captures the trend in the autocorrelation function. The reason for this mismatch can be seen in Figures 19(a) and 19(b), which compare the autocorrelation function of the timeseries $C_i(0), C_i(\delta), \cdots$ for a tagged station i. For high M (e.g., 64), TSS does not track the negative correlation between successive samples of C_i in a sample path at higher lags (e.g., 2 through 6). This is because, as mentioned before, the remaining backoff time $B_i(t)$ is being approximated depending on $C_i(t)$.

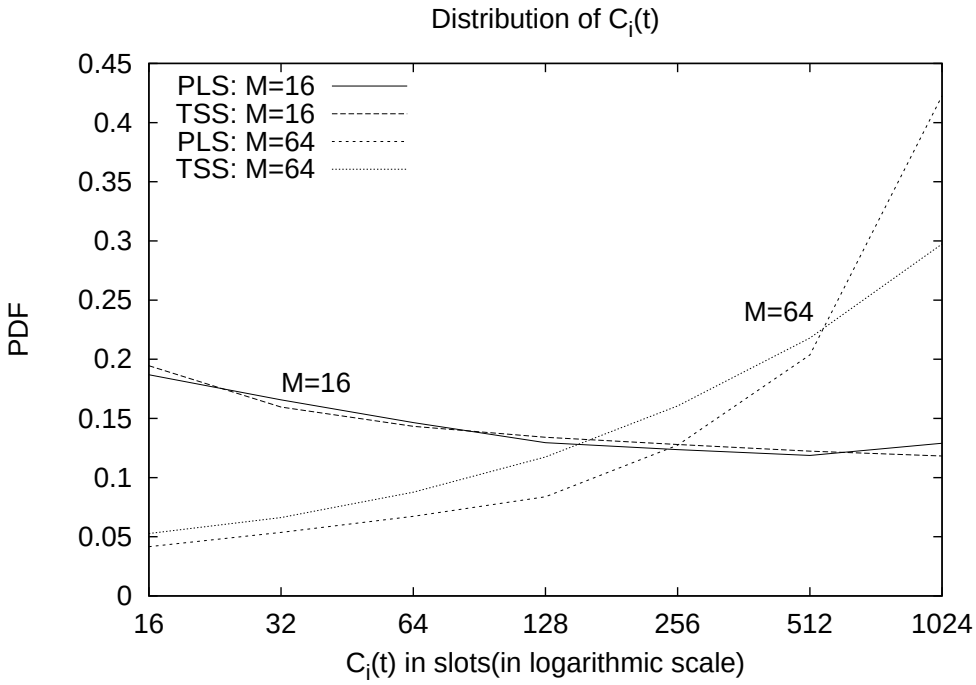
Next, we consider the crosscorrelation function computed between the goodput samples of two tagged stations i and j, i.e., between the timeseries $N_i(0), N_i(\delta), N_i(2\delta), \cdots$ and $N_j(0), N_j(\delta), N_j(2\delta), \cdots$. As can be seen in Figures 20(a) and 20(b), the method to ensure negative correlation between goodputs works well for varying M including larger values. Finally, we consider the crosscorrelation between the timeseries $C_i(0), C_i(\delta), \cdots$ and $C_j(0), C_j(\delta), \cdots$ for the contention windows of two tagged stations i and j in Figures 21(b) and 21(a). The curves match except for the case $M = 2$ when the C_i and C_j are positively correlated (because the two stations can collide only with each other) which TSS doesn't track.

H. Time-varying M and ensemble metrics

We now consider an example where M varies over time. We allow the number of active stations to vary during a simulation run and compare the ensemble metrics predicted by TSS and PLS. Each run of this simulation lasts for 100s and the M is initially 32. M is halved at 25s,50s,and 100s eventually leading to two active stations. The time evolution of the ensemble mean and deviation of $N_i(t)$ of a tagged station i is obtained as an average over 1000 such runs. The evolution of $E[N_i(t)]$ is shown in Figure 22(a) and the evolution of $Dev[N_i(t)]$ is shown in Figure 22(b). TSS captures the ensemble mean accurately for all M, while it does not capture the ensemble deviation accurately for $M = 2$ (the time interval from 75s through 100s). A zoomed-in version of curves around the transition point at 50s is shown; TSS shows the same trend as PLS in the very next timestep.

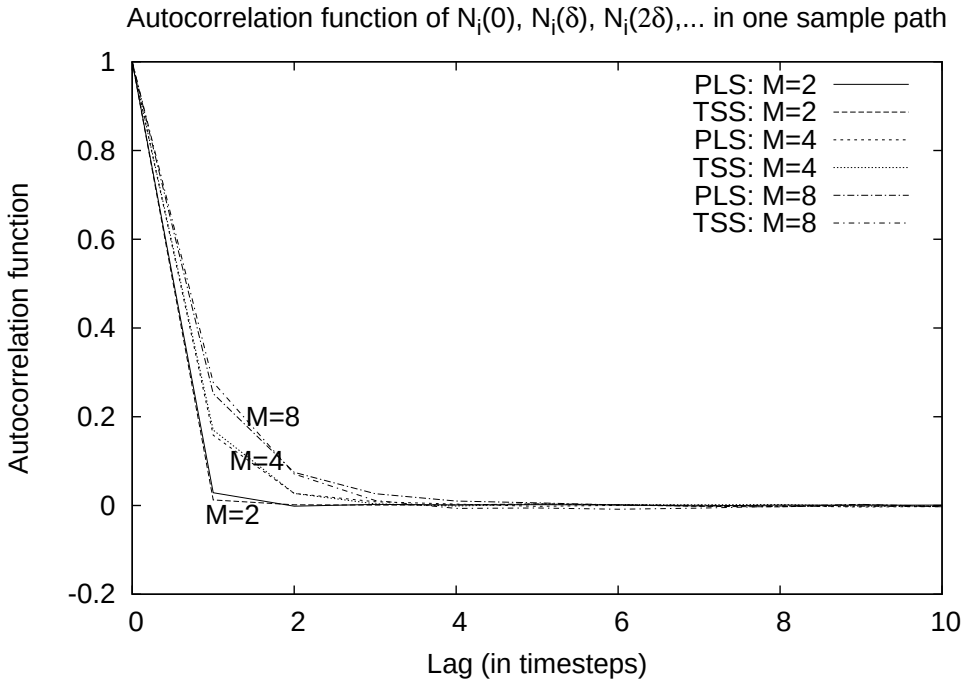


(a)

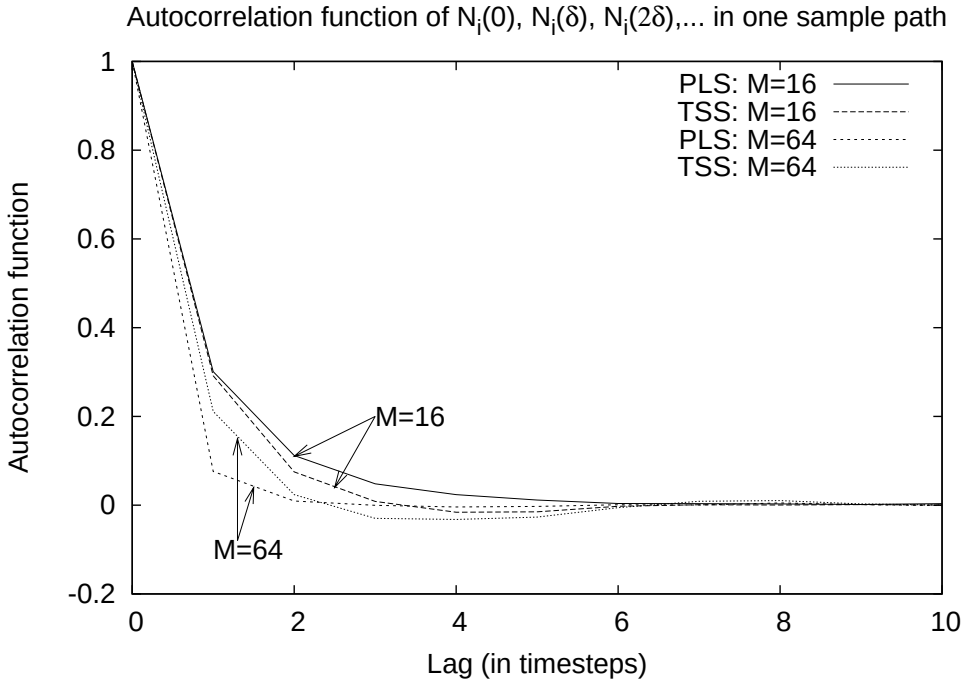


(b)

Fig. 17. Distribution of $C_i(t)$ for varying values of M.

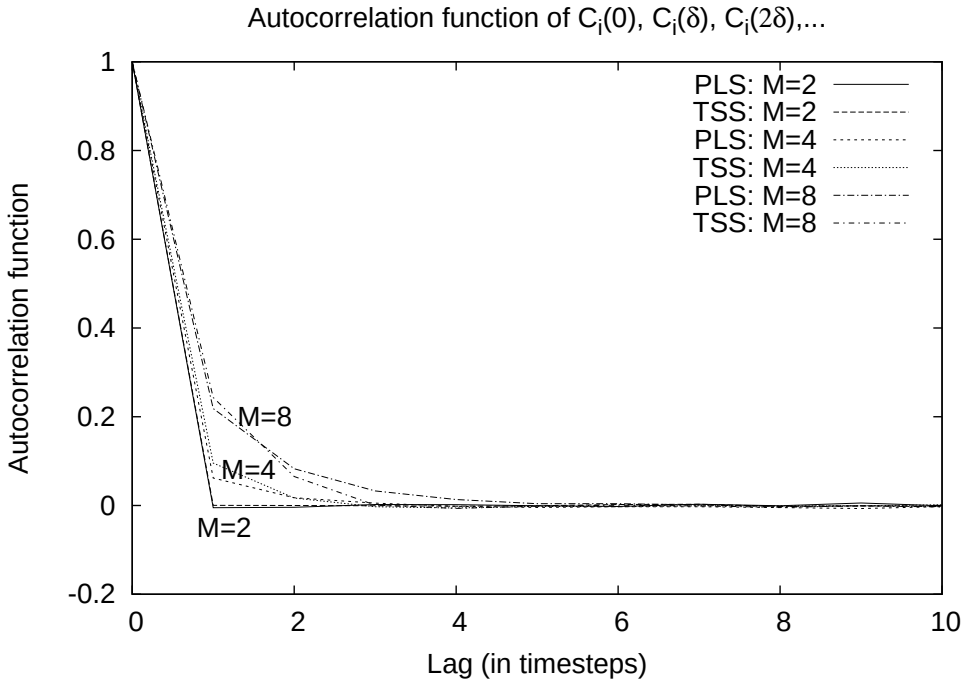


(a)

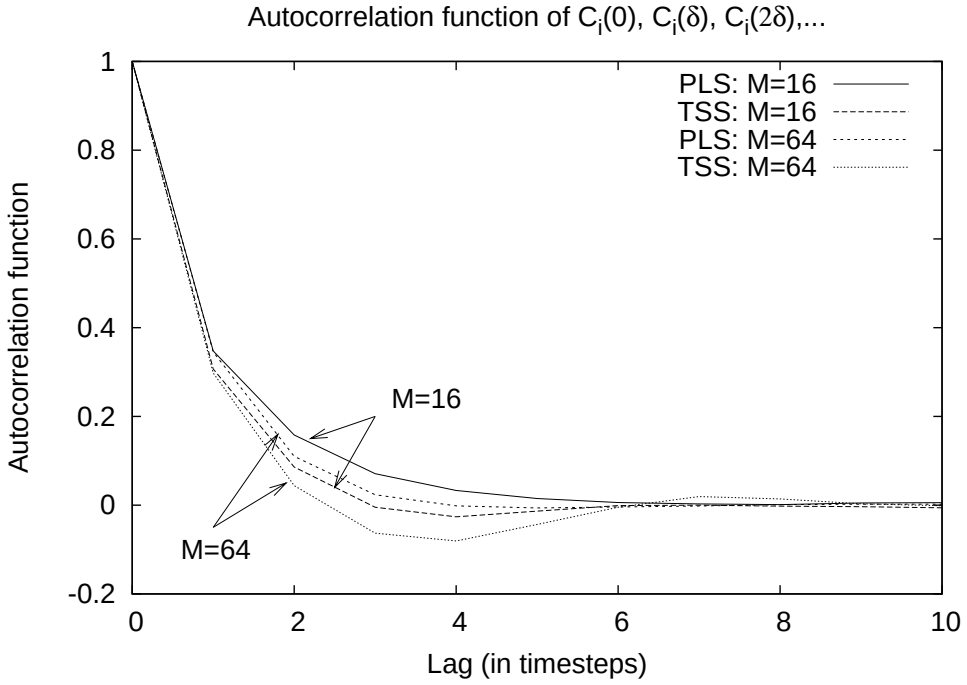


(b)

Fig. 18. Autocorrelation function obtained from samples $N_i(0), N_i(\delta), N_i(2\delta), \cdots$ of one sample path for various values of M.

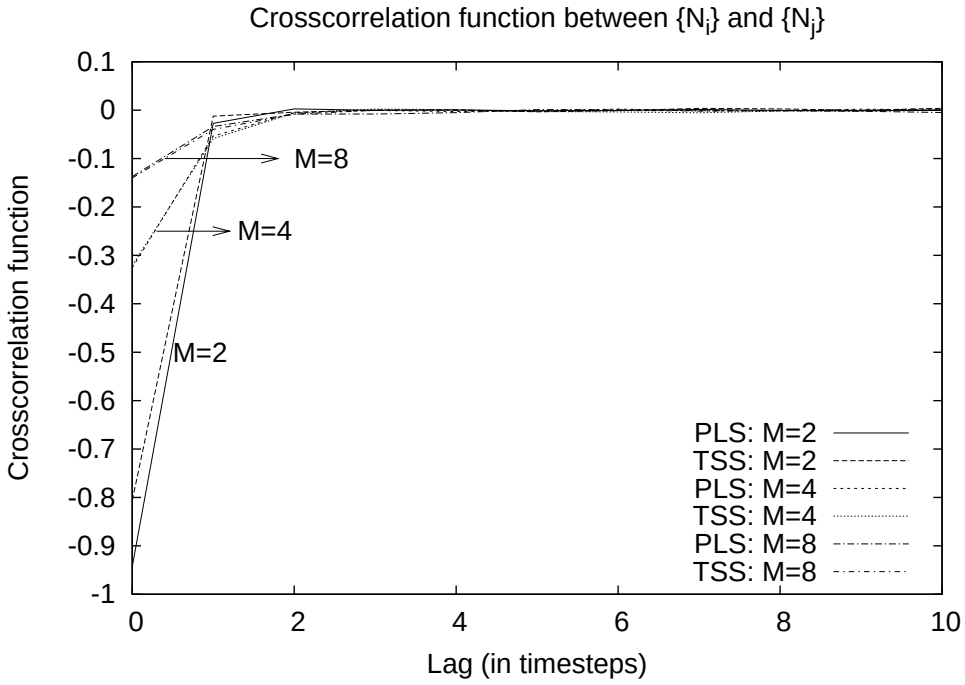


(a)

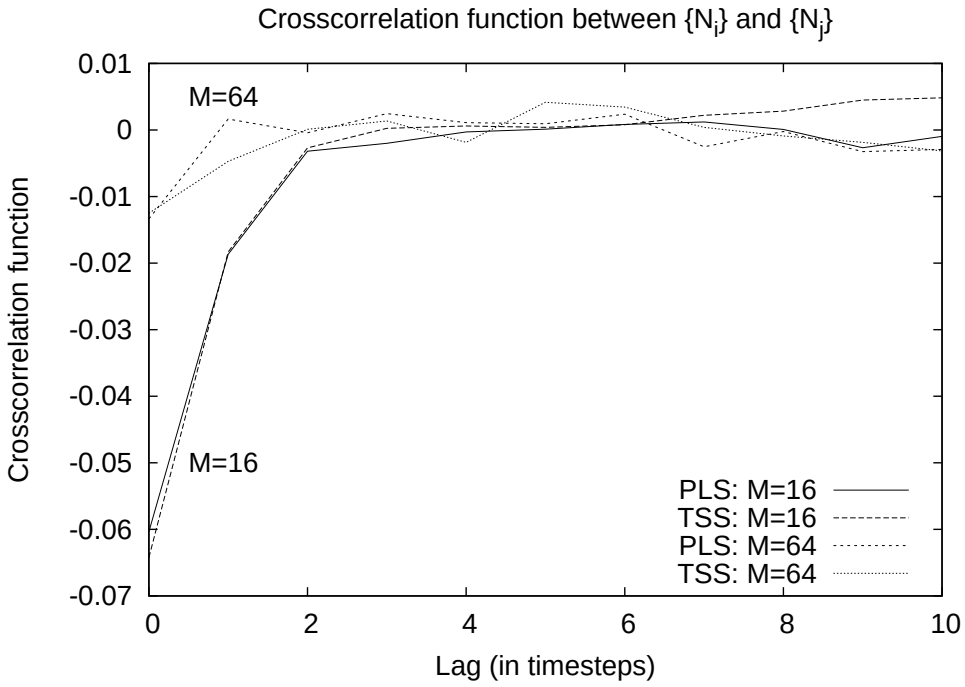


(b)

Fig. 19. Autocorrelation function obtained from samples $C_i(0), C_i(\delta), C_i(2\delta), \cdots$ of one sample path for various values of M.

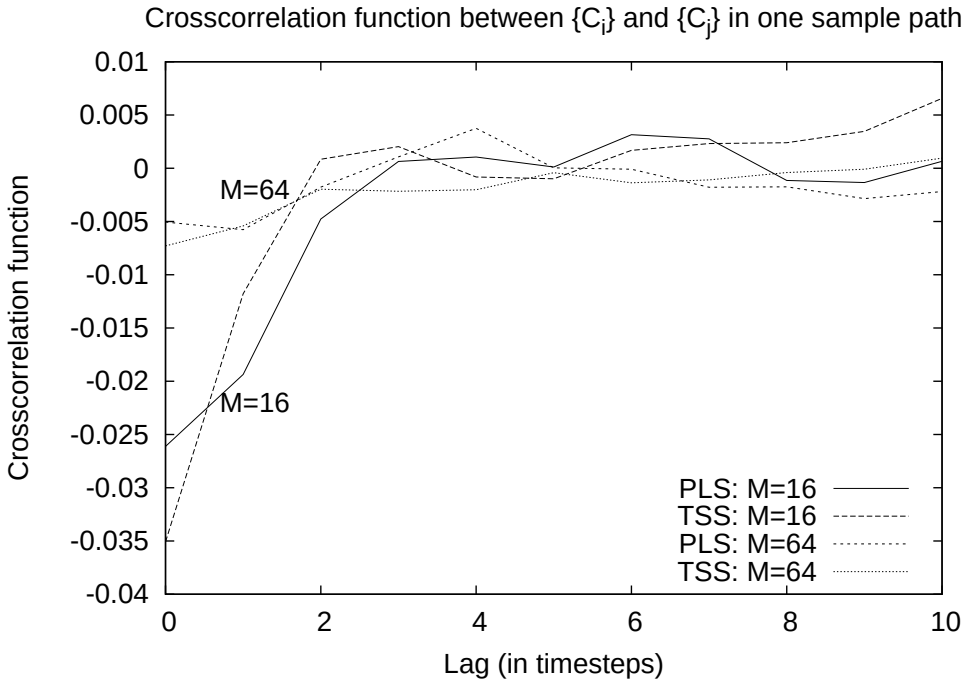


(a)

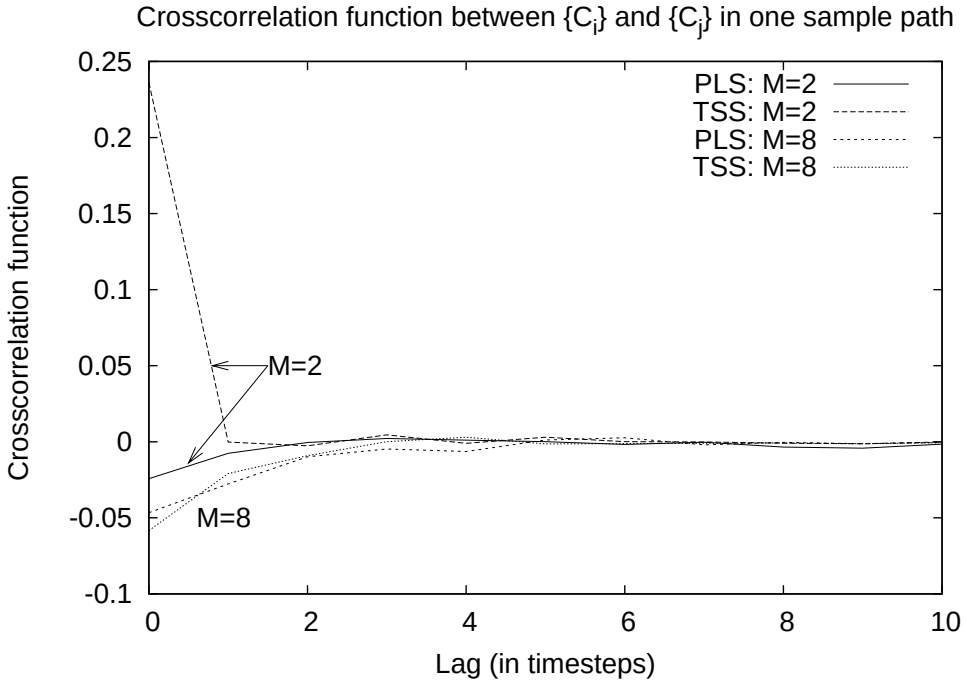


(b)

Fig. 20. Crosscorrelation function obtained from samples $N_i(0), N_i(\delta), N_i(2\delta), \cdots$ and $N_j(0), N_j(\delta), N_j(2\delta), \cdots$ of one sample path for varying values of M.

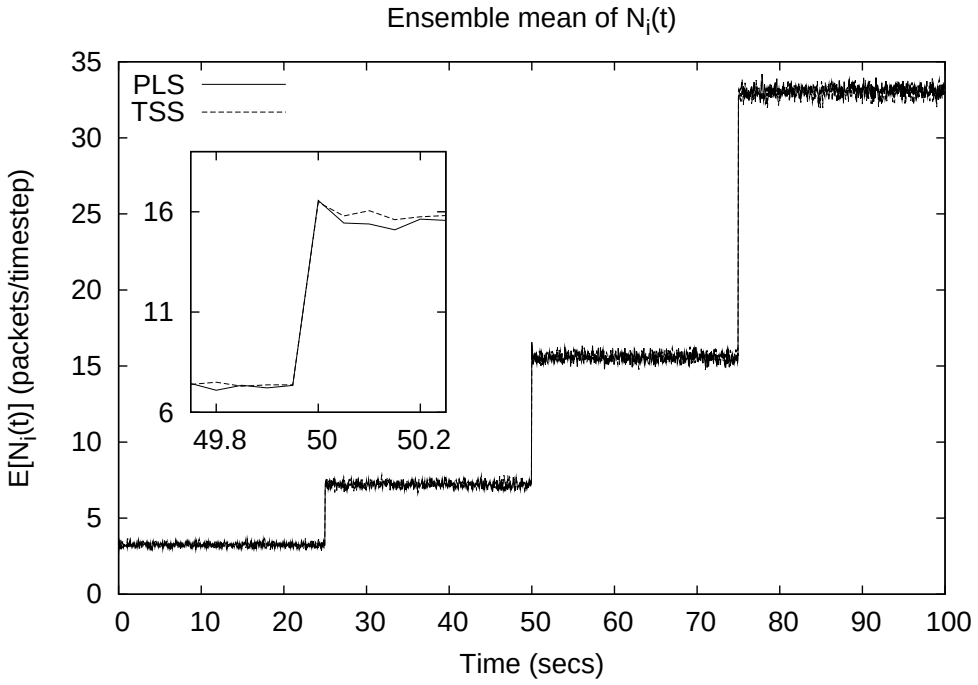


(a)

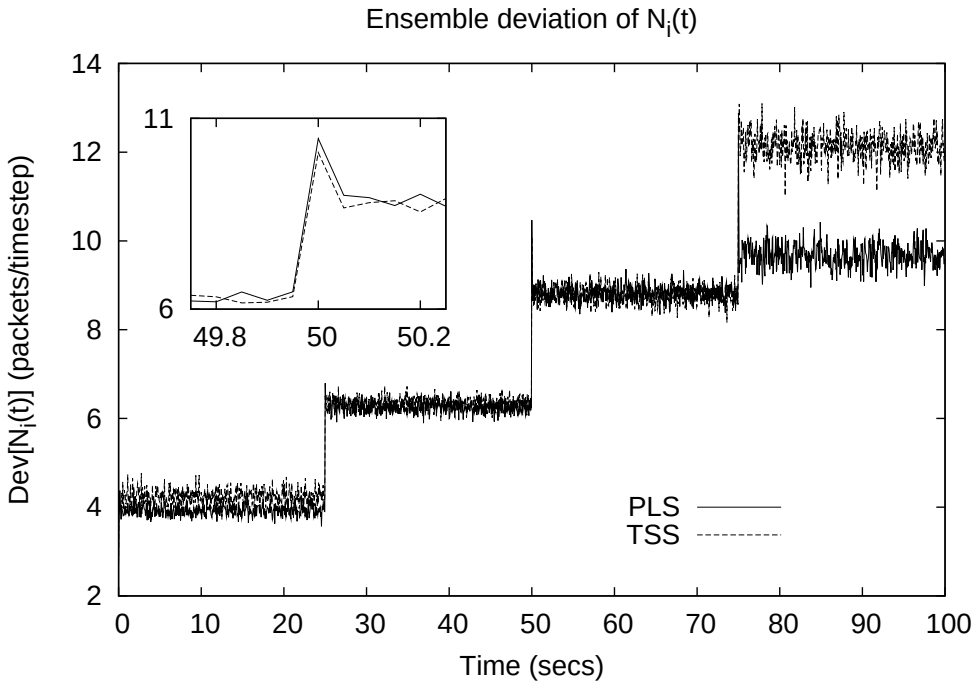


(b)

Fig. 21. Crosscorrelation function obtained from samples $C_i(0), C_i(\delta), C_i(2\delta), \cdots$ and $C_j(0), C_j(\delta), C_j(2\delta), \cdots$ of one sample path for various values of M.



(a)



(b)

Fig. 22. Time evolution of the ensemble mean and deviation of $N_i(t)$ for a tagged station i with time-varying M. After every 25s, M is halved.

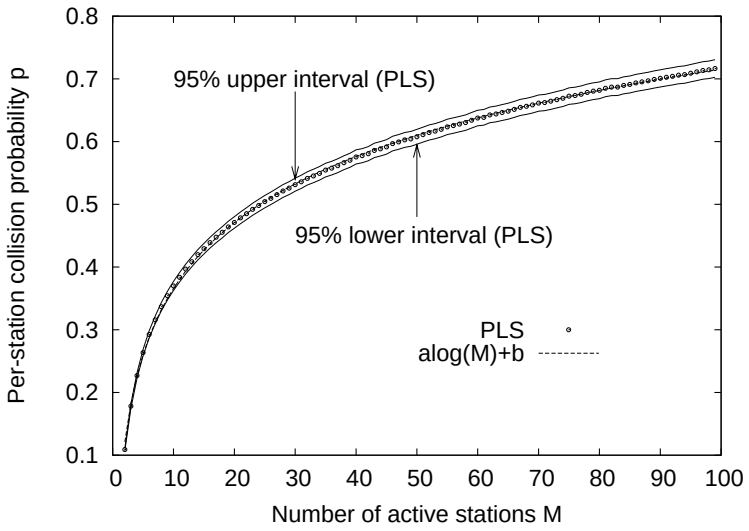


Fig. 23. $p(M)$ as obtained from simulations and an analytical fit of $0.1519\log_e(M)+0.0159$ for various values of M and $\beta = 7$. The 95% confidence interval of each simulation point is within 2% of the mean.

I. Per-station collision probability

References [2], [3], [6], [9] all provide a formula for computing the per-station collision probability as an implicit function of M. While one can use a fixed point iteration to obtain the per-station collision probability from the implicit function of M, we are interested in a simple closed-form expression. We obtained the per-station collision probability as a function of M for varying M by PLS and used MATLAB to fit the curve $0.1519\log(M)+0.0159$. Figure 23 shows the curves obtained by both simulation and the logarithmic fit for $M = 1..100$. In one run of the simulation, all M stations were active throughout an interval of length 100s and a tagged station's collision rate was obtained as a sample of the per-station collision probability for that run. Each point on the simulation curve is an average of 100 such runs. The 95% confidence interval of each simulation point is within 2% of the mean. Figure 24 compares the fit of $\min(0.1519\log(M)+0.0159, 1)$ with simulations for $M = \{100, 200, \ldots, 1000\}$. The two curves diverge above 400 stations around $p = 0.93$, the simulation based curve goes to one slower than the analytical fit.

We consider the question "Why does a logarithmic fit work?" in the appendix. Briefly, reference [2] obtains a closed form expression for the per-station collision probability when $\beta \to \infty$ using the Lambert function $\mathcal{W}$ ($\mathcal{W}(c) = x$ s.t. $xe^x = c$). We extend this work and present an approximate analysis for the per-station collision probability with finite β. The logarithmic fit works because it fits the expression involving the Lambert function that occurs in the function $p(M)$.

J. Convolution Algorithm

To estimate the order of speedup achieved by the convolution algorithm, we obtained 100 samples of the time taken to compute a 20-fold convolution for a per-station collision probability of 0.4. MATLAB's

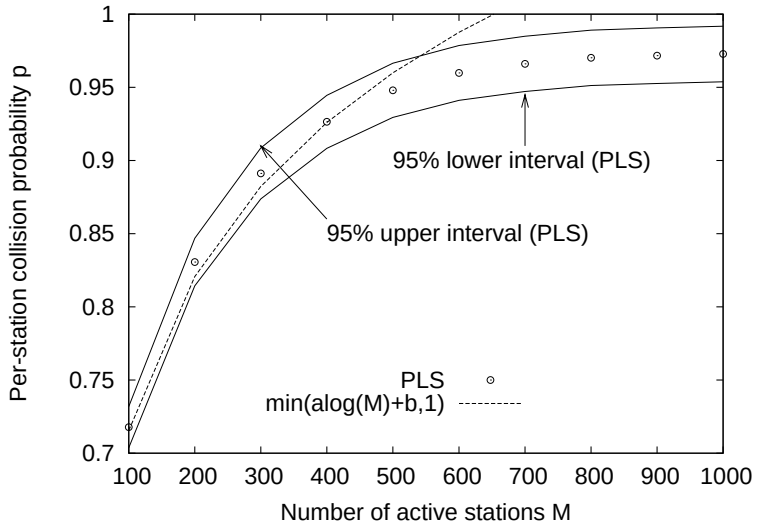


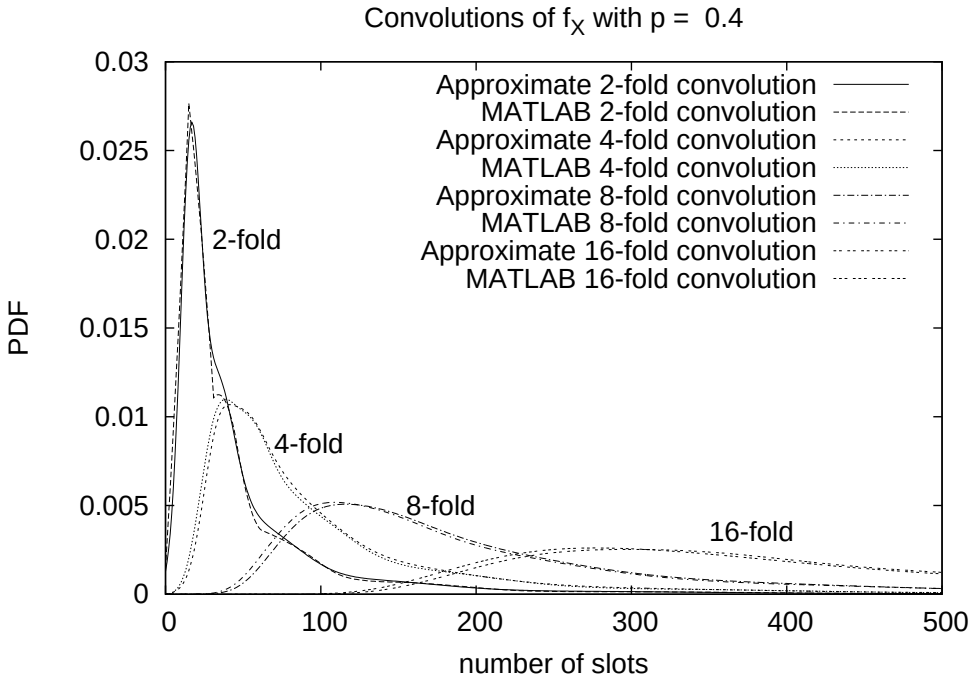
Fig. 24. $p(M)$ as obtained from simulations and an analytical fit of $\min(0.1519 \log_e(M)+0.0159, 1)$ for $M = \{100, 200, \ldots, 1000\}$ and $\beta = 7$. The two curves diverge after $M = 400$ stations at a per-station collision probability greater than 0.935.

FFT-based convolution (with the script launched from the command line to avoid any overheads due to MATLAB's GUI) takes a mean time of 2.4s (deviation 8ms) to compute a 20-fold convolution without any logging to file, while our approach takes a mean duration of 1.19s (deviation 7ms). With file logging enabled, our approach takes a mean time of 0.12s (deviation 2ms) for a 2-fold convolution while MATLAB takes 6.48s (deviation 12ms).

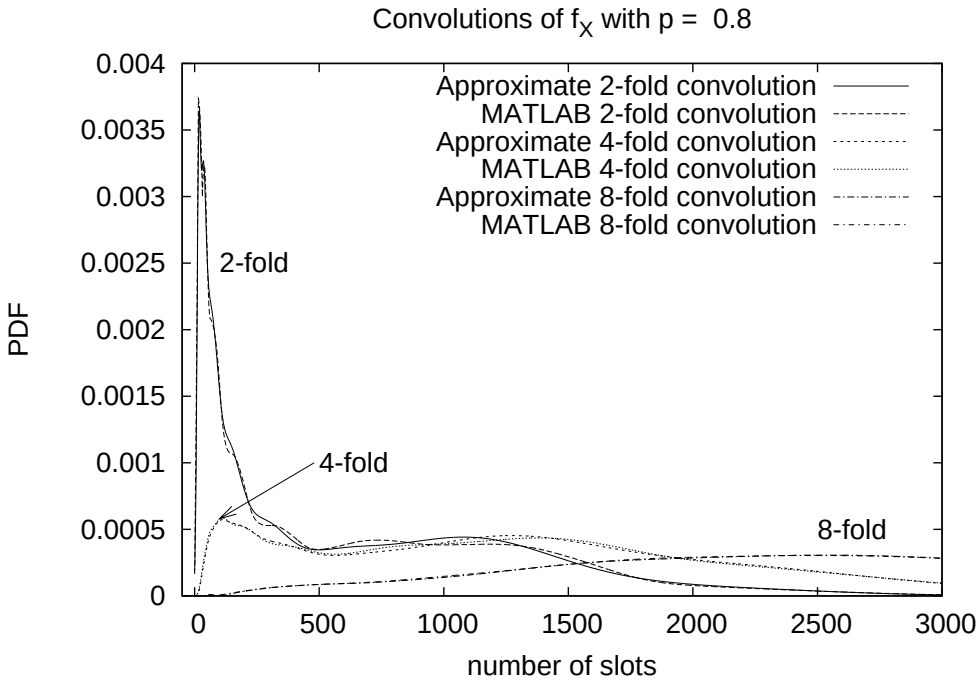
Figure 25(a) compares the approximated distribution of the f_X^n with the pdf obtained by straightforward convolution in MATLAB for $n = \{2, 4, 8, 16\}$. The probability of collision is 0.4. Note how the tail of the distribution is faithfully reproduced by the analytical approximation. In a realistic probability regime ($p < 0.5$), there are few modes in the convolution's pdf and thus the approximation works extremely well. Even for a very high per-station collision probability regime, the method works well as can be seen in Figure 25(b) which consider the same convolutions for $p = 0.8$. In this regime, the errors tend to accumulate as the number of convolutions increases because the distributions tend to become multimodal and eventually smoothen out for higher convolutions. For instance, in Figure 26, the analytical approximation predicts a mode around 50 when there is none in reality. However, the total probability mass in $[0, 100]$ is less than 0.0005, which is ignorable for our purposes. Overall, the method is highly accurate for realistic regimes and handles higher collision regimes with sufficient accuracy.

X. Related Work

Several analytical models [3], [6], [9], [21], [19] have been proposed for the evaluation of 802.11 performance. The general approach has been to observe a tagged station between two successful transmissions and estimate the average time taken for the same, thereby obtaining the *average steady state* goodput. Further, all models



(a)



(b)

Fig. 25. Comparisons of n-fold convolution of f_X for $p = 0.4$ and $p = 0.8$ as obtained from our convolution algorithm with that obtained by MATLAB for various n.

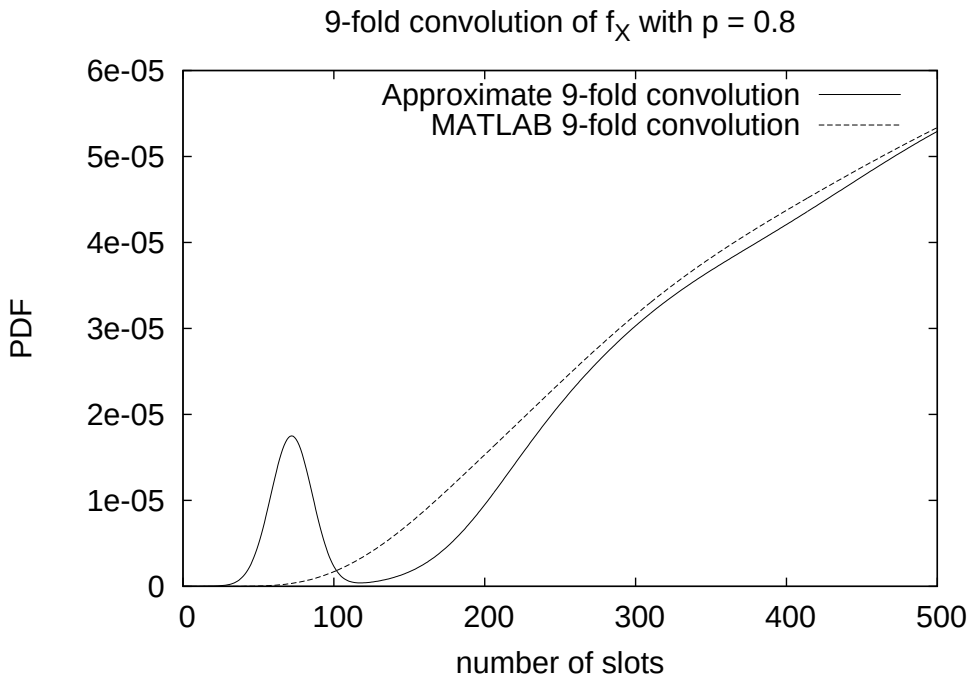


Fig. 26. Comparison of 9-fold convolution of f_X for $p = 0.8$ as obtained from our convolution algorithm with that obtained by MATLAB.

assume that the conditional collision probability, i.e., the probability that a packet encounters a collision given it is transmitted, is constant.

Reference [6] approximates the 802.11 protocol by a persistent CSMA/CA protocol. Every station transmits with probability $1/(E[B]+1)$ in every idle (802.11) slot *independent* of other stations and its own previous attempts; $E[B]$ is the expected backoff duration for a successful packet transmission. A fixed point iteration is used to compute $E[B]$ and the collision probability of this new process, and the authors show that this model approximates 802.11 well using simulations.

Reference [3] considers the stochastic process with state given by the contention window and remaining backoff for a tagged station. This is a discrete time Markov chain under the constant collision probability assumption and is solved to obtain steady state metrics.

Reference [9] provides a "fluid" approximation to 802.11. A "fluid chunk" in their model is the interval between two successful packet transmissions, like in all prior models. The authors estimate the length of the fluid chunk by assuming that the time between transmission attempts is exponentially distributed. Again, the average contention window is obtained iteratively, and the goodput is obtained by obtaining the average length of a "fluid chunk".

Reference [19] obtains the distribution of the inter-arrival time between two frames of a tagged station in an 802.11 system. It approximates the backoff process as a renewal process and uses the remaining time theorem

from renewal theory to obtain the distribution. The main result is that the inter-arrival time distribution is typically multimodal. An equation from [21] is used to relate the probability of collision to the number of stations.

Reference [21] provides the original approximation for the collision probability. It assumes that there are sufficient number of stations so that any two arbitrary stations' transmissions are independent to estimate the collision probability, and therefore, the successful transmission rate.

As mentioned before, existing methods focus on steady-state approximations and yield the average long-term goodput. It is not clear how these methods can yield sample-path evolutions of the system state that approximate packet-level simulation well.

There has been much work on timestepped simulation of point-to-point networks (reference [12] provides a list). For WLANs, reference [10] develops a fluid timestepped deterministic simulator in which the *average* aggregate goodput is the input to the fluid model, and obtains long-run metrics. It is not clear if this model can capture the short-term effects of DCF on TCP workloads.

XI. Future work

Future work includes modeling state-dependent transport layer (like TCP) and a lower layer (a more realistic wireless PHY). For a state-dependent traffic source like TCP, the method needs to provide the metrics of the WLAN link in each timestep. We have obtained a method for predicting the goodput of the link in each timestep. The contribution of the link to the RTT seen by a TCP source is given by the inverse of the goodput. Because the link layer performs retransmissions, the drop-rate would be determined by $p(t)^{\beta}$, which is usually negligible. A TCP source would perceive a crowded 802.11 link as having large and variable RTT and occasional losses. A key issue to be handled while modeling a TCP source over 802.11 would be to handle ACKs and hence multiple-sized packets. Because data and ACKs of a TCP connection running over 802.11 share the same media, the model should handle packets of varying size. This would require replacing the transmission interval τ by the mean of the packet size distribution.

The PHY layer has been modeled implicitly in this technique. Specifically, we assumed that all stations sense each other and all collisions are lost. This sharing of the medium is reflected in the goodputs of flows being correlated. Modeling a general PHY for TSS with several interacting WLANs is challenging for the following reasons:

- Goodputs of two wireless flows are correlated if the transmitter and receiver of one flow influence or are influenced by those of the other. (For instance, goodputs of all stations in a WLAN cell associated with the same AP would be correlated.)
- In order to model PHY induced imbalances, different stations would obtain their goodput distributions in each timestep with different probabilities of collision. That is, the collision probability is generalized to be the loss probability which includes non-collision channel-condition induced losses.

XII. Conclusions

We presented a transient analysis of 802.11 and used it to perform TSS for WLANs. The transient analysis of 802.11 looks at a tagged station within a timestep and obtains conditional pdf of $N_i(t)$ conditioned on the MAC state $C_i(t)$ and that of new MAC state $C_i(t+\delta)$ conditioned on $C_i(t)$ and $N_i(t)$. The TSS technique uses this transient analysis to obtain the instantaneous goodputs of all stations such that they 1) add up to the instantaneous aggregate goodput and 2) have the required correlation structure. In sum, the method obtains the sample path evolutions of the contention windows and instantaneous goodputs of all stations with time. We validate the transient analysis and TSS technique against PLS. TSS scales well with increasing number of stations and is agnostic to bit-rate. Proposed work includes integration of the technique with traffic sources with state dependent control (e.g. TCP) and modeling the PHY layer (with arbitrary location of stations).

Appendix

Recall that λ is the attempt probability (rate), i.e., probability a tagged station starts transmission in an 802.11 slot and p is the probability that a tagged station's transmission encounters a collision. By definition, $\lambda(p) = E[K]/E[X]$, where K is the number of attempts to transmit a packet successfully with maximum number of attempts β or abort, and X is the total backoff duration. It is easy to see that $E[K] = 1 + p + \ldots + p^{\beta-1}$ and $E[X] = \sum_{i=1}^{i=\beta} p^{i-1} \frac{\left(\gamma 2^{i-1} - 1\right)}{2}$.

Consider the following two equations:

$$\lambda(p) = E[K]/E[X] \tag{1}$$

$$p(\lambda) = 1 - (1-\lambda)^{M-1} \tag{2}$$

For each M, equations (1) and (2) can be solved for $\lambda(M)$ and $p(M)$ by a fixed point iteration as in all prior work.

Reference [2] analyses this system under the assumptions $\beta \to \infty$ and $E[X] = \frac{\gamma-1}{2} \sum_{i=1}^{i=\beta} (2p)^{i-1}$. When $\beta \to \infty$, $\lambda(p) \to \frac{2}{\gamma-1}\left(\frac{1-2p}{1-p}\right)$ and a closed form expression can be obtained for the solution $p(M)$ involving the Lambert function (i.e., inverse function for xe^x).

We consider the case where β is finite (in this case, $\beta = 7$). The solution $p(M)$, obtained from simulations, looks as in Figure 23. Specifically, the solution $p(M)$ looks like $0.1519 \log_e(M) + 0.0159$ as fit by MATLAB. Our goal is to show some analytical justification for $p(M)$'s log-like behavior.

We start by approximating $\lambda(p)$ as $2(1-p)^2/(\gamma-1)$ for $\beta = 7$ by matching the actual $\lambda(p)$ and the approximation at $p = 0$. Figure 27 shows the accuracy of this approximation. Clearly, the accuracy can be made better by fitting higher order polynomials, but we stick to a second-order approximation for sake of a simple analytical expression.

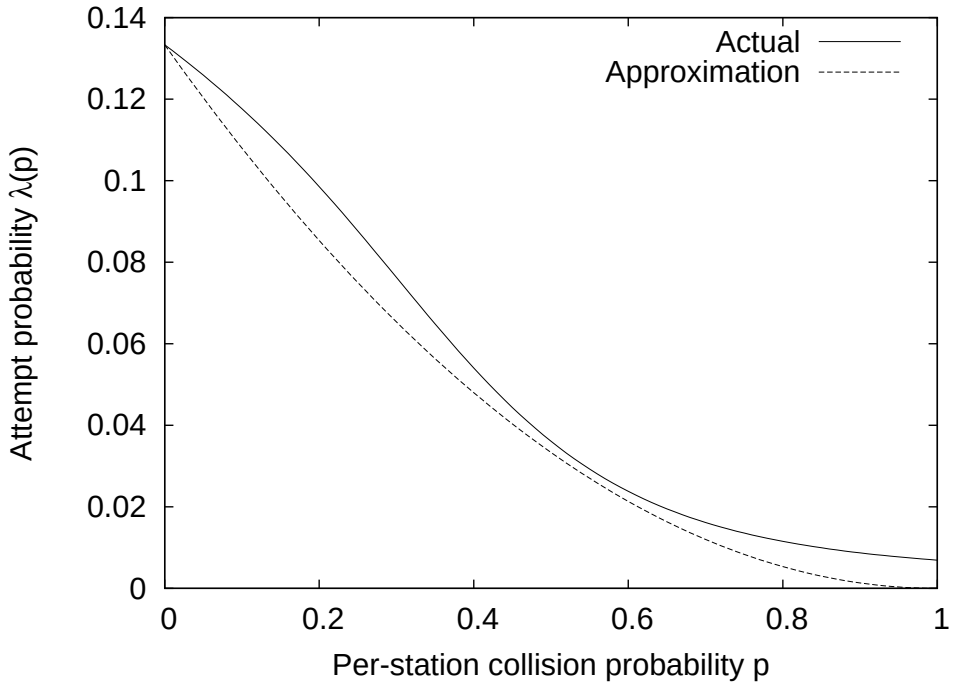


Fig. 27. $\lambda(p)$ can be approximated as $a(1-p)^2$.

We have

$$\begin{aligned} p &= 1-(1-\lambda)^{M-1} \\ &\approx 1-e^{-\lambda(M-1)} \\ &= 1-e^{-\frac{2(1-p)^2}{\gamma-1}(M-1)} \end{aligned}$$

Strictly speaking, the exponential approximation requires that $M\lambda$ go to a constant as $M \to \infty$, i.e., the system operates in a regime where BEB keeps the attempt probability λ to scale as $1/M$ with increasing M. If $\beta \to \infty$ this condition is satisfied, but for finite β, as $M \to \infty$, $p \to 1$ and $\lambda \to \lambda^* > 0$. However, even with this approximation we are able to obtain intuition on how $p(M)$ behaves with increasing M for finite β.

Rearranging terms we have $(1-p)e^{2(1-p)^2(M-1)/\gamma-1} = 1$. Let $\mathcal{W}$ denote the Lambert function, i.e., $x = \mathcal{W}(c)$ is the solution of the equation $xe^x = c$. We want to solve an equation of the form $x^a e^{bx} = 1$. Straightforward algebraic manipulations yield the solution as $x = \frac{a}{b}\mathcal{W}(\frac{b}{a})$. In our context, $x = (1-p)^2$, $a = 1/2$, and $b = 2(M-1)/\gamma - 1)$. Thus $(1-p)^2 = \frac{\gamma-1}{4(M-1)}\mathcal{W}\left(\frac{4(M-1)}{\gamma-1}\right)$ In sum,

$$p = 1 - \sqrt{\frac{\gamma-1}{4(M-1)}\mathcal{W}\left(\frac{4(M-1)}{\gamma-1}\right)} \tag{3}$$

This expression for $p(M)$ function can be approximated by $a\log(M)+b$. We confirm this by plotting $p(M)$ as predicted by this analysis and computed using MATLAB, $p(M)$ as computed by the logarithmic

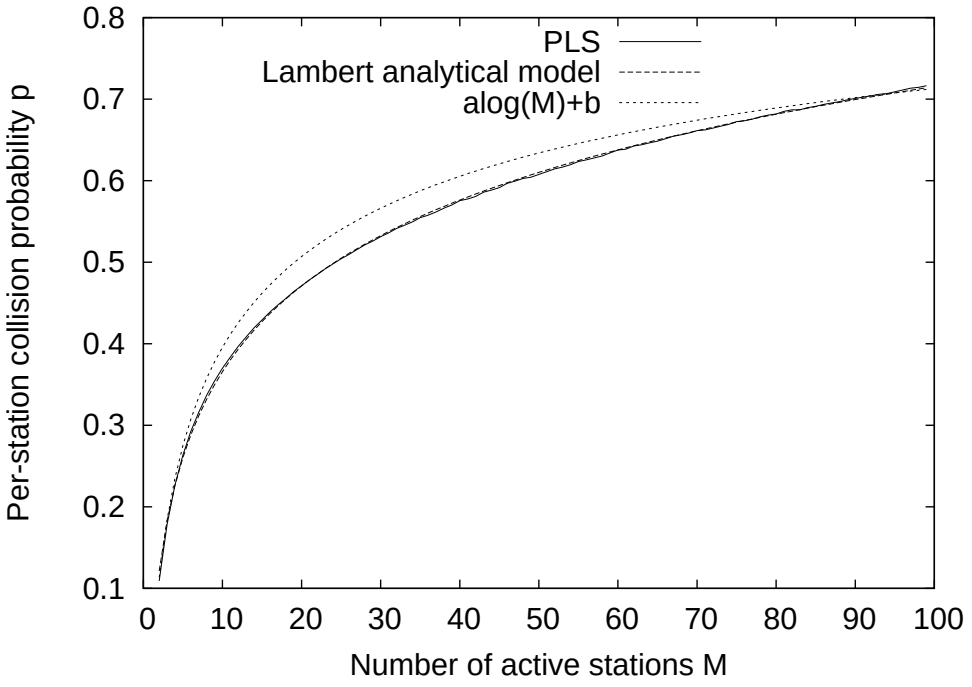


Fig. 28. Confirming the prediction of the model with simulation studies

approximation, and $p(M)$ as obtained by simulations in Figure 28. The model works best when $p < 0.6$ approximately. A more refined model that works better for higher p can be obtained by considering a fit for $\lambda(p)$ of the form $\lambda + \lambda^2/2 = a(1-p)^2$ and $1 - p = e^{-(M-1)(\lambda+\lambda^2/2)}$. This gives a similar Lambert based solution.

References

[1] NS-2 network simulator. http://www.isi.edu/nsnam/ns.

[2] A. Kumar, E. Altman, D. Miorandi and M. Goyal. New Insights from a Fixed Point Analysis of Single Cell IEEE 802.11 WLANs. In *Proceedings of IEEE INFOCOM*, 2006.

[3] G. Bianchi. Performance analysis of the IEEE 802.11 Distributed Coordination Function. In *IEEE Journal On Selected Areas in Communications*, March 2000.

[4] D. Malone, K. Duffy, and D. Leith. Modeling the 802.11 distributed coordination function in non-saturated heterogeneous conditions. *IEEE/ACM Transactions on Networking.*, 15, February 2007.

[5] D.E.Knuth. *The Art of Computer Programming: Seminumerical Algorithms.* Addison-Wesley Publishing Company, 1981.

[6] F.Cali, M. Conti, and E. Gregori. IEEE 802.11 Wireless LAN: Capacity Analysis and Protocol Enhancement. In *Proceedings of INFOCOM*, 1998.

[7] G. Berger-Sabbatel, A. Duda, O. Gaudoin, M. Heusse, and F. Rousseau. Fairness and its Impact on Delay in 802.11 Networks. In *Proceedings of IEEE GLOBECOM*, 2004.

[8] G. Berger-Sabbatel, A. Duda, O. Gaudoin, M. Heusse, and F. Rousseau. Short-Term Fairness of 802.11 Networks with Several Hosts. In *Proceedings of the Sixth IFIP IEEE International Conference on Mobile and Wireless Communication Networks, MWCN*, 2004.

[9] H.Kim and J.Hou. Improving protocol capacity with model-based frame scheduling in ieee 802.11-operated wlans. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom)*, 2003.

[10] H.Kim and J.Hou. A fast simulation framework for IEEE 802.11-operated wireless LANs. In *Proceedings of the joint international conference on Measurement and modeling of computer systems ACM SIGMETRICS/PERFORMANCE*, 2004.

[11] K. Medepalli and F. A. Tobagi. Throughput analysis of ieee 802.11 wireless lans using an average cycle time approach. In *Proceedings of IEEE GLOBECOM*, 2005.

[12] A. Kochut and A.U. Shankar. Timestep Stochastic Simulation of Computer Networks using Diffusion Approximation. In *Proceedings of IEEE/ACM MASCOTS 2006, 14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Monterey, California*, 2006.

[13] Andrzej Kochut. *Timestep Stochastic Simulation of Computer Networks using Diffusion Approximation*. PhD thesis, University of Maryland, August 2005.

[14] Benoit Mandelbrot. The Pareto-Levy Law and the Distribution of Income. *International Economic Review*, 1960.

[15] Benoit Mandelbrot. Summary of Variation of Certain Speculative Prices. *Journal of Business*, 1963.

[16] The Institute of Electrical and Electronic Engineers Inc. IEEE 802.11, 1999 edition (ISO/IEC 8802-11:1999). *http://standards.ieee.org/getieee802/802.11.html*.

[17] R. Jain, D. Chiu, and W. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *DEC Research Report TR-301*, 1984.

[18] S. M. Ross. *Introduction to Probability Models*. Academic Press, Inc.

[19] O. Tickoo and B. Sikdar. ‘‘On the Impact of IEEE 802.11 MAC on Traffic Characteristics”. *IEEE Journal on Selected Areas in Communications*, 21(2):189–203, February 2003.

[20] V. Ramaiyan, A. Kumar, and E. Altman. Fixed Point Analysis of Single Cell IEEE 802.11e WLANs: Uniqueness, Multistability and Throughput Differentiation. In *Proceedings of ACM SIGMETRICS*, 2006.

[21] Y.C.Tay and K.C. Chua. A Capacity analysis for the IEEE 802.11 MAC protocol. In *Wireless Networks*, January 2001.

[22] Z. Li, S. Nandi, and A. K. Gupta. Modeling the Short-Term Unfairness of IEEE 802.11 in Presence of Hidden Terminals. In *NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, 2004.