ABSTRACT

Title of Dissertation: Timestepped Stochastic Simulation of 802.11 WLANs

Arunchandar Vasan, Doctor of Philosophy, 2008

Dissertation directed by: Prof. A. Udaya Shankar
Department of Computer Science

Performance evaluation of computer networks is primarily done using packet-level simulation because analytical methods typically cannot adequately capture the combination of state-dependent control mechanisms (such as TCP congestion control) and stochastic behavior exhibited by networks. However, packet-level simulation becomes prohibitively expensive as link speeds, workloads, and network size increase. Timestepped Stochastic Simulation (TSS) is a novel technique that overcomes the 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 $\mathsf{Pr}(\, \mathbf{S}(t+\delta) \mid \mathbf{S}(t) \,)$ is obtained analytically, and $\mathbf{S}(t+\delta)$ is sampled from it.

This dissertation presents TSS for shared links, specifically, 802.11 WLAN links. Our method computes sample paths of instantaneous goodput (successful

transmissions per timestep) $N_i(t)$ 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 round-trip time). At typical values of δ (e.g, 50ms), $N_i(t)$'s are correlated across timesteps (e.g., a station with high contention window has low goodput for several timesteps) as well as across 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 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.

Timestepped Stochastic Simulation of 802.11 WLANs

by

Arunchandar Vasan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:

Prof. A. Udaya Shankar, Chairman/Advisor
Prof. Ashok Agrawala
Prof. Samrat Bhattacharjee
Prof. Raymond Miller
Prof. Charles Silio - Dean's representative

DEDICATION

To Ma, Pa, and Dippy.

ACKNOWLEDGEMENTS

Beggar that I am,

I am even poor in thanks;

But I thank you.

- William Shakespeare, *Hamlet*

My first thanks go to my "boss", Prof. Udaya Shankar. A constant willingness to help and patience were his two biggest virtues, the latter especially so when I waited for the Muse to whisper. Also worthy of mention are his candor, and down-to-earth approach to research and life in general.

Prof. Ashok Agrawala has been a source of ever-available feedback right from my earliest days at College Park. He has inculcated a *esprit de corps* at the MIND lab that has made me feel like I belong as well.

Prof. Ray Miller encouraged me to publish my first paper and guided me every step of the way. His insights about computing as a discipline, delivered from a doyen's perspective, have always served to inspire.

Prof. Bobby Bhattacharjee had this to say at our very first interaction: “Don’t lose your passport”. Ever since, sound advice in practical stuff, be it hacking kernels or choosing careers, has been his hallmark.

Prof. Aravind Srinivasan’s class on randomized algorithms weaned me from my innate tendency to think deterministically. I am thankful for his many technical suggestions and optimistic advice on career choices.

Prof. Atif Memon guided me in publishing my term paper in his class and, in that process, taught me what it takes to be thoroughly professional. Yet, he always lightened up the proceedings with a joke or two.

I thank Prof. Charles Silio for serving as the Dean’s representative in my defense committee. Dr. Ram Ramjee deserves my thanks for two fun and fruitful summer internships at Bell Labs, and for his very useful inputs about research in the industry.

My apartment-department orbit has thankfully intersected with those of many friendly people. My many room-mates over the years - Vijay Gopalakrishnan, Alap Karapurkar, Christopher Kommareddy, Srinivasan Parthasarathy, Guruprasad Pundoor, Prithviraj Sen, Vinay Shet, Sameer Shirdhonkar, Sadagopan Srinivasan, and Aparna Sundaram - have made my residential experience pleasant. My school buddies - Christian Almazan, Indrajit Bhattacharya, Aniket Dutta, Neha Gupta, Fatih Kaya, Andrzej Kochut, Tom Krug, Seungjoon Lee, Elvita Lobo, Matthew Ma, Bharath Madhusudan, Arunesh Mishra, Archana Ragotthaman, Mustafa Tikir, Bao Trinh, Chadd Williams, Chang-Shieh “Joe” Wu, Yuan Yuan, and Moustafa Youssef - ensured

that school was not all work. Andrzej Kochut and Moustafa Youssef have also been great researchers to work with. Renatta Kochut tolerated our experiments at hours that can charitably be called insane. Fatima Bangura, Jodie Gray, Gwen Kaye, Heather Murray, and Jenny Story ensured that admininstrative issues were handled very smoothly. Adelaide Findlay and Brenda Chick were ever so cheerful in responding to all logistics requests.

I thank my other friends and extended family both here in the US and elsewhere for their support and encouragement. Last but most, I would like to "thank" my parents and sister: But for them, I simply wouldn't have been patient enough to complete this.

I can honestly blame the toll these years have taken on my memory for any omissions of names here. In my defense, I only offer my apologies and paraphrase what Saint Tyagaraja sang on the fertile banks of the Cauvery: "To all helpful souls, my thanks".

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Performance evaluation is a *sine qua non* for the design, deployment, and evolution of computer networks. Modern computer networks exhibit stochastic behavior due to randomness in user activity, variations in router service time, random access based link scheduling, randomized queue management and application protocols, etc. They also also rely extensively on non-linear state-dependent control mechanisms (e.g., TCP). These mechanisms operate on small timescales (e.g., the round-trip time of a TCP connection). The use of such control means that small changes in the system state can lead to large changes in the system state over time. Thus any performance evaluation technique must adequately handle the stochastic nature and state-dependent control of computer networks.

The two main approaches to performance evaluation of computer networks have been analytical modeling and packet-level simulation. Of these, packet-level simulation has been the workhorse of choice primarily because analytical methods have typically been unable to capture state-dependent control mechanisms adequately. Purely analytical methods simplify the model for the sake of tractability (e.g., Poisson arrivals, stationarity, etc.) so much that their predictions are considered by most to be too inaccurate for performance evaluation

purposes. However, packet-level simulation becomes prohibitively expensive as link speeds, workloads, and network size increase [19] because it simulates every packet's arrival and departure at all relevant elements of the network.

This situation has motivated several simulation techniques based on fluid approximation (e.g., [11, 30, 41, 48]). These methods obtain the evolution of the system state at timesteps rather than at packet-level detail, and thus are faster than packet-level simulators. However, they do not capture the stochastic nature of networks. This is because they replace the random system state by its expected value. Thus these methods yield an evolution of the system state, which is assumed to be representative of all possible evolutions of the system in some ensemble averaged sense. Because these methods do not yield individual sample paths, they are limited to systems in which the sample paths are "close" to the representative evolution predicted by them, but cannot handle those with state-dependent control. In particular, sample path metrics may be completely off from the ensemble averaged evolution.

We schematically illustrate the limitation of fluid approaches with an example. Figure 1.1 shows the transfer time for a file using TCP over a link. The link delay shown is that seen by a station in a typical 802.11 WLAN. Because of the stochastic nature of the link delay, *the transfer time is a random variable.* This is shown in the upper part of the figure labeled "Stochastic". A fluid model of this link would replace the stochastic link delay with its expectation. Thus a *fluid model would predict the total transfer time as a fixed value.* This is shown in the lower part of the figure labeled "Fluid". Clearly, a stochastic link model is required to obtain a more accurate representation of the total transfer time.

Timestepped Stochastic Simulation (TSS) is a method to achieve the model-

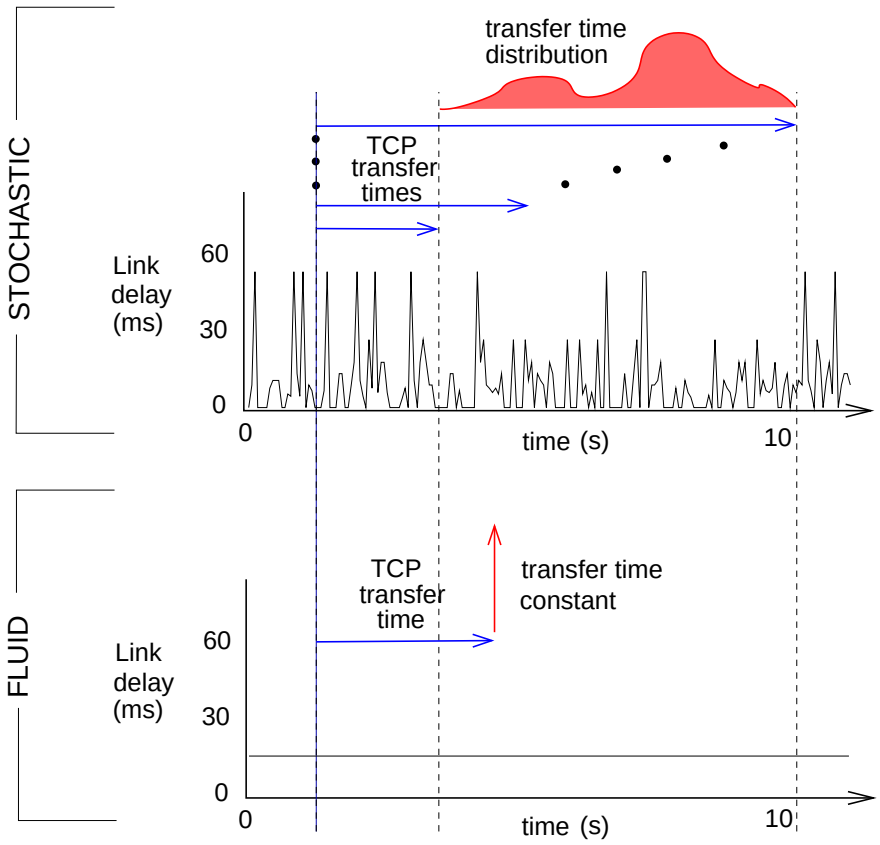


Figure 1.1: Schematic representation of TCP transfer time according to fluid and stochastic link models. A stochastic link model gives a distribution of values for the total transfer time, while a fluid based link model gives one fixed value. The link delay shown is that seen by a station in a typical 802.11 WLAN.

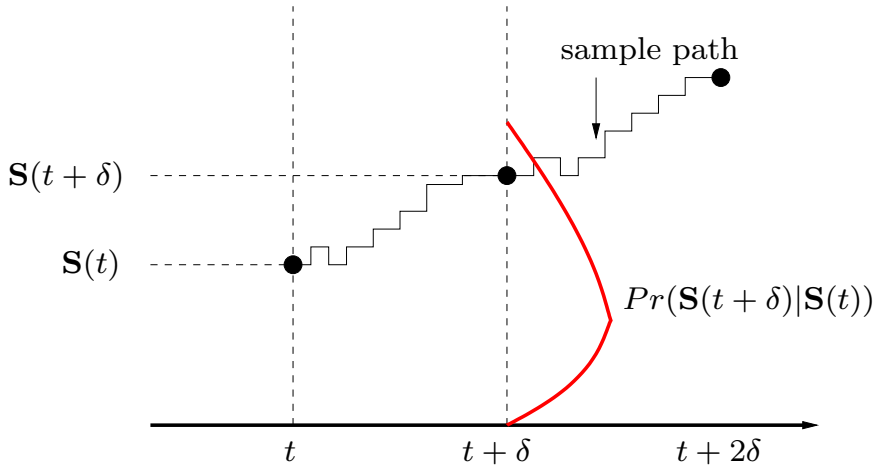


Figure 1.2: In each step of TSS, the conditional distribution of the new system state $\mathbf{S}(t+\delta)$ given the old system state $\mathbf{S}(t)$ is obtained, and the new state is sampled from it.

ing accuracy of packet-level 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, as in fluid based approaches, 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. Figure 1.2 illustrates one timestep of TSS. In order for the time-invariance assumption to hold, δ has to be lesser than the feedback time-scale of the end-to-end control mechanisms employed (e.g., TCP's round-trip time). We use $\delta = 50ms$, which is reasonable considering the round-trip time of typical TCP connections. Because TSS updates the system state probabilistically, it can model both the stochastic behavior and state-dependent control accurately.

TSS has been developed for networks of point-to-point links [37, 36]. There,

the diffusion approximation (proposed by Kolmogorov [38] and later extended by Feller [21]) is used to obtain the distribution of the queue size of a single link's queue at the end of a timestep $[t, t + \delta]$ conditioned on the queue size at time t as a function of the first two moments of the arrival and service time distributions. This method is then extended to point-to-point networks by various approximations that yield the first two moments of a queue's output processes, as well as the first two moments of processes formed by splitting or merging processes.

This dissertation extends TSS to **shared links**, where a link is shared among many senders through the use of a Media Access Control (MAC) protocol. The presence of the MAC protocol introduces correlation in the the service times of the senders' output queues. The nature of the correlation depends on the type of the MAC protocol. In the case of a deterministic time division multiple access, the problem reduces to a each sender having an independent queue, like in the point-to-point case, and the evolution of a station's output queue can be readily obtained in isolation. At the other extreme, in the case of a stateless random access protocol (e.g., like ALOHA [66]), the service process of a station can be easily approximated in terms of the load offered by the rest of the stations. However, a MAC protocol that tries to retain the best features of both time division and random access multiplexing is not amenable to such straightforward analysis.

The 802.11 Distributed Coordination Function (DCF) [49, 43, 66], which is the basic MAC protocol in all WLANs, is an example of such a MAC protocol. It provides random access augmented with a history. This dissertation presents a method to perform TSS of 802.11 wireless networks (WLANs). The 802.11

DCF is a variant of Carrier Sense Multiple Access (CSMA) [66, 62, 35] that is designed to perform Collision Avoidance (CA). The stations in a WLAN operate on a common slotted timeline. 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 given by 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. When station i has a packet to send, it 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 Chapter 2.)

Consider an 802.11 WLAN 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 $N_i(t)$ denote the **goodput** of station i in timestep $[t, t+\delta]$, defined as the number of packets successfully transmitted by station i in the timestep. (The **throughput** of station i in the timestep includes all unsuccessful attempts as well.) $N_i(t)$ 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.

Our method computes evolutions of the goodputs $N_i(t)$'s for all stations i 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. The current MAC state influences the future attempts (rate) made by a station, and in turn, depends on the attempts and successes of a station in the past. Thus there is a positive correlation in a station's goodput $N_i(t)$ across timesteps t. Because all active stations share the same channel and have load available throughout the timestep, a high goodput for a station necessarily implies that the goodputs of other stations have to go down. Hence, there is negative correlation between the goodputs $N_i(t)$'s of stations across i within a 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 goodputs and DCF states **of all stations** 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 packet-level simulations 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 speedup 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. We obtain this in the following main steps:

- **Step 1:** Obtain the distribution $\mathsf{Pr}(\, N_A(t)\,)$ of the **aggregate goodput**

$N_A(t) \triangleq \sum N_i(t)$, and sample $N_A(t)$ from it.

- **Step 2:** For each active station i, obtain the marginal goodput distribution of $N_i(t)$ given its MAC state at t.

- **Step 3:** Dependently sample $N_i(t)$ from the marginal goodput distributions for all i such that the sampled $N_i(t)$'s are correlated and add up to $N_A(t)$.

- **Step 4:** For each active station i, obtain the new MAC state distribution at $t + \delta$ conditioned on the old MAC state at t and its goodput sample $N_i(t)$, and sample the new MAC state from it.

Because all probability distributions involved can be parametrized in terms of the number of active stations and the timestep length δ, they can be precomputed or cached across simulation runs. In the computation of each timestep of TSS, the random sampling from the distributions is independent of the underlying transmission bit-rate, and this explains the scalability of TSS.

1.1 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 this dissertation 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., [8, 20, 29, 69, 58]) obtain the average aggregate steady-state goodput over a sufficiently long interval of time.

- We obtain the distribution of the instantaneous aggregate goodput 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 [3] which considers infinite number of retries) and present a simple logarithmic approximation for this function.

1.2 Organization of the dissertation

Chapter 2 introduces the notation and explains the operation of the DCF protocol. It identifies the two key operating states of the channel: transmission times and varying idle intervals determined by the backoff mechanism. Chapter 3 examines related work, focusing on various simulation studies and analytical models for 802.11

Chapter 4 first states the modeling assumptions and then presents an overview of the algorithm for TSS of WLANs accounting for correlations across stations and time.

Chapters 5 through 9 deal with various components of the TSS algorithm. Chapter 5 deals with the distribution of instantaneous aggregate goodput. Using the idle interval distribution, the first two moments of the time taken for one success in the aggregate attempt process are obtained, using which, the distribution of the instantaneous aggregate goodput is obtained.

Chapter 6 obtains the marginal distribution of the instantaneous goodput of a tagged station in a timestep conditioned on its MAC state at the beginning of the timestep using the distribution of the total backoff duration for one successful transmission of a tagged packet.

Chapter 7 obtains the distribution of the total backoff duration of a tagged packet and presents an algorithm to obtain its n-fold convolutions; this is used in Chapter 6. The discrete pdf of the total backoff duration is approximated in terms of a gaussian mixture, and the structure of this mixture distribution is exploited to obtain the convolution efficiently.

Chapter 8 explains how we obtain correlated samples of $N_i(t)$ by dependently sampling the marginal distributions. Specifically, the method considers stations according to a random permutation of their id's. Each station is allocated goodput from a specific part of its marginal distribution depending on the sum of all goodputs allocated prior to it.

Chapter 9 explains how we obtain the new MAC state given the old MAC state and goodput from the previous timestep. We first obtain the pdf of the last time instant within a timestep when the MAC state is reset. From this, the new MAC state distribution is obtained.

Chapter 10 puts all the pieces of analysis together, and presents the pseudocode for the TSS simulator. Chapter 11 deals with the speedup obtained by TSS

over PLS. It first quantifies the the memory requirement and the time taken for the precomputation of pdf's in TSS. Then it compares the time taken by TSS, inclusive of the time to load the precomputed pdf's from disk and the amortized precomputation time, against the time taken by a custom packet-level simulator.

Chapter 12 compares the metrics obtained by TSS against those obtained by PLS for fixed number of active stations, while Chapter 13 does the same for simulation scenarios with varying number of active stations. Chapter 14 discusses possible extensions and concludes.

Chapter 2

Overview of 802.11 DCF

In this chapter, we present an overview of the 802.11 Distributed Coordination Function (DCF). An 802.11 network evolves in slotted time (of $9\mu s$ slots for 802.11a). 802.11 allows for the use of basic physical carrier sensing and so-called virtual carrier sensing. In the basic physical carrier sensing, the channel is sensed continuously for an 802.11 slot by the physical-layer hardware, and towards the end of the slot is declared as having been idle or busy for that slot. Virtual carrier sensing comes to play when the RTS/CTS option is used. Here, the channel state is marked (as part of a station's MAC state) to be busy by the exchange of RTS (Request-To-Send) and CTS (Clear-To-Send) control packets. A station that intends to transmit a data frame, first sends an RTS frame using the DCF protocol; any station that receives an RTS frame intended for it sends the CTS frame if it senses the channel to be idle. Any other station that hears the CTS (RTS) frame, marks the channel to be busy for the duration of the data frame and ACK transmission (and CTS). In this dissertation, we assume that the RTS/CTS mechanism is not employed. The implications of this assumption are discussed later in Chapter 4.

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
For a tagged packet of a station:	
Y_j	backoff duration for j th attempt
For aggregate attempt process:	
I	idle interval;
	variable interval between successive packet
	transmissions when all stations decrement
	their backoff counters

Table 2.1: Notation for 802.11 operation

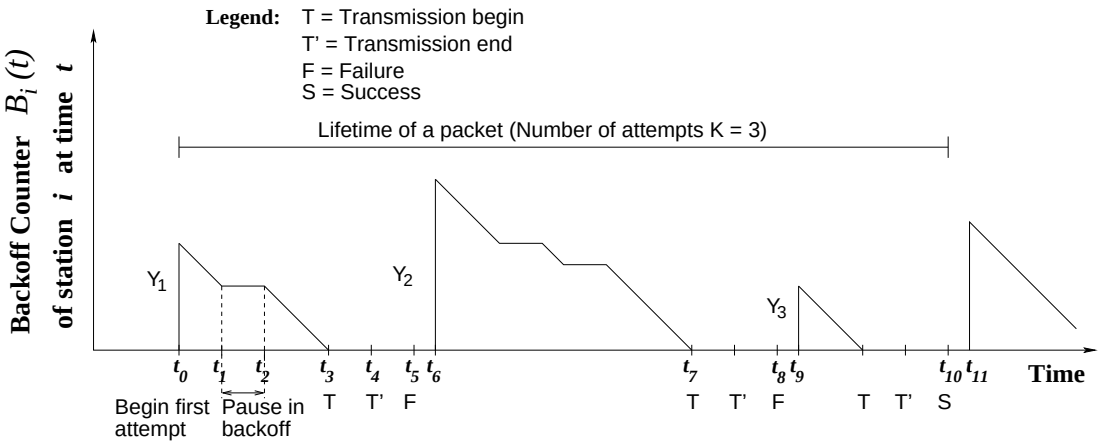


Figure 2.1: Evolution of $B_i(t)$ during a packet's lifetime at station i. $C_i(t)$ changes at t_5 and t_8 to 2γ and 4γ respectively.

2.1 Protocol operation

Each evolution of basic 802.11 DCF (i.e., no RTS/CTS) consists of a sequence of successful or unsuccessful (collision) transmission intervals 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 [49, 43, 66]:

- If a receiving station's physical layer deciphers an 802.11 packet with a checksum error (due to a collision or noise), 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 [49]) 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 [29, 30, 17, 33] do the same. Some prior works (e.g., [8]) 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.

2.2 Evolution of backoff counter

Figure 2.1 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 $t_4 = t_3 + PKT$
- 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 super-position of the per-station attempt processes results in the WLAN-wide **aggregate attempt process** as shown in Figure 2.2. 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 2.2 shows the the WLAN-wide transmissions and the associated timing details during the interval $[t_0, t_6]$ of Figure 2.1. At t_0 station i gets a packet to transmit and starts the backoff procedure. From time t_1 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.

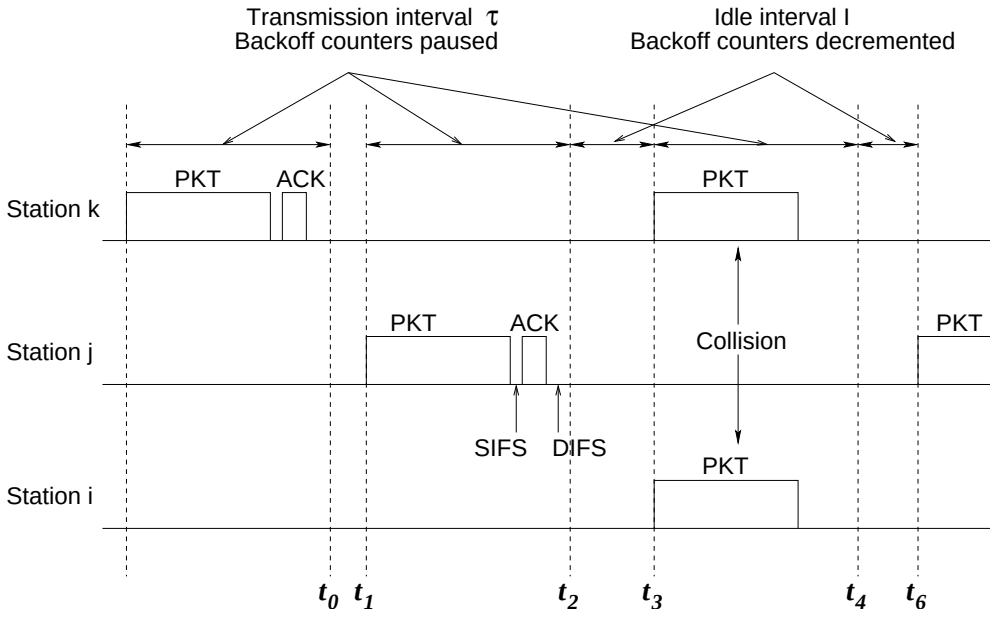


Figure 2.2: Aggregate evolution of a WLAN, with evolution of a tagged station i shown in more detail.

Note that I does not include the fixed overhead $SIFS + ACK + DIFS$ after each packet transmission.

Chapter 3

Related Work

Work related to this dissertation can be broadly categorized into two areas: simulators and simulation studies for 802.11 DCF and analytical modeling of 802.11 DCF. Most simulators for 802.11 operate in a packet-level manner modeling the MAC layer in detail and the PHY layer at various levels of abstraction; a notable exception is the so-called fluid simulator in reference [30] that we discuss later. Analytical performance modeling of 802.11 WLANs is primarily done through examining one tagged station and tracking its MAC state assuming that other stations are always active. Typically, approximations are made for the sake of tractability and are validated by discrete-event packet-level simulation. In this chapter, we first discuss simulation approaches for 802.11 modeling and then focus on analytical ones.

3.1 Simulators for 802.11

NS-2 [1] is a popular academic simulator for modeling computer networks. This simulator has a split programming language interface; the simulation scenarios are described in a scripting language while the runtime engine is implemented in C++. The 802.11 PHY layer is implemented in detail with carrier

sensing by both physical and virtual mechanisms. As explained in Chapter 2, the physical carrier sensing is done by the PHY layer in every slot; and the virtual carrier sensing is done by the MAC layer RTS/CTS control frames. The 802.11 MAC layer follows the protocol specification for most part, and schedules events in all stations that are part of the WLAN. Because each event is simulated at all stations of the WLAN, the simulation complexity grows almost linearly with the number of stations and the transmission bit-rate.

Qualnet [2] is a commercial simulator from Scalable Networks, Inc. A simulation scenario is described by a flat text file, and the runtime engine is implemented in C. Several portions of the simulator are available as a binary-only release. Consequently, we are unable to describe exactly how the PHY and MAC layer are implemented. However, the general design conforms to a packet-level simulation with events triggered in all stations of a WLAN corresponding to the transmission of a single station. Therefore, we believe it, too, suffers from the same scaling limitation with increasing transmission bit-rate and number of stations.

A timestepped fluid simulator for 802.11 has been described in reference [30]. This work obtains the *average* goodput in terms of the number of active stations, and replaces the entire PHY/MAC layer in ns-2 by computing the average goodput of the WLAN in a timestep. Because this simulator operates in a timestepped manner, it is much faster than the native ns-2, and this has been reported in reference [30] in detail. However, this simulator suffers from the limitations of using the expectation of a random variable to approximate the random variable. In fact, it approximates an entire random vector (of goodputs of all stations) by the per-component average. As can be seen later in Chapter 12, there is correlation among the goodputs of stations across stations as well as time. Thus

this approach does not quite capture the dynamics of 802.11 performance. For instance, consider a scenario for the TCP transfer time over an 802.11 link. In such a scenario, as explained in Chapter 1, a fluid based approach cannot track the stochastic nature of instantaneous metrics. In fact, the metrics considered in reference [29] are the the normalized (relative to the transmission bitrate) average goodput and the total number of packets, both over an entire simulation run.

The NS-2 simulator has been used in several studies for studying DCF. Reference [67] studies TCP over 802.11 by a simulation using the NS-2 simulator, and provides heuristics to improve the capacity attained by the protocol by bunching together the TCP ACK transmission corresponding to a TCP DATA segment. In other words, when a TCP DATA segment is transmitted the channel is reserved if the recipient needs to send a TCP ACK back. The study shows that this heuristic improves both the throughput and fairness of TCP over 802.11.

Reference [40] studies various enhancements proposed to DCF to provide QoS by augmenting the NS-2 simulator. The metrics considered are throughput, utilization, collision probability, and delay. The main contribution is the non-saturated heterogeneous workload considered, which is difficult to examine using analysis. The reference reports that the enhanced DCF suffers from high collision rates and starvation of lower-priority traffic.

Reference [65] studies 802.11 delay and uses that to build a distributed control algorithm for service differentiation in general radio control. The key idea is to use the channel-sensed metrics like delay and loss, and estimate the level of contention more accurately than the adaptation of the 802.11 MAC. This information is conveyed to applications which can tune their parameters. Again, this is implemented in the NS-2 platform and studied.

Several MAC level packet-level simulation studies using custom simulators have also been used to quantify 802.11 performance under several environments such as channel conditions, load, and protocol parameters.

Reference [32] studies 802.11 DCF by simulation under the workload of VoIP connections. In general, the 802.11 capacity (and hence delay) is severely limited by the small packet sizes used in comparison to the per-packet overhead. However, small packet sizes also mean lesser probability of error in packet reception. Using packet-level simulation, this reference studies this trade-off by selecting the packet size appropriately depending on the delay requirement and the channel condition.

Reference [72] studies 802.11 DCF in the unsaturated case and demonstrates that the maximum throughput cannot be obtained in the saturated case. Further, it identifies this optimal operating point through packet-level simulations and suggests that admission control be employed to operate the protocol around this regime.

3.2 Analysis of 802.11

Several analytical models [8, 20, 42, 22, 23, 31, 29, 3, 57, 69, 58, 59, 51, 5] have been proposed for the evaluation of 802.11 performance in the last few years. Many of these are paired with custom packet-level 802.11 DCF simulators to validate their approximations and optimization heuristics.

The protocol is typically analyzed under saturation conditions, i.e., the stations are always active. 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 assume that the conditional collision probability, i.e., the probability that a packet

encounters a collision given it is transmitted, is constant.

Reference [20] approximates the 802.11 protocol by a persistent CSMA/CA protocol. Recall that $B_i(t)$ is the backoff counter of station i at t. Let B be distributed according to the stationary distribution of $B_i(t)$ (assuming that the number of active stations is fixed). In the model proposed in [20], every station transmits with probability $1/(E[B]+1)$ in every idle (802.11) slot *independent* of other stations and its own previous attempts. The analysis proceeds on two observations: 1) From the attempt probability in terms of $E[B]$, the collision probability can be obtained. 2) From the collision probability, $E[B]$ can be obtained. The functional equations corresponding to these two observations are solved to obtain the collision probability and the goodput. The results predicted by the model are the goodput and collision probability; they are confirmed by comparison with packet level simulation.

Reference [13] extends the work in reference [20] to dynamically tune the optimal operating point of the protocol according to the number of stations. The key observation is that when there are two few number of stations, the contention windows are higher than optimal. When there are too many stations, the contention windows are lower than optimal. So the idea is to guess the number of active stations and tune the contention windows appropriately to make the protocol operate near optimal performance.

Reference [8], perhaps the most cited technique for 802.11 performance modeling, makes the approximation that the evolution of one station can be decoupled from the rest of the active stations except for a constant collision probability encountered by that station in each attempt. Under this decoupling approximation, the stochastic process of the MAC state of a tagged station, i.e., the tuple

of the contention window $C_i(t)$ and the backoff counter $B_i(t)$, is modeled as a discrete time markov chain. Each virtual "slot" in this discrete time markov chain is either an 802.11 idle slot of the varying idle interval, a collision period, or a success period. This markov chain is solved to obtain the attempt probability (after an 802.11 varying idle slot) and collision probability as a functions of each other. Once these two functional equations are solved numerically, the goodput is obtained as the ratio of the expected payload in a virtual (markov chain) slot divided by the expected duration of a virtual (markov chain) slot.

Reference [29] 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 [58] obtains the distribution of the inter-arrival time between two frames of a tagged station in an 802.11 system; it does not focus on the goodput. The main result is that the inter-arrival time distribution is typically multimodal. The reason for this multimodal behavior is that other stations typically get to transmit between successive transmissions of a station. Because packet transmission times are significantly higher than the varying idle intervals, the inter-arrival time has peaks corresponding to the transmission times of other stations. An equation from reference [69] is used to relate the probability of collision to the number of active stations.

Reference [3] presents a highly simplified yet accurate model for 802.11 performance. This models the per-station backoff process as a semi-markovian process

with the states of the chain corresponding to the varying contention windows of the station. From the renewal reward theorem [56, 39], the average attempt-rate and hence the per-station collision probability are readily obtained. Because of the use of the renewal reward theorem, this work cleanly sidesteps the need to solve the underlying markov chain consisting of the contention window and the remaining backoff counter as in reference [8]. We also note that reference [69] provides an analysis that is very similar to that in reference [3].

Reference [15], perhaps the first work in modeling the performance of 802.11, takes into account the physical layer, but gives only a lower bound on the throughput, concluding that an exact analysis is impossible.

Analysis of 802.11 DCF has also led to proposals for optimizing the protocol (goodput) depending on the functional dependencies between protocol parameters and average goodput. Because the doubling of the contention window is limited by a maximum contention window, the protocol degrades in performance with increasing load in terms of the number of active stations due to increasing collision probability (the order of growth is analyzed in the appendix). Hence some form of adaptation of the backoff better than doubling the contention window and finite retries has been proposed in various works. For instance, references [29, 27, 12] identify the optimal backoff interval from the number of active stations; the number of active stations is guessed from the varying idle interval. More refined approaches to the adaptive estimation of the number of stations are given in reference [63], which uses a Bayesian estimator, and reference [9], which uses a Kalman filter to do the same. Adaptively modifying the contention window with a TCP like additive increase multiplicative decrease for the attempt rate has been proposed in [28].

Another form of admission control has been to impose a hierarchy on the number of competing stations. The WLAN is divided into a number of groups. Each group has a number of competing terminals that use basic DCF, while the groups themselves can be scheduled like time division multiple access. The optimality of time division over random access schemes under high load has been studied as early as in reference [61]. In the 802.11 context, references [60, 6] suggest the grouping idea. References [7, 42] provide for admission control depending on the application service requirements.

Other optimizations based on analysis of 802.11 have also been proposed. Reference [16] proposes that stations choose a backoff counter for the next attempt early and announce it in the packet header; any other station sensing this would choose a new backoff counter that does not collide with this. Bursting of several small packets in one 802.11 DATA/ACK transaction has been proposed in reference [68] to avoid sending several small packets with per-packet overhead.

As mentioned before, existing methods focus on steady-state approximations and yield the average long-term goodput; they do not obtain distributions of the goodputs. In this dissertation, we follow the analysis in reference [3] to obtain the per-station collision probability as a function of the number of active stations. As explained in Chapter 5 in detail, we extend this work to obtain *distributions and sample paths* of the system state and performance metrics.

Chapter 4

TSS for WLANs

In this chapter, we present an overview of TSS for WLANs. First, we state the assumptions we make in developing the transient analysis of 802.11 DCF. Then, we explain how TSS for WLANs computes per-station metrics accounting for correlations across stations and time.

4.1 Modeling assumptions

Notation used in TSS is shown in Table 4.1. We assume the following within a given timestep:

- The number of stations with packets to transmit is constant and denoted by M; the set of stations is denoted $\mathbf{M}$.
- 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 Chapter 5).
- The transmission interval of a collision is the same as that of a successful packet transmission.

Symbol	Stands for	Type within timestep
δ	timestep of TSS	constant
$\mathbf{M}$	set of active stations	constant
M	# of active stations, i.e., $\lvert\mathbf{M}\rvert$	constant
For one tagged packet:		
K	number of transmission attempts	random variable
Y_j	backoff duration for j th attempt	random variable
X	total backoff duration; equals $Y_1 + Y_2 + \cdots + Y_K$	random variable
For per-station attempt process:		
p	per-station collision probability	constant
N_i	goodput of station i	random variable
For aggregate attempt process:		
p_A	aggregate collision probability	constant
I	generic idle interval	random variable
η	equals $E[I]/(E[I]+\tau)$	constant
$\mathbf{N}$	vector of N_i for all i	random variable
N_A	aggregate goodput, $\sum_i N_i$	random variable
L	number of aggregate attempts for one aggregate success	random variable

Table 4.1: 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.

- 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 12.5 of Chapter 12 or a fixed point iteration as in reference [3]).
- 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:

- 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 4.1 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 variability from the average is negligible. That is

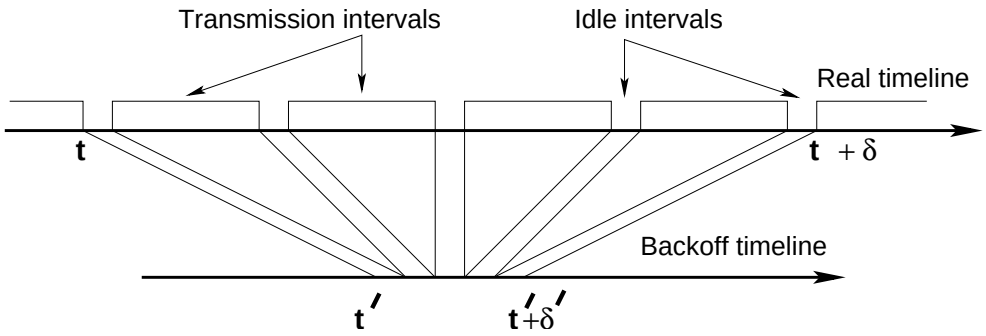


Figure 4.1: Real-to-backoff-timeline contraction approximation. An interval $[t', t' + \delta']$ in the backoff timeline corresponds to an interval $[t, t + \delta]$ in the real timeline, where $\delta' = \eta\delta$ and $\eta \triangleq E[I]/(E[I] + \tau)$.

- 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 5.4 of Chapter 5 justifies this in detail.)

4.2 Overview of TSS for WLANs

We now explain how TSS obtains sample path evolutions of the goodputs and the MAC states. Specifically, 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 6.3 of Chapter 6. So we need to probabilistically obtain $C_i(t+\delta), N_i(t)$ given $C_i(t)$ for all active 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 [3] (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. First, from the number of active stations, the distribution of the idle interval is obtained. Next, using the idle interval distribution, the first two moments of the time taken for one success in the aggregate attempt process are obtained (prior works focus on the mean alone). Finally, the normal distribution of the instantaneous aggregate goodput follows from the central limit theorem for renewal processes. (Analysis in Chapter 5.)

- **Step 3:** For each active station i, obtain $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$ by abstracting the interaction with the rest of the stations by an average per-station 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** (denoted by X) can "fit" within this interval. The total backoff duration 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). This analysis makes use of the real-to-backoff timeline contraction approximation. Unconditioning on t_s gives $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$. (Analysis in Chapter 6.)

  Thus we need to obtain the distribution of the the total backoff duration and its convolution. An efficient algorithm to obtain n-fold convolution of the distribution of total backoff duration is given in Chapter 7.

- **Step 3:** Dependently sample $N_i(t)$ from $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ for all active 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 $N_A(t)$. Specifically, the method considers stations according to a random permutation of their id's. Each station is allocated goodput from a specific part of its marginal distribution depending on the sum of all goodputs allocated prior to it and the aggregate goodput constraint. (Explained in Chapter 8.)

- **Step 4:** For each active 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 $[t, t+\delta]$. This gives the last successful transmission time instant within the timestep. From this, the new MAC state distribution is obtained from the observation that the attempt process started afresh at t_f and has not successfully transmitted till the end of the timestep. (Analysis in Chapter 9.)

Note that each of the steps computes some probability distribution. These probability distributions can be parametrized in terms of the number of active stations and the timestep duration. Therefore, they can be precomputed or cached, and thus are a one-time cost for all sample paths of the same simulation scenario. Further, these precomputed pdf's are stored as tables of the inverse of the corresponding cdf. Thus sampling from these precomputed distributions is equivalent to indexing into these tables, an $O(1)$ operation.

Chapter 5

Distribution of Aggregate Goodput

In this chapter, we obtain the first of the various pdf's involved in each timestep of the TSS simulator: the distribution of the instantaneous aggregate goodput $\mathsf{Pr}(\, N_A(t)\,)$. From the number of active stations, the distribution of an idle interval is obtained. Using the idle interval distribution, the first two moments of the time taken for one success in the aggregate attempt process are obtained, using which, the distribution of the instantaneous aggregate goodput is obtained.

Recall that the per-station collision probability p is available as a function of M. Given this relationship, we obtain the distribution of $N_A(t)$ in $[t, t+\delta]$ in terms of the number of active stations M, the transmission interval τ, and the timestep δ. In Sections 5.1 and 5.2, we leverage results from reference [3]. Section 5.1 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 5.2 analyzes 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.

Then, in Section 5.3, 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 [3] obtains the mean of $N_A(t)$ alone.) Section 5.4 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.

5.1 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 5.1 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 $Y_1 + Y_2 + \cdots + Y_K$.

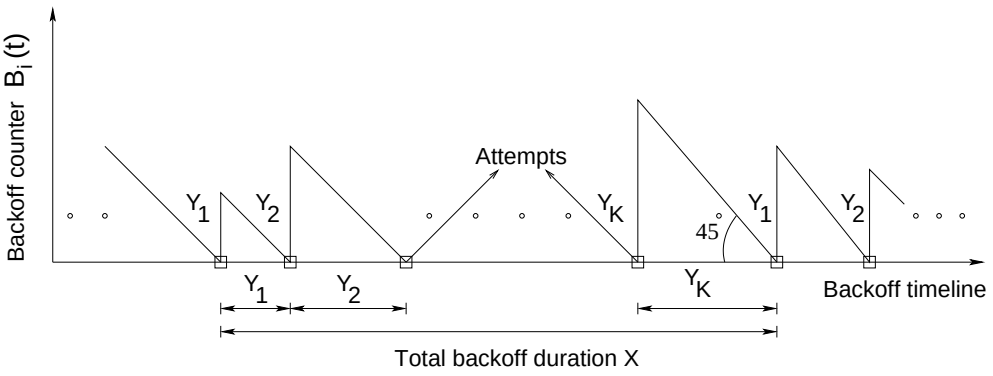


Figure 5.1: Per-station attempt process of station i in backoff timeline driven by backoff counter $B_i(t)$. Attempts are made when $B_i(t)$ hits zero, and $B_i(t)$ is renewed according to the backoff process. The angle 45° indicates that $B_i(t)$ decreases with slope -1 everywhere except at the attempt points.

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 [56], 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 $E[Y_i] = \gamma 2^{i-2} - 1/2$. 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

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

5.2 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.

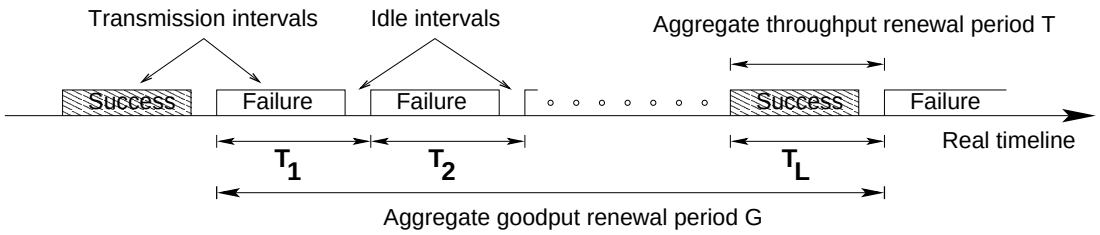


Figure 5.2: Aggregate attempt process in the real timeline. Aggregate goodput renewal period G is the time between two successful transmissions.

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 $p_A \approx p/2$.]

5.3 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 [56], 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 5.2 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., [3, 8, 29]) 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 $Var[L] = p_A/(1-p_A)^2$. Now we have the first two moments of the renewal period of a renewal process. Again, by a straightforward application of the central limit theorem for renewal processes, we have

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

5.4 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 Chapters 12 and 13, this approximation does not significantly compromise the accuracy.

Chapter 6

Conditional Distribution of Per-station Goodput

In the previous chapter, we obtained the distribution of the instantaneous aggregate goodput by considering the aggregate attempt process. In this chapter, we obtain the conditional distribution of the instantaneous per-station goodput $N_i(t)$ of a tagged station i conditioned on its MAC state at the beginning of the timestep. Specifically, we condition on the contention window $C_i(t)$ and approximate the backoff counter $B_i(t)$ in terms of $C_i(t)$.

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 the distribution of $N_i(t)$ conditioned on $C_i(t)$

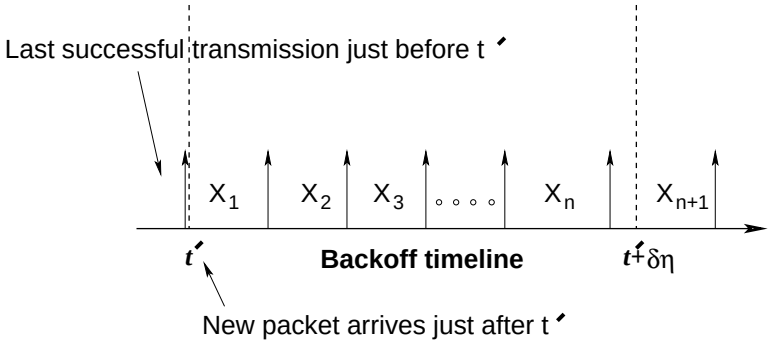


Figure 6.1: Successful transmissions of a tagged station in the interval $[t', t' + \delta\eta]$ in the backoff timeline. The timestep $[t, t + \delta]$ in the real timeline is contracted to $[t', t' + \eta\delta]$ in the backoff timeline.

alone.

6.1 Obtaining Pr($N_i(t)|C_i(t) = 0, B_i(t) = 0$)

We now analyze the case $\langle C_i(t) = 0, B_i(t) = 0\rangle$, i.e., station i transmitted successfully just before t and gets a new packet for to transmit just after t. Figure 6.1 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 $h(n, \eta\delta)$, we have

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

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 Chapter 7), the pdf $h(n, \eta\delta)$ can be obtained as above. Note that $h(n, \eta\delta)$ depends solely on δ and η. In turn, η can be determined in terms of the number of active stations and hence this pdf can be parametrized in terms of the number of active stations and the timestep duration.

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

We now obtain the distribution of $N_i(t)$ given an arbitrary starting state $B_i(t), C_i(t)$. Figure 6.2 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

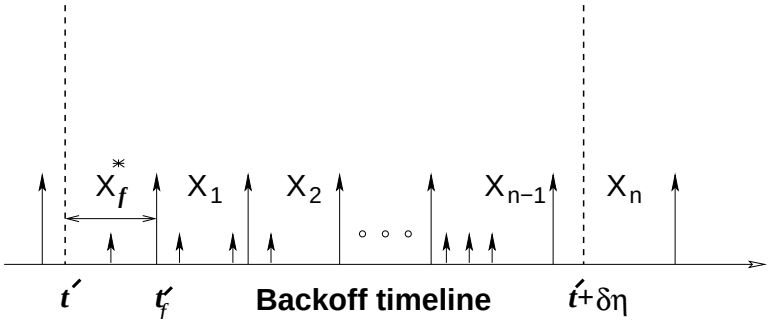


Figure 6.2: Transmissions of a tagged station in the backoff timeline interval $[t', t' + \eta\delta]$ corresponding to real timeline interval $[t, t + \delta]$ when $\langle B_i(t), C_i(t)\rangle \neq \langle 0, 0\rangle$. A shorter arrow indicates a failure, a longer arrow success. The first successful transmission occurs at t'_f and X^*_f is the backoff duration $t'_f - t'$.

$h(n-1, \eta\delta - X^*_f)$. So we want to obtain the pdf of backoff time to first success X^*_f.

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,

$$
\begin{aligned}
&Pr(X_f^* = l | B_i(t) = b, C_i(t) = \gamma 2^{c-1}) \\
&= \sum_{i=1}^{i=\beta-c} Pr(b + Y_c + \cdots + Y_{c+i} = l) Pr(K = i) \qquad (6.1) \\
&Pr(N_i(t) = n | X_f^*) = h(n-1, \eta\delta - X_f^*) \qquad (6.2)
\end{aligned}
$$

Using Equations 6.1 and 6.2, we have

$$
\begin{aligned}
&Pr(N_i(t) = n | B_i(t) = b, C_i(t) = \gamma 2^{c-1}) \\
&= \sum_{l=0}^{l=\eta\delta} Pr(X_f^* = l | B_i = b, C_i = \gamma 2^{c-1}) \quad \times \quad h(n-1, \eta\delta - l) \qquad (6.3)
\end{aligned}
$$

6.3 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/residual time) of the distribution $Uniform[0..C_i(t)-1]$. For a random variable $U \sim Uniform[0..a]$, the forward recurrence time [56, 39] 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 $Pr(N_i(t)|C_i(t))$.

6.4 Short-term unfairness in 802.11

Short-term unfairness in 802.11 has been the subject of much research [24, 25, 64, 71]. Reference [71] examines short-term unfairness for hidden terminals while references [25, 24] 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 [52, 54]. 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 Chapter 12 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 6.1 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,

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 6.1: 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.

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).

Chapter 7

Convolution of Total Backoff Duration Distribution

To evaluate the pdf obtained in the previous chapter, 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. We first obtain 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.

7.1 Distribution of total backoff duration

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 Chapter 5, 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

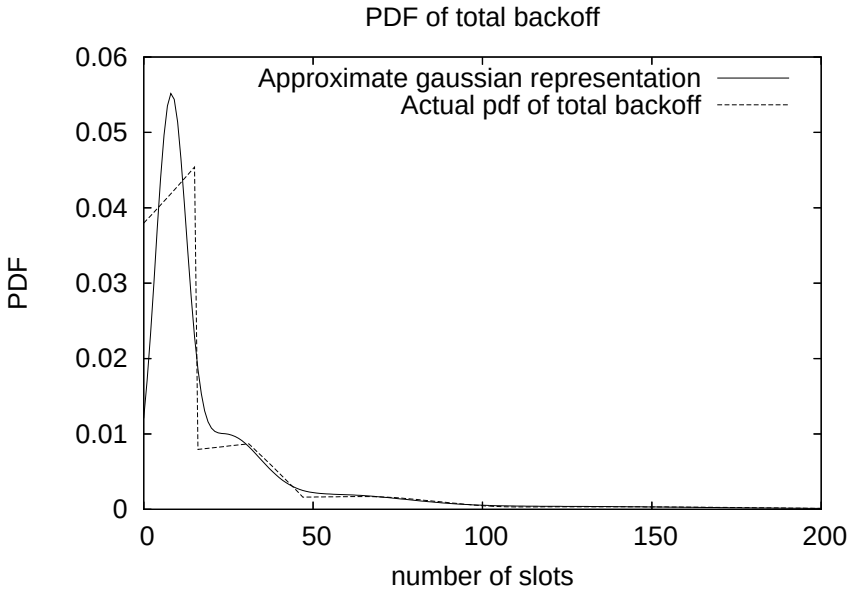


Figure 7.1: Illustrating the accuracy of the weighted gaussian approximation to the pdf of the total backoff duration in a packet's lifetime X.

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 7.1 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 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.

7.2 Impracticality of a normal approximation

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 7.6. 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 [44, 45, 10, 55, 53].) 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 $N_A(t)$.

7.3 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 $\sqrt{s_1^2 + s_2^2}$ [4]o. 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. Like-

wise, 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 10 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 (line 9), *new-list* is shrunk by calling procedure SHRINK-LIST with *new-list* as parameter and *curr-list* is updated from the return value. Procedure SHRINK-LIST first sorts *list* in lexicographically increasing order according to the tuple (m_k, s_k) in line 9.

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 7.
- 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

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 *curr-list* $\leftarrow$ SHRINK-LIST(*new-list*)

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 9 through 11.

- 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 13 and 14.

SHRINK-LIST(*list*)

1 *shrunk-list* $\leftarrow \{\}$

▷ Input *list* has tuples of form

▷ $\langle weight, (mean,\ dev)\rangle$

2 Sort *list* according to increasing $(mean, dev)$

3 $\langle w_0, (m_0, s_0)\rangle \leftarrow$ first(*list*)

4 $w \leftarrow 0$, $m \leftarrow m_0$, $s \leftarrow s_0$

5 **for** each successive $\langle w_k, (m_k, s_k)\rangle$ in *list*

▷ ϵ is a threshold

6 **if** $(w_k < \epsilon)$

▷ Ignore gaussian of very low weight

7 $w \leftarrow w + w_k$

8 **elseif** $|m_k - m| \leq \theta m$ and $|s_k - s| \leq \theta s$

▷ Combine two "close" gaussians

▷ Closeness parameter is θ

9 $p_1 \leftarrow w/(w_k + w)$; $p_2 \leftarrow w_k/(w_k + w)$

10 $m \leftarrow p_1 m + p_2 m_k$; $s \leftarrow \sqrt{p_1 s^2 + p_2 s_k^2 + p_1 p_2 (m - m_k)^2}$

11 $w \leftarrow w + w_k$

12 **else** ▷ This entry cannot be combined anymore

13 Add $\langle w, (m, s)\rangle$ to *shrunk-list*

14 $w \leftarrow w_k$, $m \leftarrow m_k$, $s \leftarrow s_k$

15 Add $\langle w, (m, s)\rangle$ to *shrunk-list*

16 **return** *shrunk-list*

7.4 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 $O(na\beta^2 \log \beta)$. 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 [34]. If $\beta^2 \log \beta$ is $O(1)$ w.r.t. input size $O(na)$, then our approach reduces $O(na \log na)$ to $O(na)$.

7.5 Optimization

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

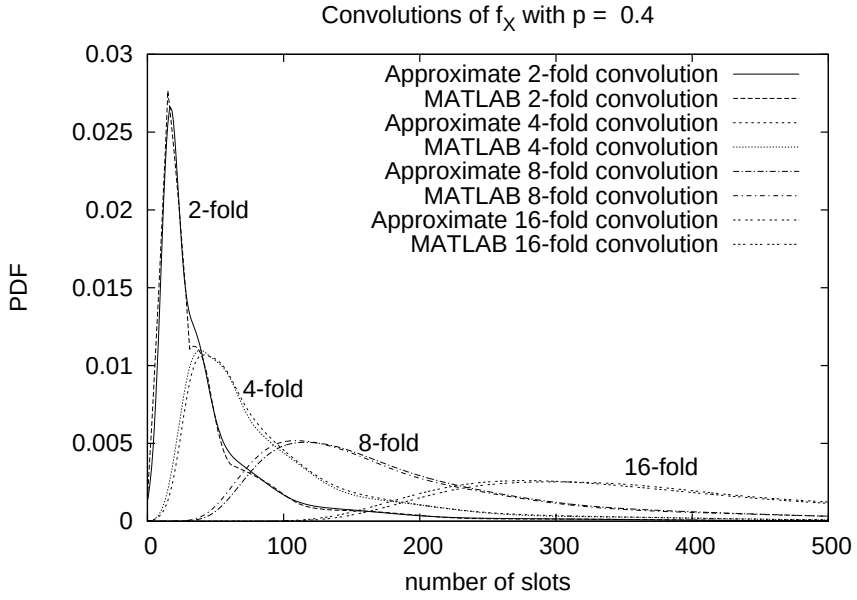
pdf's for timesteps of length δ.

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.

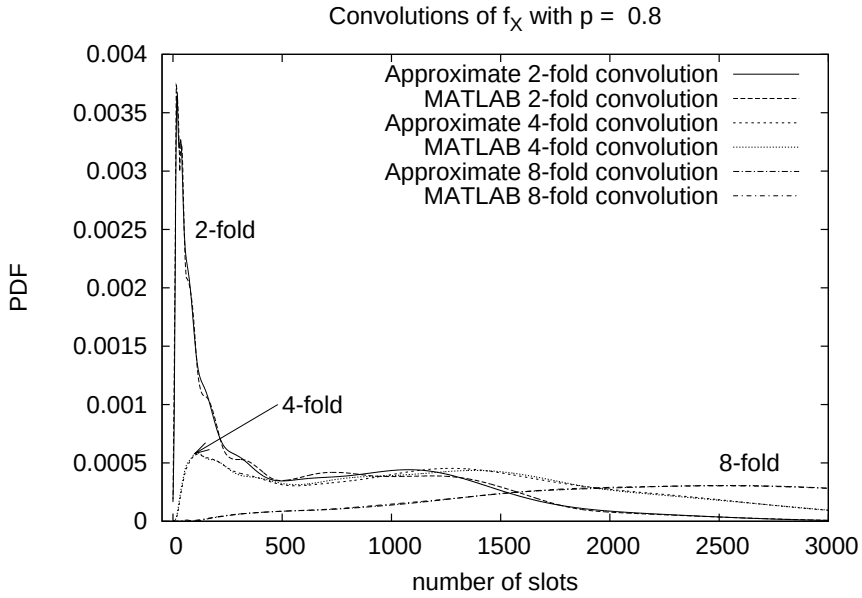
7.6 Validation and speedup

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 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 7.2(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 7.2(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



(a)



(b)

Figure 7.2: 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.

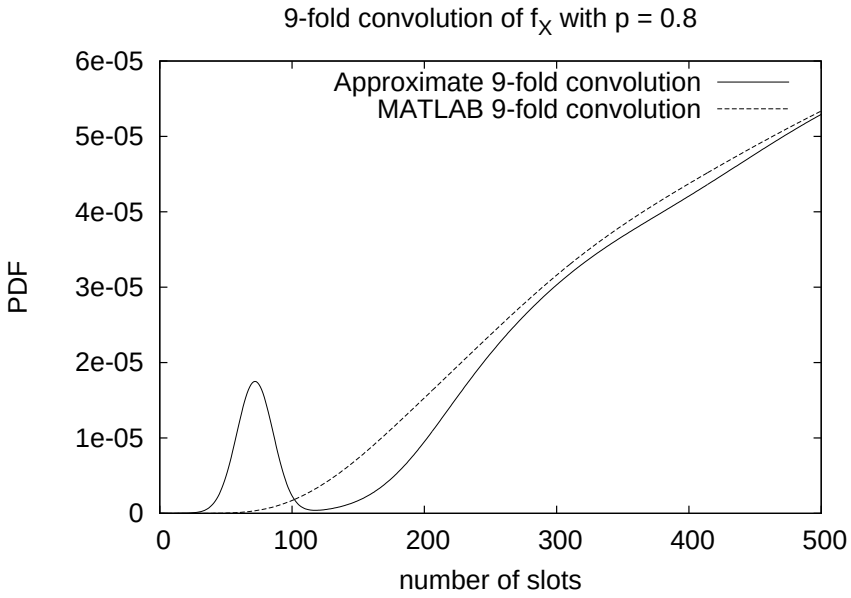


Figure 7.3: Comparison of 9-fold convolution of f_X for $p = 0.8$ as obtained from our convolution algorithm with that obtained by MATLAB.

multimodal and eventually smoothen out for higher convolutions. For instance, in Figure 7.3, 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.

Chapter 8

Dependent Sampling of Per-station Goodputs

The goodput sample $N_i(t)$ 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 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 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. Note that the marginal goodput distribution does indicate that if the number of active stations goes up, the per-station collision probability goes up and hence the range of values of $N_i(t)$ goes down. However, the extent of correlation will not be captured adequately by this abstraction of other stations, because we are abstracting the random variables by some form of

average behavior. Thus we want a method that will sample $N_i(t)$'s from their conditional distributions in a manner that reflects their negative correlation accurately. In this chapter, we present a randomized algorithm that dependently samples the conditional distributions. Specifically, stations are ordered according to a random permutation, and a station's marginal distribution is sampled according to the sum of the samples of all goodputs allocated prior to it.

8.1 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.

8.2 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 indices of stations that are in the set of active stations $\mathbf{M}$. 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 increases the negative correlation between stations whose indices are considered last (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.

8.3 Algorithm for sampling per-station goodputs

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-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 $allotted, expected, upper\text{-}tolerance, lower\text{-}tolerance \leftarrow 0$

3 $\forall i \;\; N_i(t) \leftarrow 0$, $M \leftarrow$ number of active stations

4 $\pi \leftarrow$ random permutation of id's in $\mathbf{M}$

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** $allotted > expected + upper\text{-}tolerance$

9 $\quad\quad s \leftarrow$ sample lower tail of $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$

10 $\quad$ **elseif** $allotted < expected$ - $lower\text{-}tolerance$

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 $\quad upper\text{-}tolerance \leftarrow \theta_1 \times expected$

22 $\quad lower\text{-}tolerance \leftarrow \theta_2 \times expected$

23 **if** $(count = M)$

24 $\quad\quad N_{\pi(M)}(t) \leftarrow max(N_A(t) - allotted, 0)$

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.

8.4 Runtime

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 [18]. 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.

Chapter 9

Conditional Distribution of New MAC State

So far we obtained the marginal per-station goodput distributions $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ in Chapters 6 and 7, and presented a dependent sampling algorithm that uses these marginal distributions to obtain the per-station goodputs in Chapter 8. To complete the inductive step of TSS in each timestep, we need to update the MAC state at $t + \delta$, i.e., obtain $C_i(t + \delta)$.

In this chapter, we obtain the distribution $\mathsf{Pr}(\, C_i(t + \delta)\,)$ of the new MAC state 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.

9.1 Analysis with non-zero goodput

We first analyze the case $N_i(t) \neq 0$. Figure 9.1 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

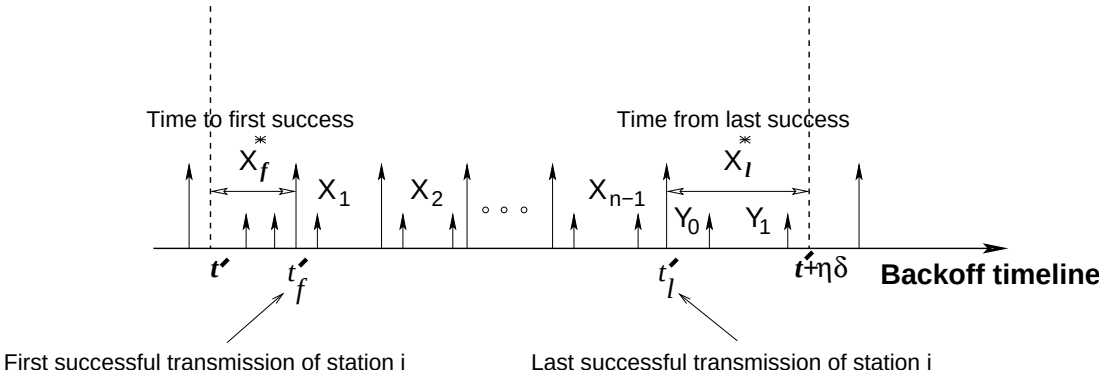


Figure 9.1: Transmissions of a tagged station in the backoff timeline interval $[t', t' + \eta\delta]$ corresponding to the real timeline interval $[t, t + \delta]$. A longer arrows indicates a successful transmission, a shorter arrow failure.

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 Chapter 6. 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:

$$Pr(N_i(t) = n|C_i(t)) = \sum_{r=0}^{r=\eta\delta} Pr(X_f^* = r|C_i(t)) \sum_{s=0}^{s=\eta\delta-r} f(r, s)$$

where

$$f(r, s) \triangleq Pr(X_1 + \cdots + X_{n-1} = s)Pr(X_n > \eta\delta–r–s)$$

By Bayes' theorem we have:

$$\begin{aligned} &Pr(X_l^* = s|N_i(t) = n, C_i(t)) \\ &= \frac{\sum_{r=0}^{r=\eta\delta-s} Pr(X_f^* = r|C_i(t))f(r, \eta\delta - r - s)}{Pr(N_i(t) = n|C_i(t))} \end{aligned}$$

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 $c-1$ 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 $\mathsf{Pr}(\, C_i(t+\delta) | C_i(t), N_i(t)\,)$.

9.2 Analysis with zero goodput

When $N_i(t) = 0$, as in Chapter 6, we approximate $B_i(t)$ as the forward recurrence time of $C_i(t)$. Note that because the goodput is zero, the transmission attempts, if any, are all unsuccessful. Specifically, if the station makes k unsuccessful attempts, then the time spent for backoff in the timestep for the first transmission is $B_i(t)$. Now let backoff times for each of the remaining $k-1$ attempts be $Y_{i_1}, Y_{i_2}, \cdots Y_{i_{k-1}}$. The new $C_i(t+\delta)$ corresponds to $C_i(t)$ and i_{k-1} unsuccessful attempts if $B_i(t) + Y_{i_1} + \cdots + Y_{i_{k-1}}$ just exceeds $\eta\delta$. Thus the probability distribution of the new MAC state can be obtained by convolving the distributions of the Y_i's and $B_i(t)$.

Chapter 10

The Timestepped Simulator

We describe how the analysis fits together as the TSS generates a sample path. The pseudo-code of the simulator is shown in Algorithm *TSS-WLAN*. 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 6 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 or looked up from M in line 7. 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 9.
- For each active station i, $\mathsf{Pr}(\, N_i(t)|C_i(t)\,)$ is obtained in 10.

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 14.

5 $\quad \mathbf{M} \leftarrow$ set of active stations

6 $\quad M \leftarrow |\mathbf{M}|$ (number of active stations)

7 $\quad$ compute/look up collision probability p for M stations

8 $\quad$ compute/look up $\mathsf{Pr}(\ N_A\ )$ using M, δ, p

9 $\quad N_A \leftarrow$ sample from $\mathsf{Pr}(\ N_A\ )$

10 $\quad$ **for** each station i in $\mathbf{M}$

11 $\quad\quad$ compute/look up $\mathsf{Pr}(\ N_i|C_i\ )$ using M, δ, p

12 $\quad$ SAMPLE-GOODPUTS($\mathbf{N}, \mathbf{M}$)

13 $\quad$ **for** each station i in $\mathbf{M}$

14 $\quad\quad$ compute/look up distribution of $\mathsf{Pr}(\ C_i(t + \delta)|N_i(t), C_i(t)\ )$ and sample

15 $\quad t \leftarrow t + \delta$

- Algorithm SAMPLE-GOODPUTS is used to sample the goodput of each station i in line 12.

- For each active station i, the distribution of $\mathsf{Pr}(\ C_i(t+\delta)\ )$ is obtained given $C_i(t)$ and $N_i(t)$ and sampled to obtain the new state. This is done in lines 13 through 14

Algorithm SAMPLE-GOODPUTS takes $O(M)$ time. All further random sampling for updating the MAC state can be done in $O(M)$ time. Hence 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 $O(M.\ \textit{sim-duration})$, which is **independent of the bit-rate.**

Chapter 11

Runtime Speedup

Our main results are broadly along two directions: quantifying speedup and validating accuracy. In this chapter, we quantify the speedup. Chapters 12 and 13 validate the accuracy of TSS against PLS.

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. For the runtime of PLS, in addition to the actual runtime of the code, we include the time to load the precomputed pdf's from disk as well as the amortized precomputation time.

11.1 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

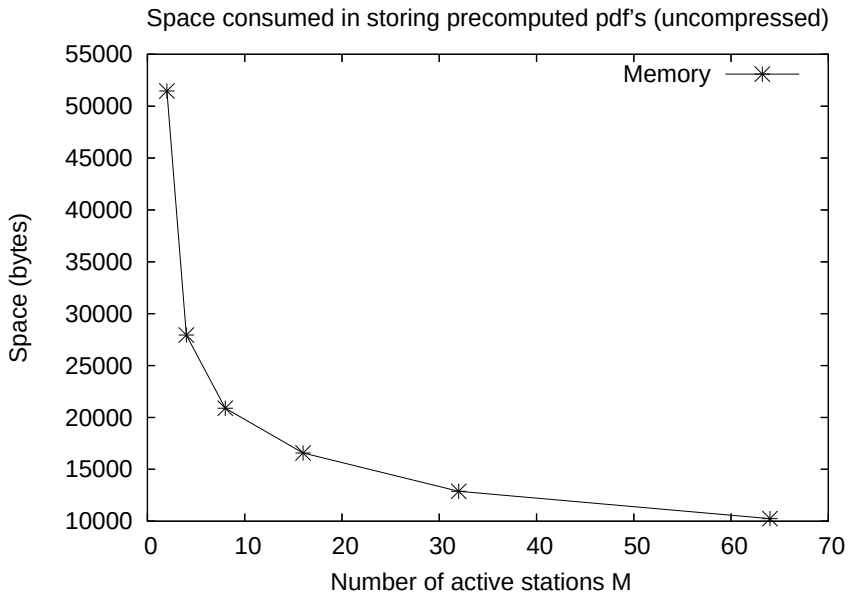
of two constant bit rate (CBR) flows sharing one 802.11 channel takes about 4.5 seconds in our custom simulator with logging enabled, while ns-2 takes about 70 seconds with all logging disabled.

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.

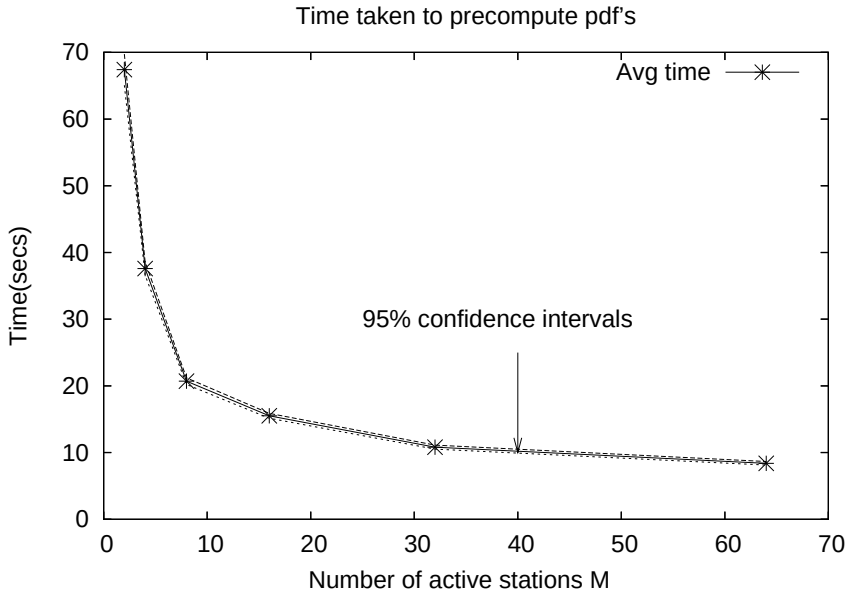
11.2 Precomputation costs in space and time

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).



(a)



(b)

Figure 11.1: 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 $[2, 4, 8, \cdots, 64]$.

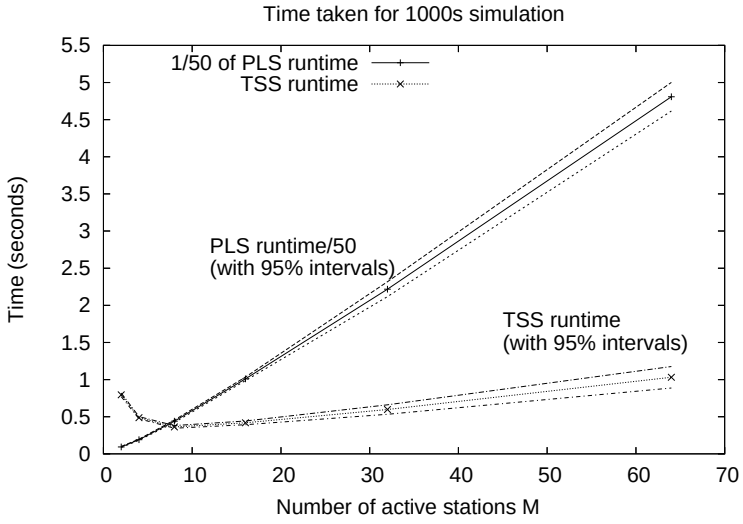
Figure 11.1(a) shows the space requirement for pdf's for $\delta = 50$ms with M varying in $[2, 4, 8, \cdots, 64]$. Because n_m decreases with increasing M, the space required decreases with increasing M. A similar trend can be seen in Figure 11.1(b), which shows the time taken to precompute the pdf's 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).

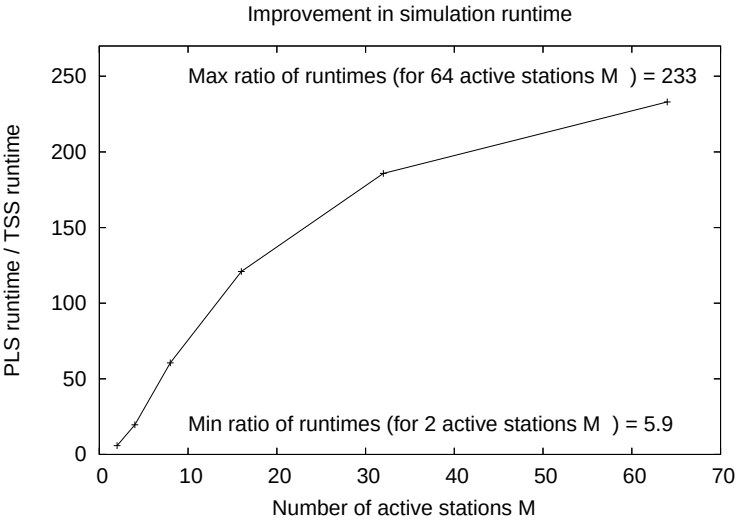
We note that n_m increases with increasing δ, so the table sizes increase with increasing δ. Even though the memory required is low, the nature of TSS allows us to trade off space with time as follows: instead of precomputing tables for large δ, precompute for, say, $\delta/2$ and perform computation 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, if not impossible, to achieve in PLS.

11.3 Runtime comparison

TSS for WLANs provides an improvement up to two orders of magnitude in the runtime over PLS. Figure 11.2(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 11.2(b) shows the ratio of the runtimes for PLS and TSS. Each point plotted in Figure 11.2(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



(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

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 11.1(b) and 11.1(a).

Chapter 12

Validation of TSS with Fixed Number of Active Stations

In this chapter, 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, a secondary result is presented in this chapter for a closed form approximation and analysis of the per-station collision probability as a function of

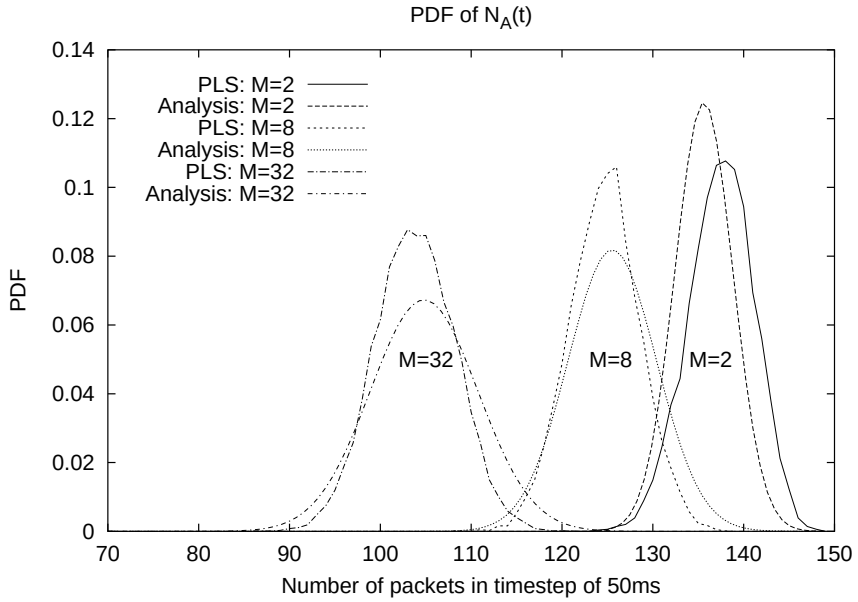


Figure 12.1: Comparison between empirically obtained pdf of $N_A(t)$ for $t = 5$s and $\delta = 50$ms for varying M. The deviations of $N_A(t)$ predicted by the analysis are overestimates for $M > 2$ while for $M = 2$, it is an underestimate.

M.

12.1 Aggregate goodput distribution

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 12.1.

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 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 12.2 shows the cross-correlation function between the sequences $\{I_i\}$ and $\{F_i\}$. The peak at lag 1 illustrates factor 1. Figure 12.3 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.

12.2 Conditional distributions of per-station goodput and MAC state

We now consider the distributions $\mathsf{Pr}(\ N_i(t)|C_i(t)\ )$ and $\mathsf{Pr}(\ C_i(t+\delta)|N_i(t), C_i(t)\ )$. 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 12.4(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

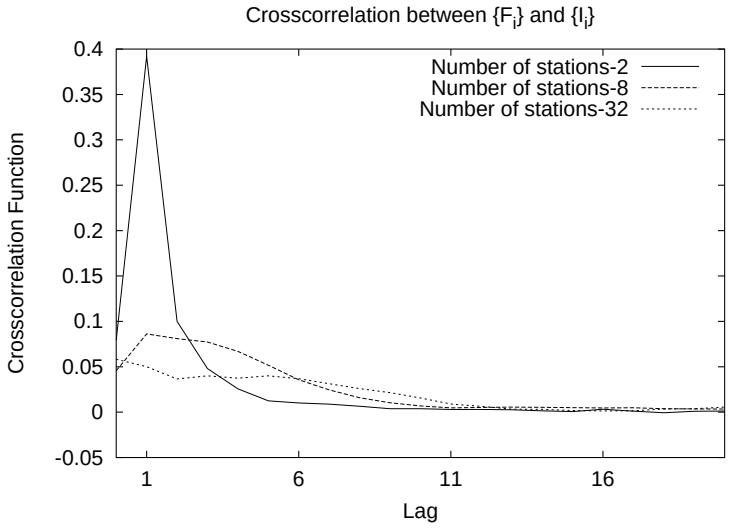


Figure 12.2: Crosscorrelation function between sequences $\{I_i\}$ and $\{F_i\}$ obtained over 1000000 samples for each M

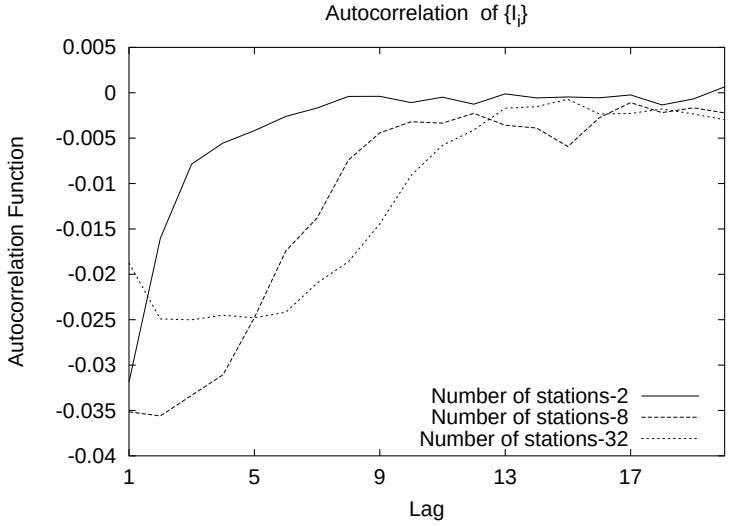
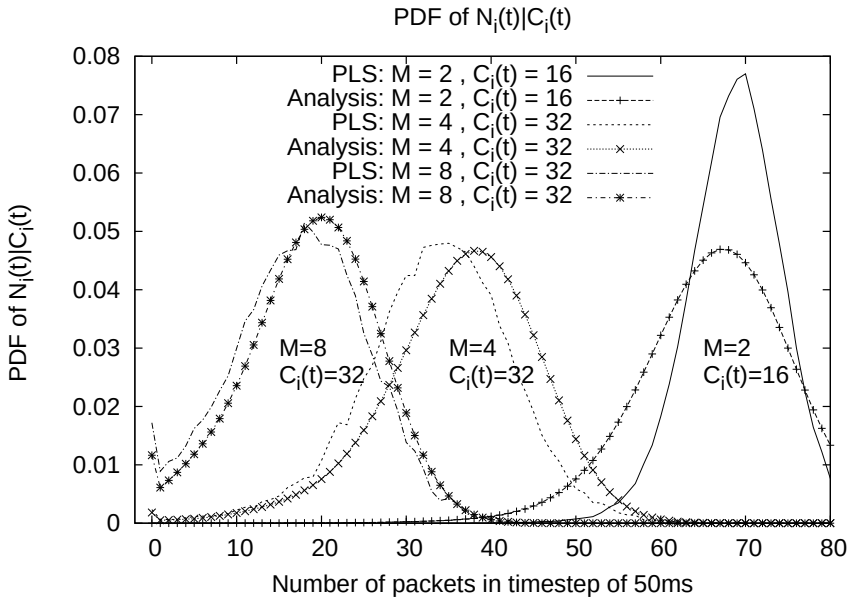
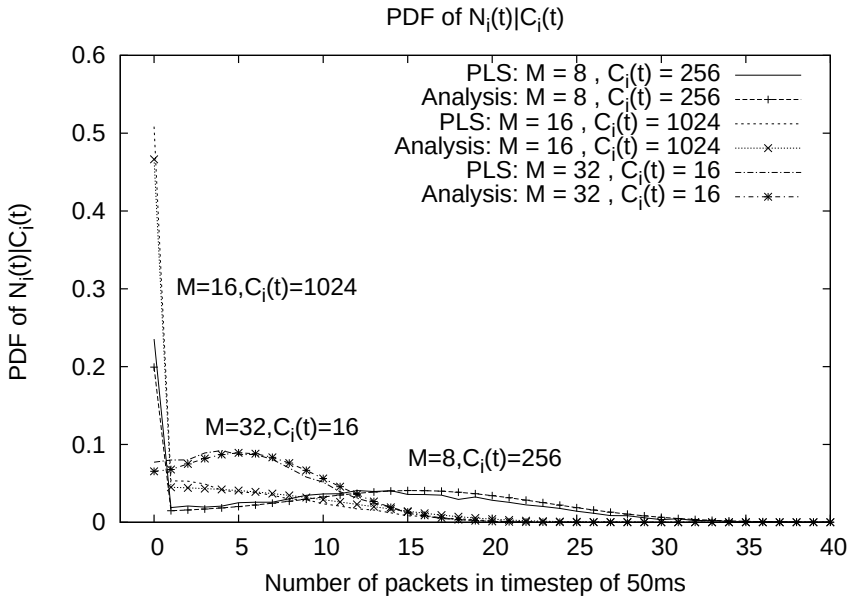


Figure 12.3: Autocorrelation function of I_i sequence obtained over obtained over 1000000 samples for each M. At lag 0, the function is exactly 1 and not shown in the figure.



(a)



(b)

Figure 12.4: PDF of $N_i(t)|C_i(t)$ for various values of $C_i(t)$ and varying M.

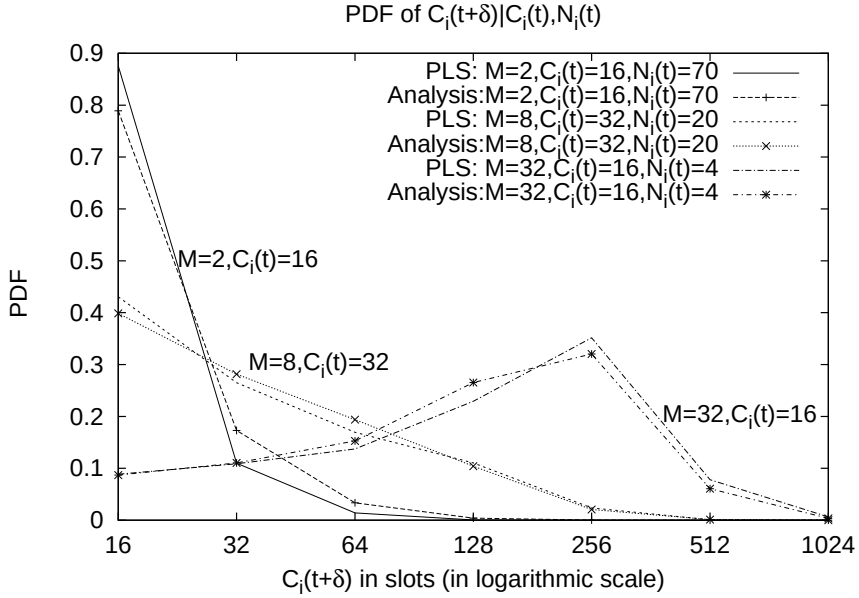
contention window size increase, the goodput starts deviating from normal significantly with an increased probability of zero instantaneous goodput. The analysis captures this trend as can be seen in Figure 12.4(b). We also note that $Pr(N_i(t) = 0|C_i(t) = 1024)$ for $M = 16$ is much higher (around 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 12.5(a) and 12.5(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 12.5(a) covers low values of $C_i(t)$ while Figure 12.5(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 Chapter 9.

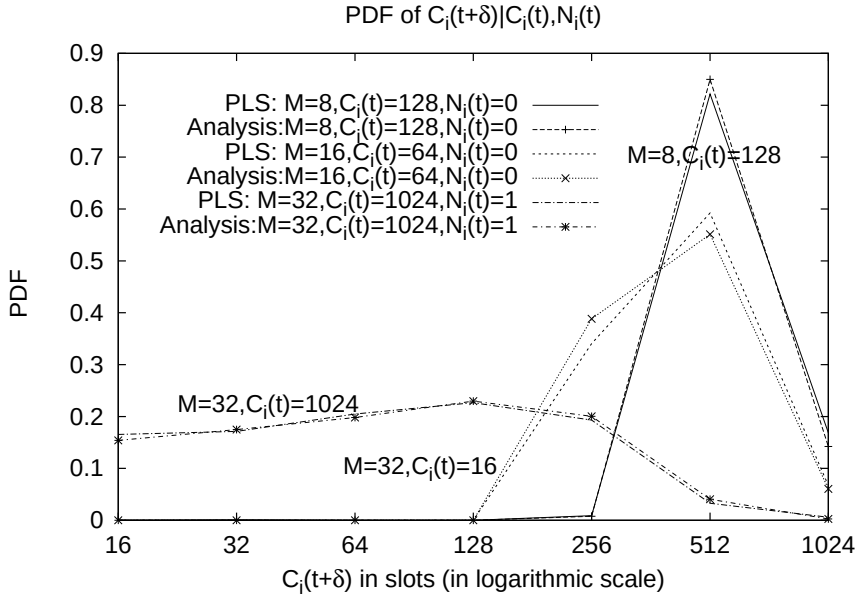
12.3 Unconditional distributions of per-station goodput and MAC state

We now compare the unconditional distributions of $N_i(t)$ and $C_i(t)$. As can be seen from Figures 12.6(b) and 12.6(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 12.7(a) and 12.7(b). Note that this distribution so obtained is an approximation of the frequency distribution of the time spent by the tagged

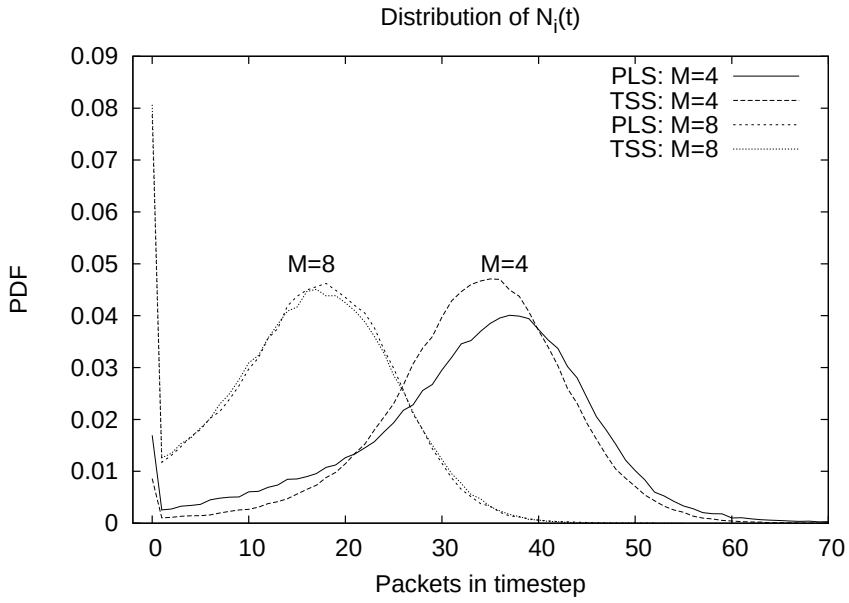


(a)

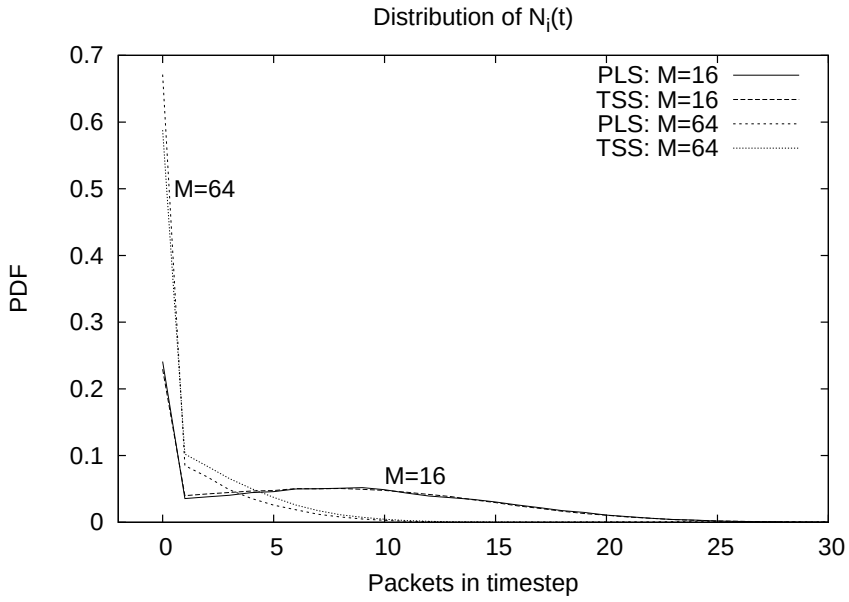


(b)

Figure 12.5: 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)

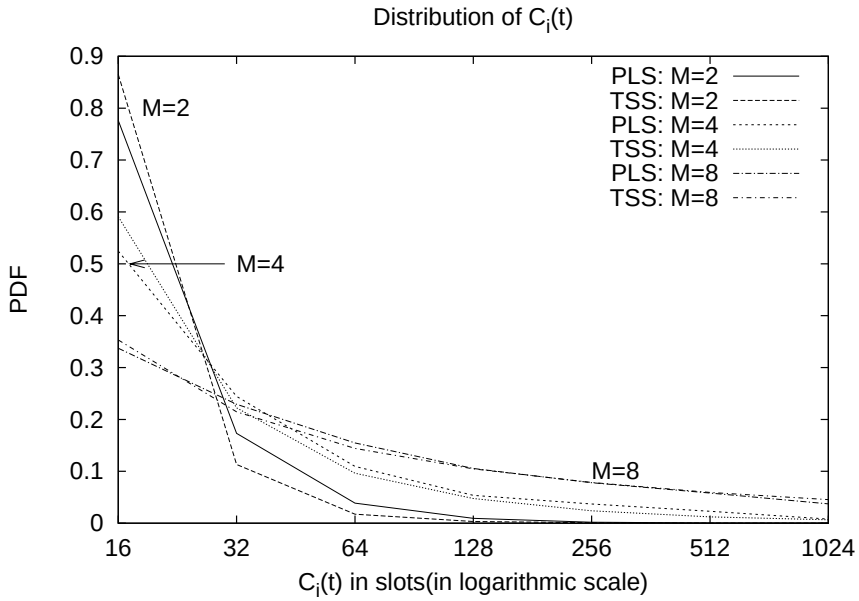
Figure 12.6: Distribution of unconditional $N_i(t)$

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 previous paragraph.

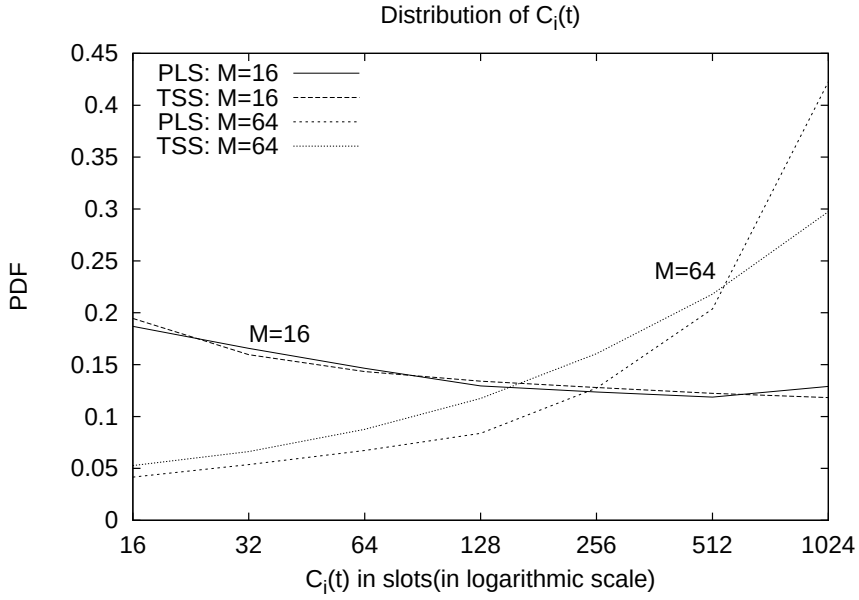
12.4 Comparing sample paths

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 12.8(a) and 12.8(b) show the autocorrelation function obtained from TSS and PLS. TSS tracks the correlation in N_i across time



(a)



(b)

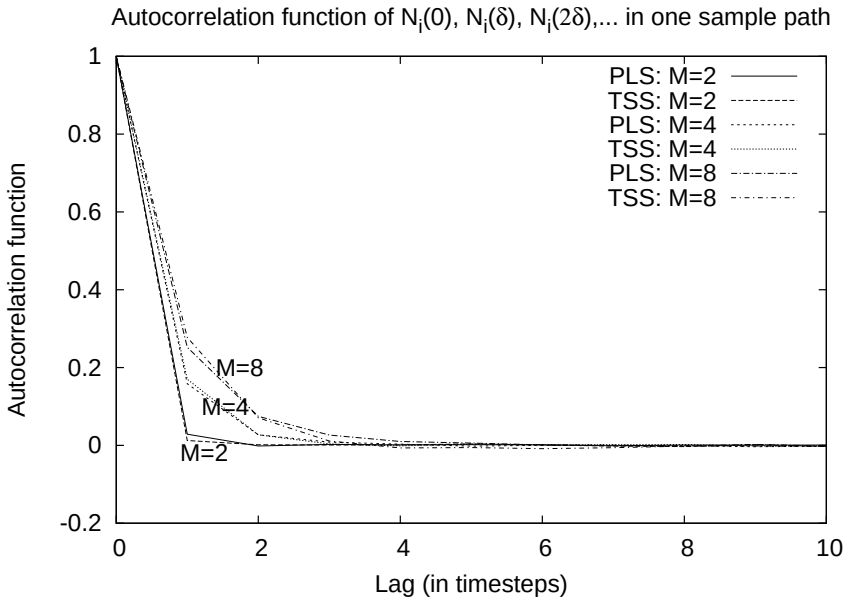
Figure 12.7: Distribution of $C_i(t)$ for varying values of M.

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 12.9(a) and 12.9(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 $C_i(t)$.

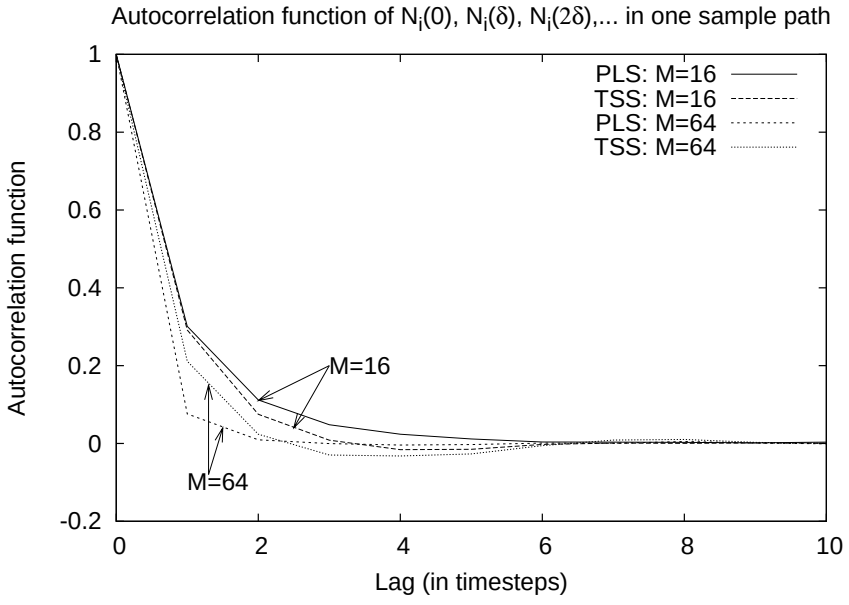
Next, we consider the crosscorrelation function computed between the goodput samples of two tagged stations i and j, i.e., between the timeseries $N_i(0), N_i(\delta), N_i(2\delta), \cdots$ and $N_j(0), N_j(\delta), N_j(2\delta), \cdots$. As can be seen in Figures 12.10(a) and 12.10(b), the method to 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 12.11(b) and 12.11(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.

12.5 Per-station collision probability

References [3, 8, 20, 29] 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 12.12 shows the curves

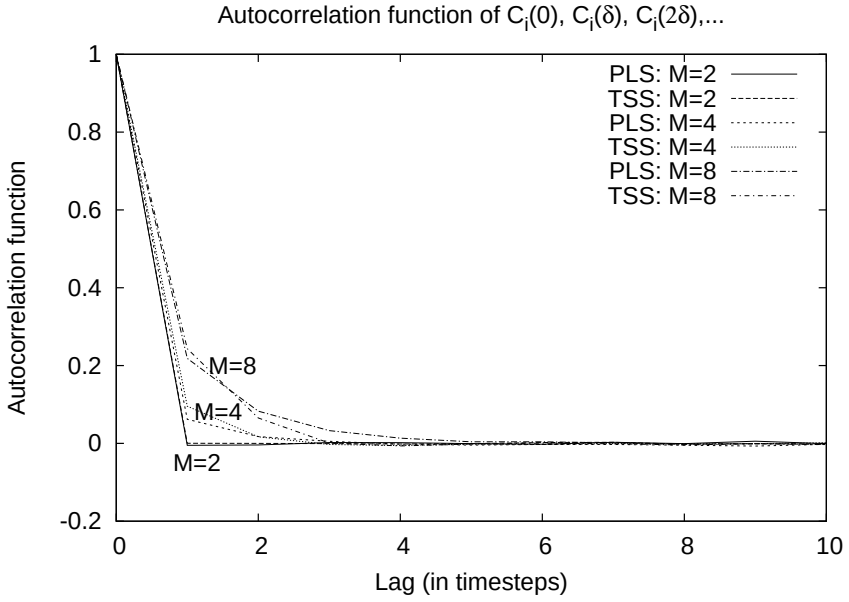


(a)

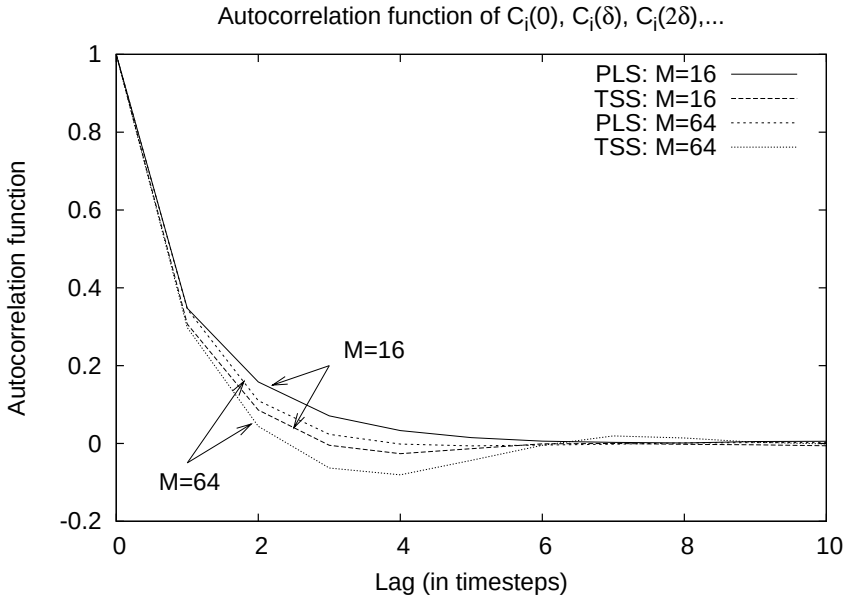


(b)

Figure 12.8: 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.

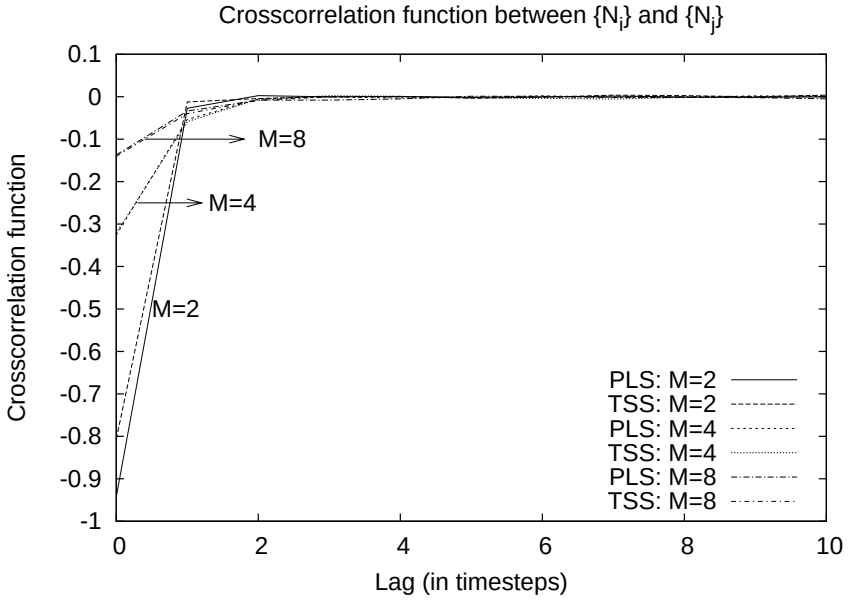


(a)

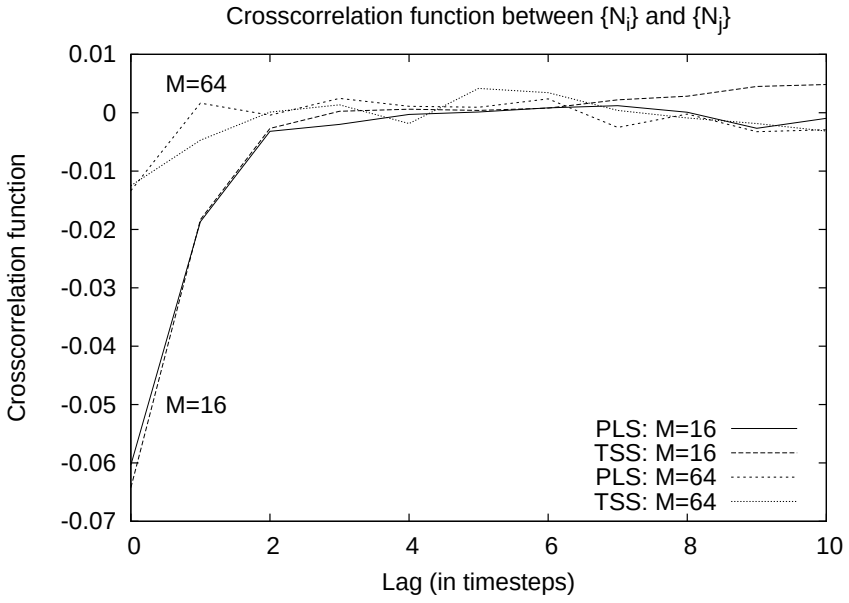


(b)

Figure 12.9: 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.

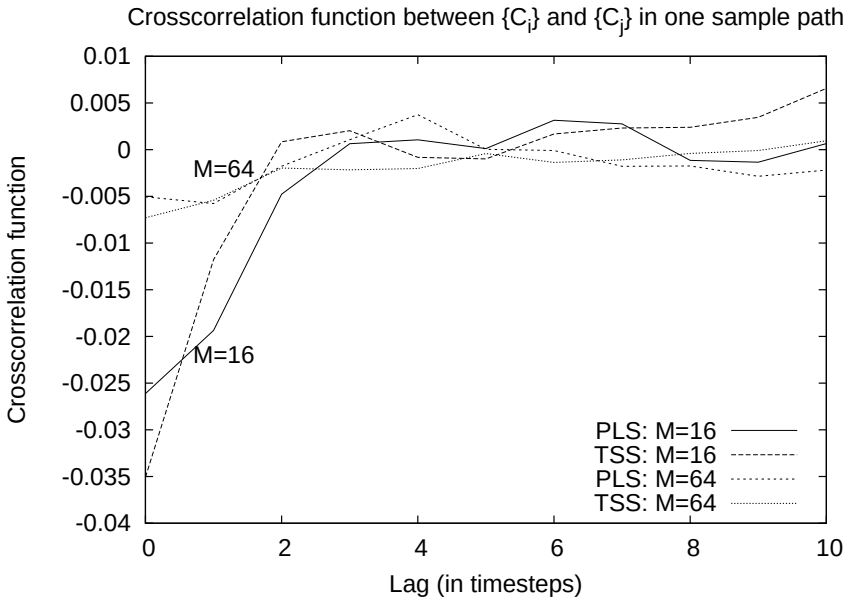


(a)

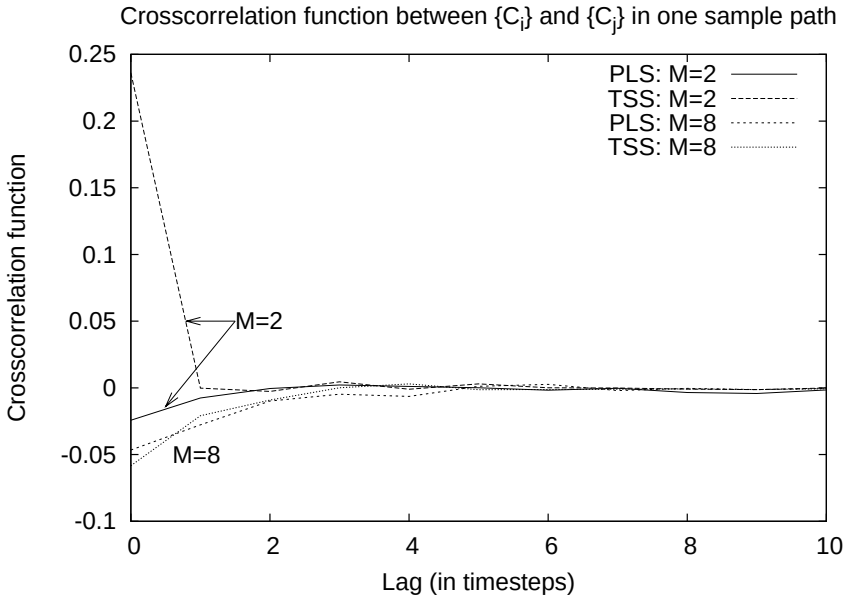


(b)

Figure 12.10: 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)

Figure 12.11: 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.

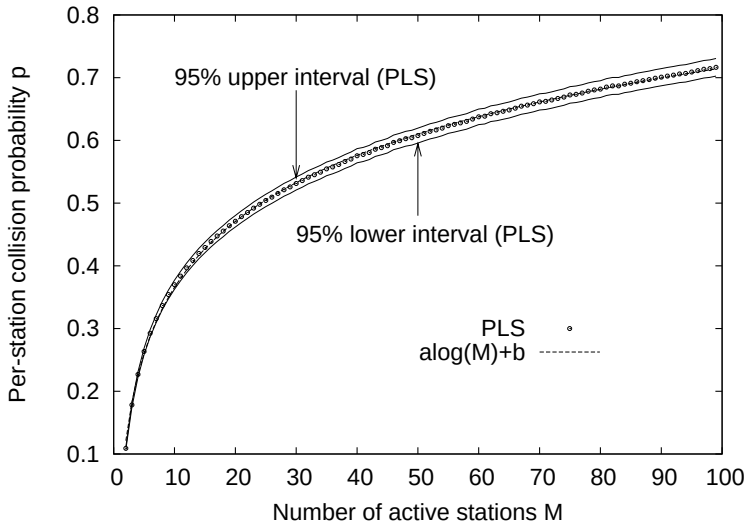


Figure 12.12: $p(M)$ from simulations and an analytical fit of $0.1519\log(M) + 0.0159$ for various values of M and $\beta = 7$. The 95% confidence interval of each simulation point is within 2% of the mean.

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 12.13 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 [3] 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

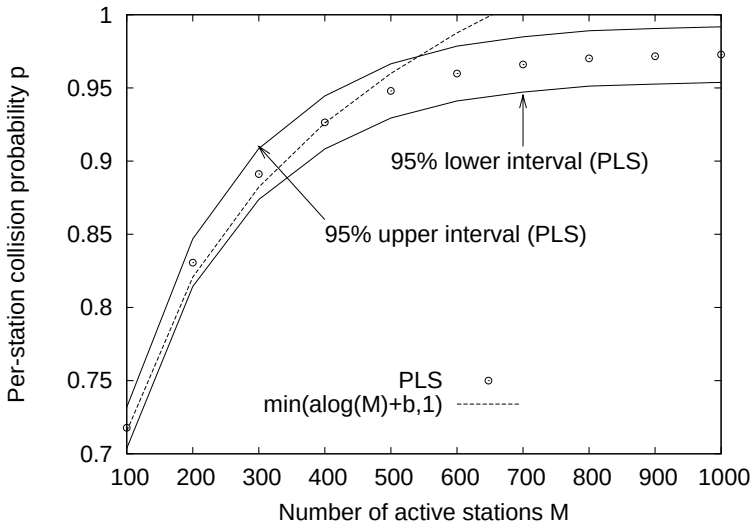


Figure 12.13: $p(M)$ from simulations and an analytical fit of $\min(0.1519 \log(M) + 0.0159, 1.0)$ for $M = \{100, 200, \ldots, 1000\}$ and $\beta = 7$. The two curves diverge after $M = 400$ stations at a per-station collision probability greater than 0.935.

collision probability with finite β. The logarithmic fit works because it fits the expression involving the Lambert function that occurs in the function $p(M)$.

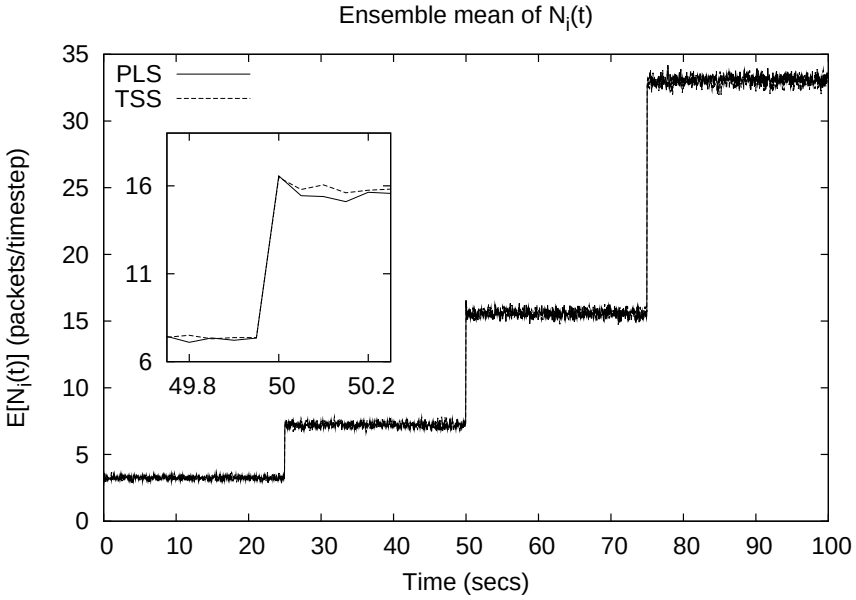
Chapter 13

Validation of TSS with Varying Number of Active Stations

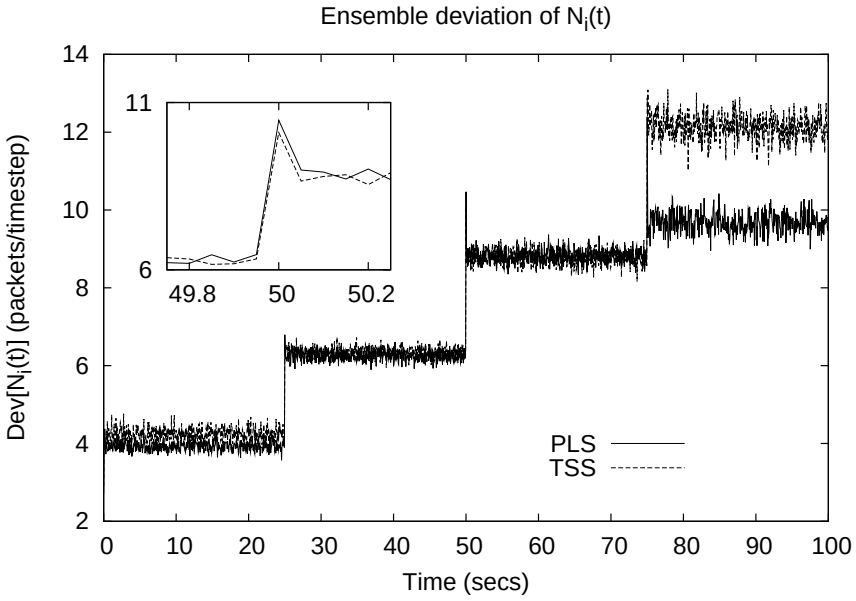
In the previous chapter, we evaluated the accuracy of TSS by comparison against PLS with the number of active stations being fixed throughout a simulation scenario. In this chapter, we evaluate the accuracy of TSS again by comparison against PLS with the number and the set of active stations varying across time. First, we present an example, where the variations are simple. Then, we present an example where the complex variations are randomly chosen.

13.1 Simple variations in set of active stations

We now consider an example where M deterministically 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. Note that a station that becomes inactive remains so throughout the simulation run. 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 13.1(a) and the evolution of



(a)



(b)

Figure 13.1: 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.

$Dev[N_i(t)]$ is shown in Figure 13.1(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.

13.2 Complex variations in set of active stations

We now evaluate the performance of TSS where a station is randomly active/inactive. Specifically, we consider a WLAN of 9 stations. Stations 1 through 8 are active for a random period chosen from $Uniform[0, 200]$ timesteps and are inactive for a random period chosen from $Uniform[0, 100]$; these periods alternate for the duration of the simulation. Station 9 is always active; this is done to ensure that there are at least two active stations at any point during the simulation run. An activity pattern is generated given this model for a a total duration of 20000 timesteps (corresponding to $1000s$). Given this activity pattern, 1000 runs of PLS and TSS are executed and performance metrics obtained.

13.3 Activity pattern

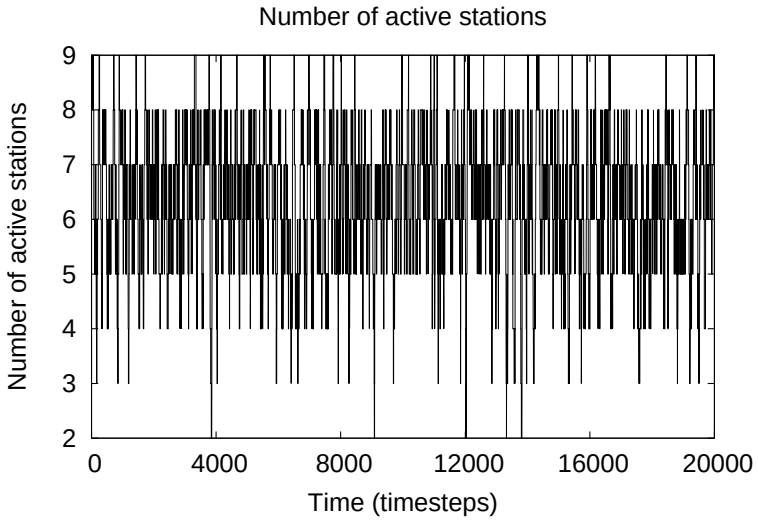
Figure 13.2(a) shows the number of active stations as a function of time for the entire 20000 timesteps, while Figure 13.2(b) shows the same for the first 2000 timesteps (for which ensemble curves are obtained later). In this activity pattern, the (sample path) average number of active stations in a timestep is 6.35 (with a deviation of 1.29), which can be explained as follows. Neglecting the initial transient (when all stations are active to start with), by the renewal-

rewards theorem [56, 39], the probability of a station being active in any timestep is 2/3 because of the expected active/inactive periods. For stations 1 through 8, by independence of the station's activities, the number of active stations is binomially distributed with parameters $B(8, 2/3)$. Because station 9 is always active, the expectation of the number of the total active stations is $5.33+1 = 6.33$ and its deviation is 1.33, which closely matches what is observed.

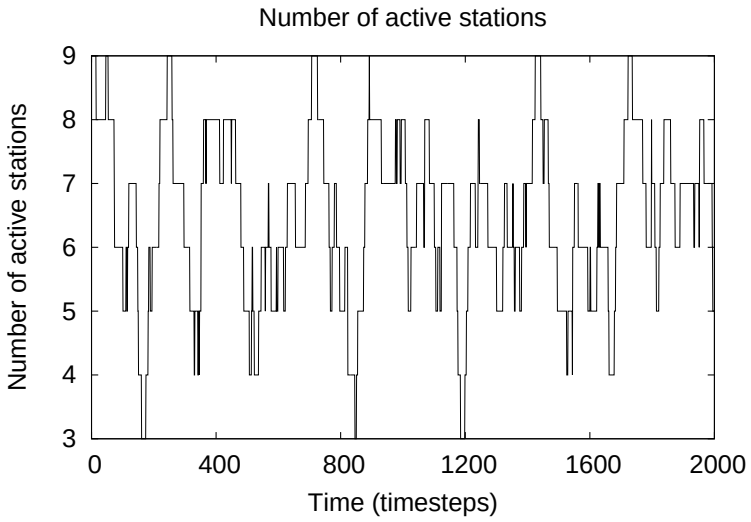
13.4 Sample path metrics - Aggregate

All sample path metrics are obtained over one long-run of 20000 timesteps For instance, the sample path averaged distribution of N_A is computed from samples $N_A(0), N_A(\delta), \cdots, N_A(19999)$ Unless otherwise, mentioned the tagged station is the one with id 1, i.e., it is a station that has varying activity.

First, we consider the aggregate goodput N_A. The sample path averaged distribution is shown in Figure 13.3(a) for both PLS and TSS. There is a very good match between TSS and PLS. The distribution of N_A appears gaussian even though it is sample path averaged and doesn't have a stationary number of active stations. This result can be explained as follows. For a given t, $N_A(t)$ is a gaussian distribution determined by the number of active stations. For a sample path, the distribution of N_A is a compound distribution. Specifically, it is a gaussian mixture distribution where the mixing parameters are determined by the time-varying number of active stations. The number of active stations is between 2 and 9 with the corresponding gaussian means of 137 packets/timestep through 122 packets/timestep and deviations of 3.62 packets/timestep and 3.98 packets/timestep. As we saw in Chapter 7, gaussian mixture distributions that are close in mean and deviation can be approximated well by a single gaussian



(a)



(b)

Figure 13.2: Number of active stations as a function of time in the random activity pattern.

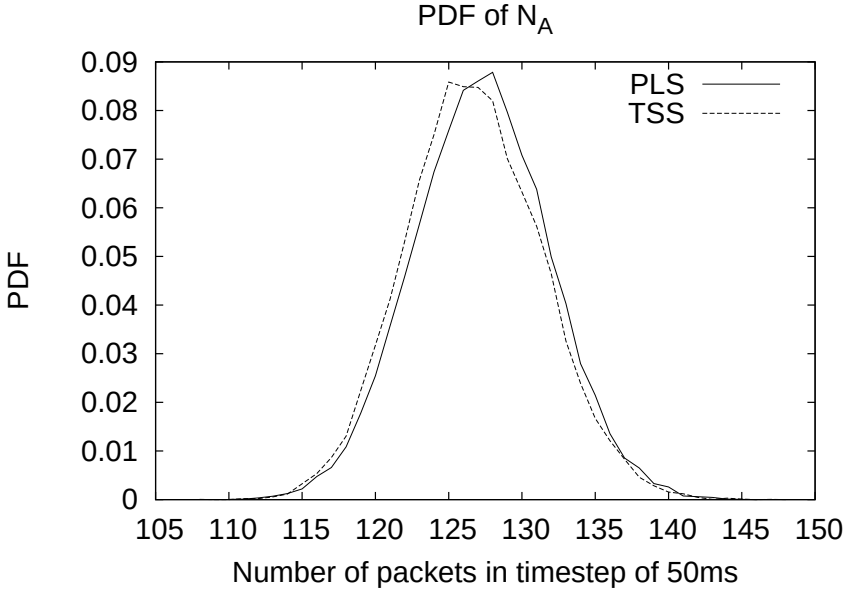
distribution with a suitable mean and deviation. Hence, the time-averaged aggregate goodput distribution is gaussian.

The autocorrelation function of $N_A(t)$ is shown in Figure 13.3(b). It exhibits a cyclic dependency which decays over time. This behavior can be explained as follows. The aggregate goodput in a timestep is essentially determined by the number of active stations. The number of active stations, in turn, is determined by the superposition of the activity patterns of all stations. The activity of each station is determined by a renewal process with a renewal period whose expectation is 300 timesteps (200 for active and 100 for inactive). Thus the aggregate goodput has a dependency which can extend to a lag of 300 timesteps, which seems to be observed in the autocorrelation of the aggregate goodput with the peaks being separated by 300 timesteps.

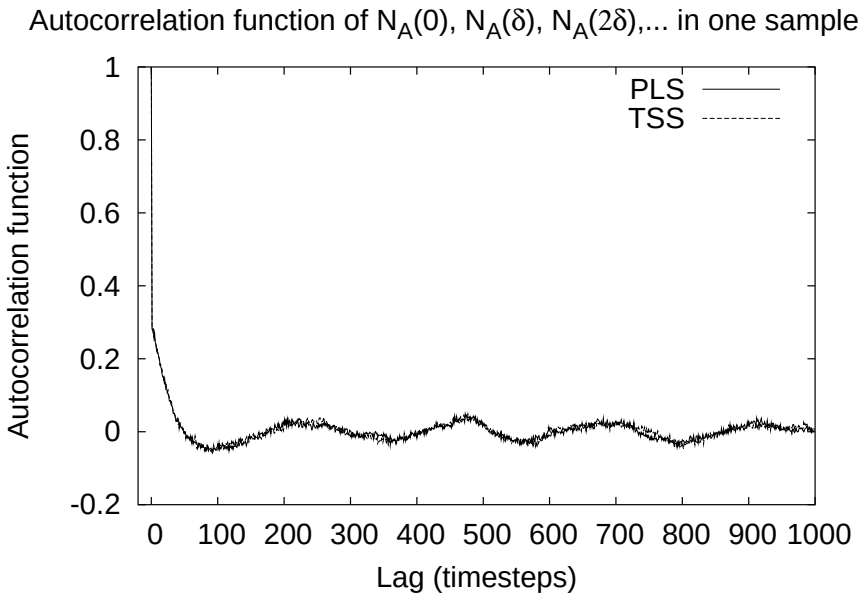
13.5 Sample path metrics - Per-station

We next consider the per-station goodput $N_i(t)$. Figure 13.4 shows the sample path averaged distribution of the per-station goodput N_i for $i = 1$. This distribution is obtained from samples through one run, i.e., it is a time-averaged distribution. The match between PLS and TSS for the distribution of N_i is excellent. The probability of zero goodput is about 0.37, which can be explained as follows. The station is inactive for about 1/3 the time, and 0.04 comes from contending with "5.3" active stations on average (as explained in Section 13.3).

Figure 13.5 shows the autocorrelation function of a tagged station's (station 1) goodput in one sample path. The correlation are determined by two factors: one, the semi-markovian nature of the arrivals, and two, the MAC level interactions. Specifically, the cyclic nature of the crosscorrelation is due to the alternation of



(a)



(b)

Figure 13.3: Time-averaged distribution of $N_A(t)$ and its autocorrelation function computed over one long run of 20000 timesteps.

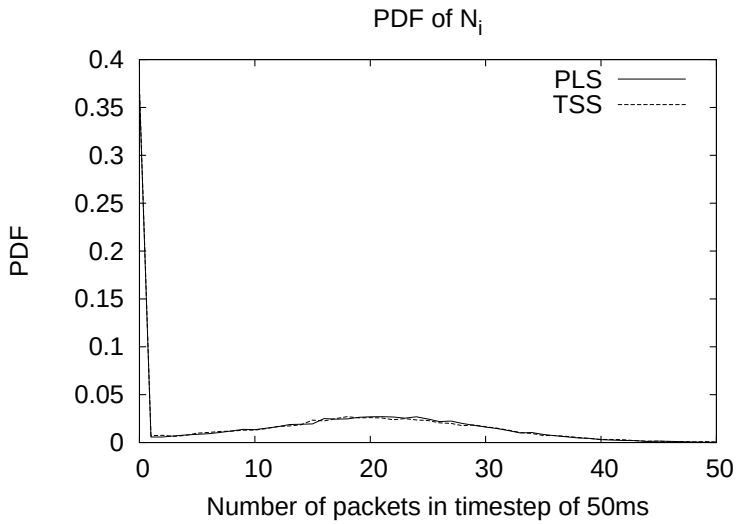


Figure 13.4: Sample path averaged PDF of the instantaneous per-station goodput $N_i(t)$ for $i = 1$ computed over 20000 samples from one run.

the active/inactive periods. As can be seen in the figure, the numbers predicted by TSS are in close agreement to those of PLS.

Figure 13.6 shows the crosscorrelation between two tagged stations, in this case, stations 1 and 2; this, too, is obtained from one sample path. Again, the crosscorrelation is determined by two factors: one, the semi-markovian nature of the arrivals, and two, the MAC level negative correlations. At lag 0, the negative correlation is pronounced, i.e., if two stations are active in the same timestep, then their goodputs are negatively correlated as expected. At low lags, the numbers predicted are visually indistinguishable between PLS and TSS (e.g, at lag 0, TSS predicts -0.1612 while PLS predicts -0.1603), while at higher lags, the numbers are in close agreement with each other.

Figure 13.7 shows the sample path averaged distribution of the contention window C_i for $i = 1$. The curves do not match exactly. Specifically, TSS overestimates $Pr(C_i(t) = 16)$. The reason for this behavior is as follows. When a

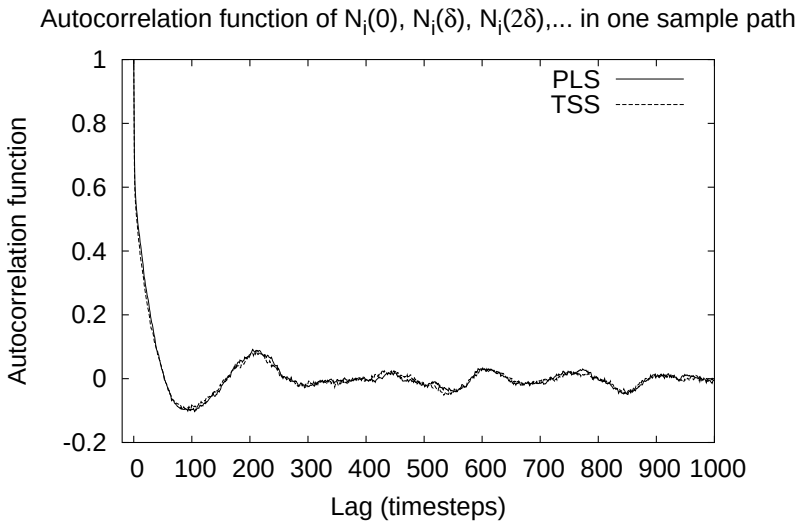


Figure 13.5: Autocorrelation function obtained from samples $N_i(0), N_i(\delta), N_i(2\delta), \cdots$ of one sample path for $i = 1$.

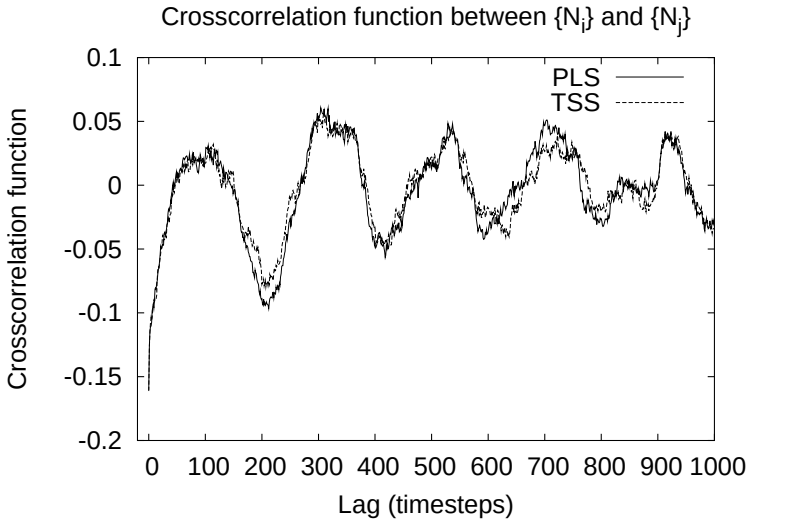


Figure 13.6: Crosscorrelation function obtained from samples $N_i(0), N_i(\delta), N_i(2\delta), \cdots$ and $N_j(0), N_j(\delta), \cdots$ of one sample path for $i = 1, j = 2$.

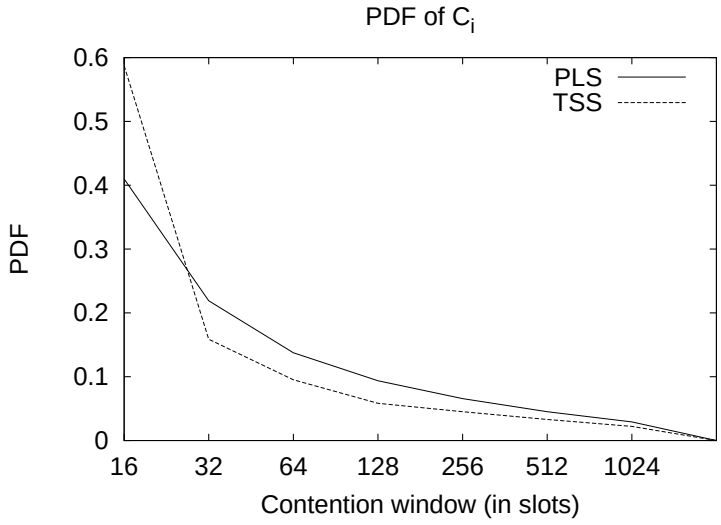


Figure 13.7: Sample path averaged PDF of the contention window C_i for $i = 1$ computed over 20000 samples from one run.

station i becomes inactive at the boundary of a timestep, TSS resets the contention window $C_i(t)$ of the station to be 16 immediately at the boundary of the timestep. However, PLS does not updates the $C_i(t)$ even if it became inactive till the current ongoing transmission either results in a success or is aborted well into the next timestep. Because the distribution $C_i(t)$ is obtained only from samples at the boundaries of timesteps (and not over all time), the TSS distribution is skewed from the actual computed by PLS. This can be confirmed by examining the ensemble-averaged curves for the distribution of $C_i(t)$ for $t = 20s$ and $t = 40s$ in Figures 13.18 and 13.19 respectively on page 111. There is no change in activity at these two time instants for station i. Consequently, there is a very close match between PLS and TSS.

Figure 13.8 show the crosscorrelation of the contention windows of two tagged stations C_i and C_j. Figure 13.9 and autocorrelation of the contention window Both figures have TSS values that do not completely match with those of PLS.

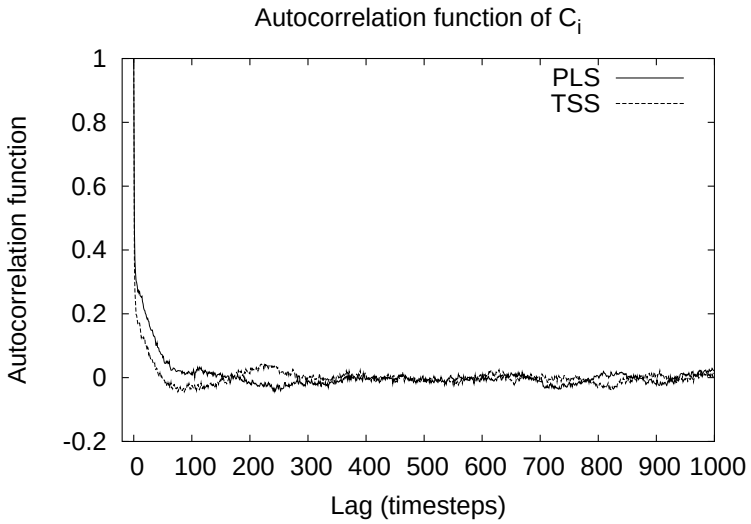


Figure 13.8: Autocorrelation function obtained from samples $C_i(0), C_i(\delta), C_i(2\delta), \cdots$ of one sample path for stations with random activity.

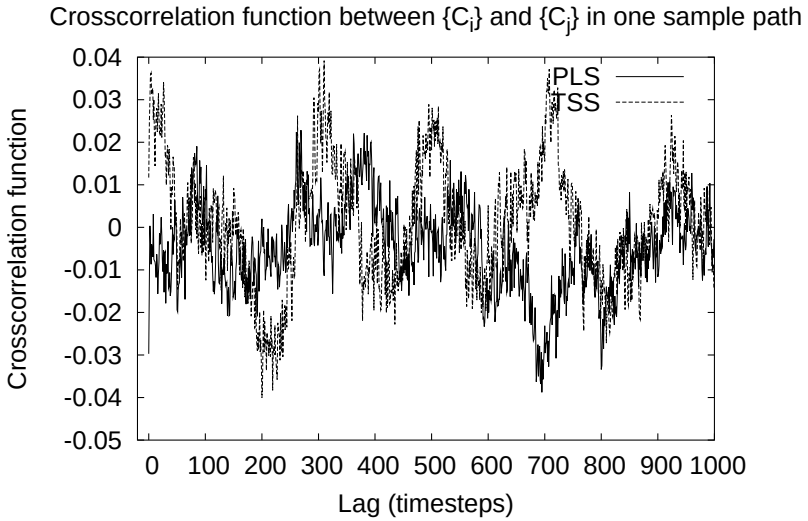


Figure 13.9: Autocorrelation and crosscorrelation function obtained from samples $C_i(0), C_i(\delta), C_i(2\delta), \cdots$ and $C_j(0), C_j(1), \cdots$ of one sample path for stations with random activity.

The reason, as explained before, is that TSS overestimates $Pr(C_i(t) = 16)$. However, the trend in PLS is captured by TSS for the autocorrelation function, while the crosscorrelation function seems to indicate that there is not much dependence between the $C_i(t)$ and $C_j(t)$ timeseries.

13.6 Ensemble metrics - Aggregate

We now consider the ensemble metrics for $t = 20s$ and $40s$. The ensemble means, deviations, and distributions were obtained from 1000 samples corresponding to 1000 different simulation runs by both TSS and PLS for $i = 1$. At $t = 20s$ and $t = 40s$, there were 6 and 5 active stations respectively.

Figures 13.10 and 13.11 show the ensemble mean and deviation of the instantaneous aggregate goodput $N_A(t)$ as a function of time for the first 100s or 2000 timesteps. The curve obtained for TSS is consistently off from that of PLS by an insignificant value; we believe this is due to minor inaccuracies from the analytical approximations. Note that this curve is almost a (scaled and inverted) mirror-image of the number of active stations that is shown in Figure 13.2(b). The ensemble deviation computed by TSS also matches that of PLS reasonably well. The seemingly significant difference is due to the small scale of the Y-axis of Figure 13.11, but the relative difference in the ensemble deviation in comparison to the ensemble mean is trivial.

Figures 13.12 and 13.13 show the pdf of the instantaneous aggregate goodput $N_A(t)$ at $t = 20s$ and $t = 40s$. Note that the mean of $N_A(t)$ at $t = 20s$ is about 125 packets/timestep and lower than the mean at $t = 40s$ which is about 130 packets/timestep. This is in line with the number of active stations; an decrease in the number of active stations (from 6 to 5) causes the increase in aggregate

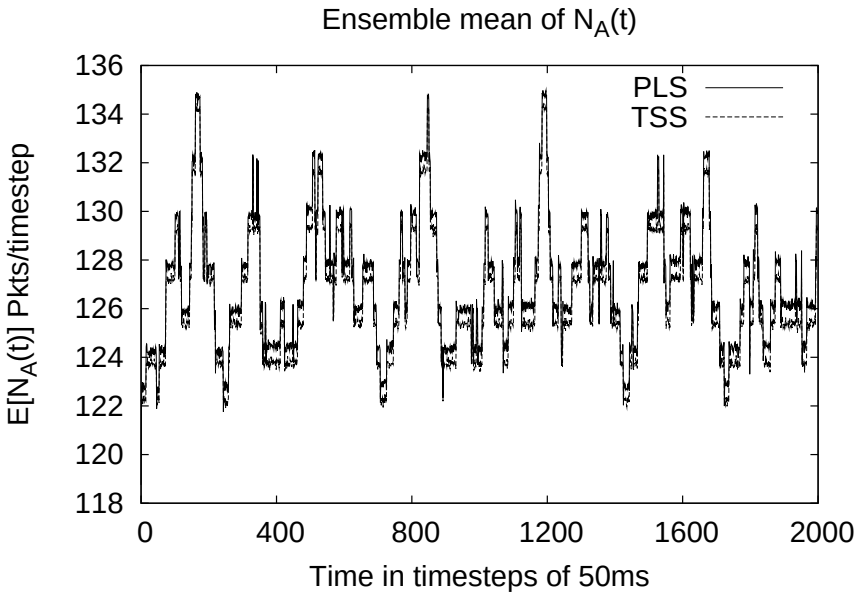


Figure 13.10: Ensemble mean of the instantaneous aggregate goodput $E[N_A(t)]$ as a function of time. Each point is computed over 1000 runs.

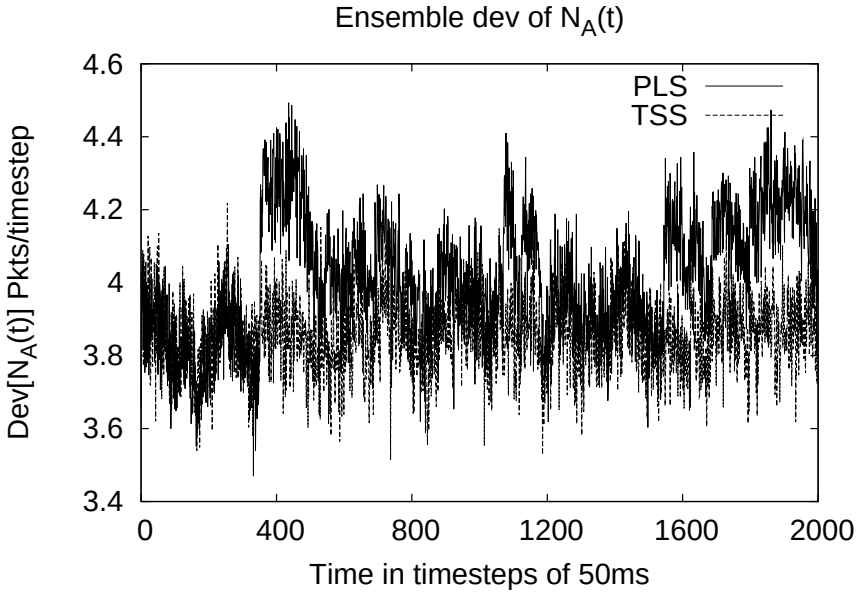


Figure 13.11: Ensemble deviation $Dev[N_A(t)]$ of the instantaneous aggregate goodput. Each point is computed over 1000 runs.

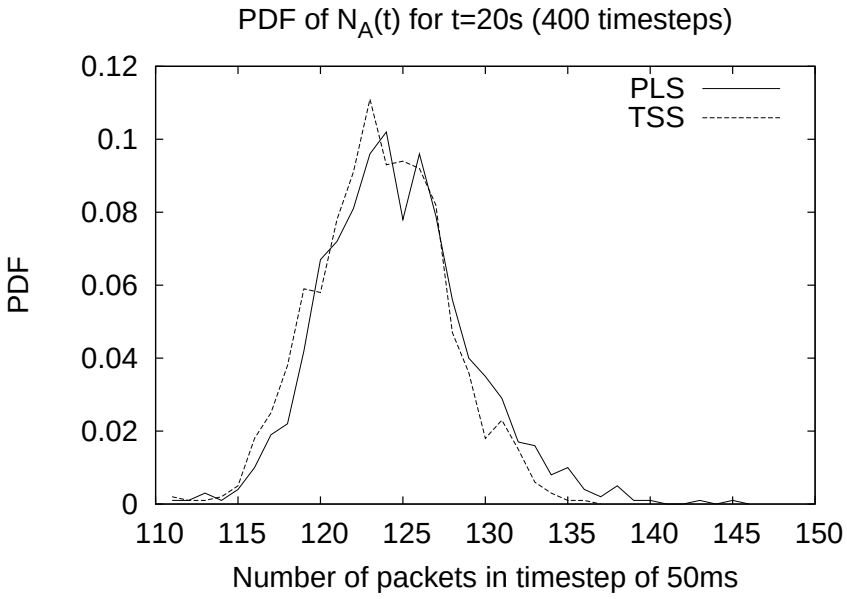


Figure 13.12: Ensemble PDF of $N_A(t)$ at $t = 20s$ computed over 1000 runs.

goodput. The distributions look gaussian for reasons explained before.

13.7 Ensemble metrics - Per-station

Figure 13.14 shows the ensemble mean of the instantaneous goodput of one tagged station as a function of time; Figure 13.15 shows the corresponding ensemble deviation. The ensemble average which was obtained over 1000 runs shows a close correspondence between the values predicted by TSS and PLS for the mean; they are visually indistinguishable. The deviation for PLS differs slightly from TSS, but the trend is captured effectively by TSS.

Figures 13.18 and 13.19 show the pdf of the contention window $C_i(t)$ at $t = 20s$ and $t = 40s$. The curves show an excellent match between PLS and TSS. Note that, as explained before, this explains the apparent discrepancy in the sample-path averaged distribution of $C_i(t)$ that is shown in Figure 13.7. Thus TSS tracks

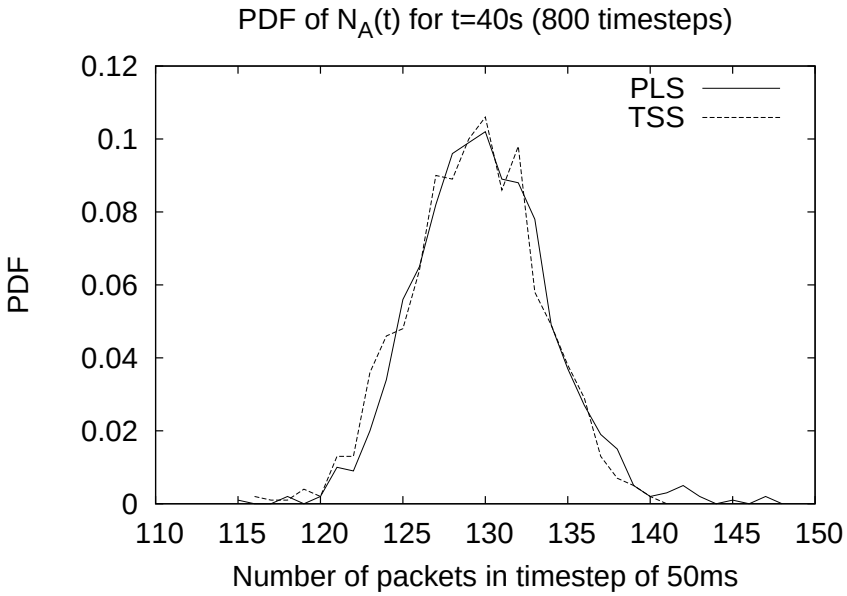


Figure 13.13: Ensemble PDF of $N_A(t)$ at $t = 40s$ computed over 1000 runs.

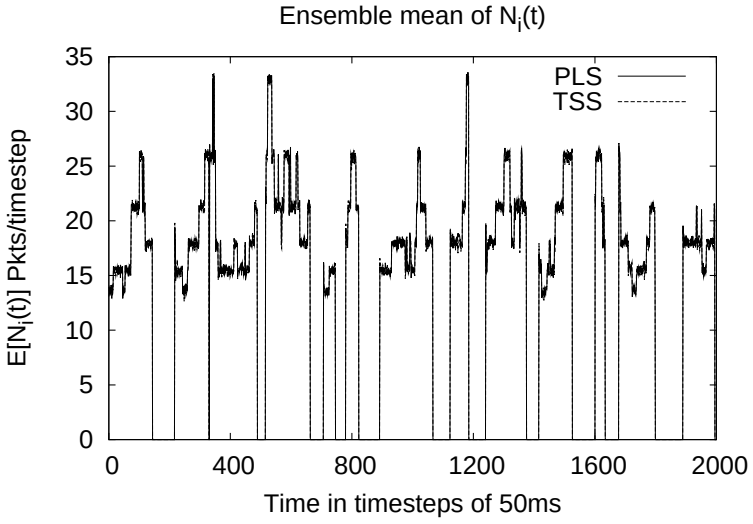


Figure 13.14: Ensemble mean of the instantaneous per-station goodput $N_i(t)$ for $i = 1$ versus time computed across 1000 runs.

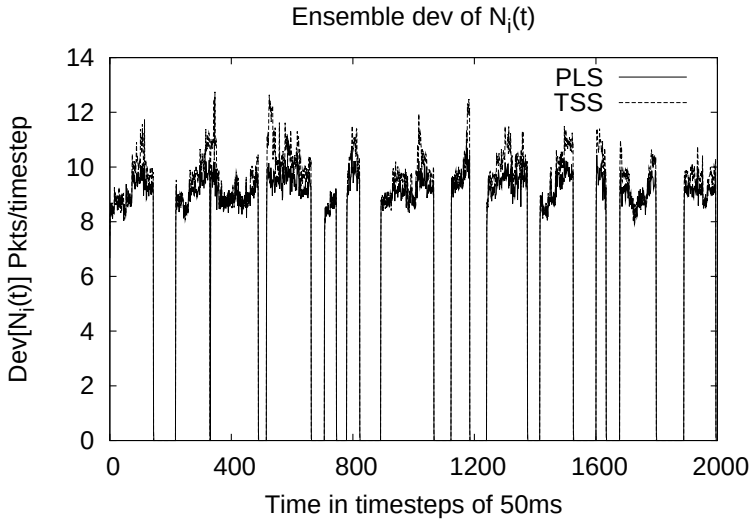


Figure 13.15: Ensemble deviation of the instantaneous per-sta goodput $N_i(t)$ for $i = 1$ versus time computed across 1000 runs.

the contention window accurately.

Figures 13.16 and 13.17 show the pdf of the per-station goodput $N_i(t)$ at $t = 20s$ and $t = 40s$ respectively. The means of the distributions are 15.61 and 26.11 respectively. The increase in the per-station mean is due to the decrease in the number of active stations from 6 to 5. At $t = 20s$ and $t = 40s$, station 1 is active, therefore probability of zero-goodput in those timesteps is only due to the MAC level interactions. These probability values of 0.074 and 0.024 are in line with the number of active stations.

Note that once a precomputed pdf is loaded from disk into memory by TSS, it is cached. Thus repeated loading of precomputed pdf's is avoided. The average time taken for one $1000s$ PLS run was $40.28s$ while for TSS it was $1.35s$ including the amortized pre-computation time and the full time for loading the precomputed pdf's from the disk for each simulation run.

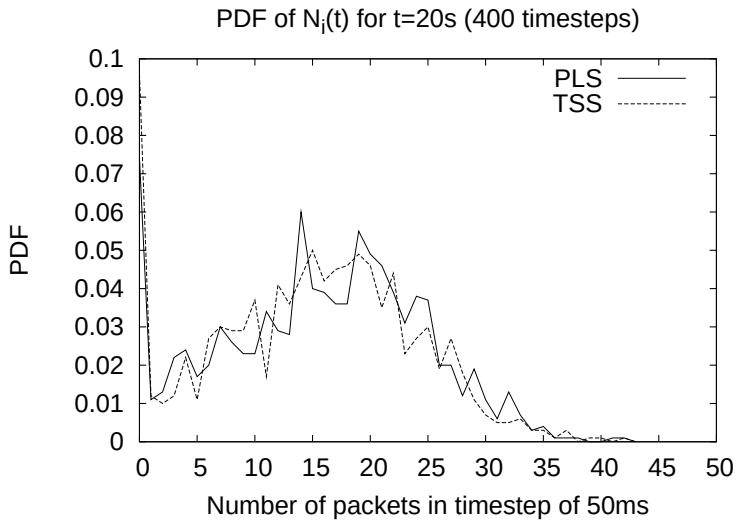


Figure 13.16: Ensemble PDF of the instantaneous per-sta goodput $N_1(t)$ at $t = 20s$ computed over 1000 runs.

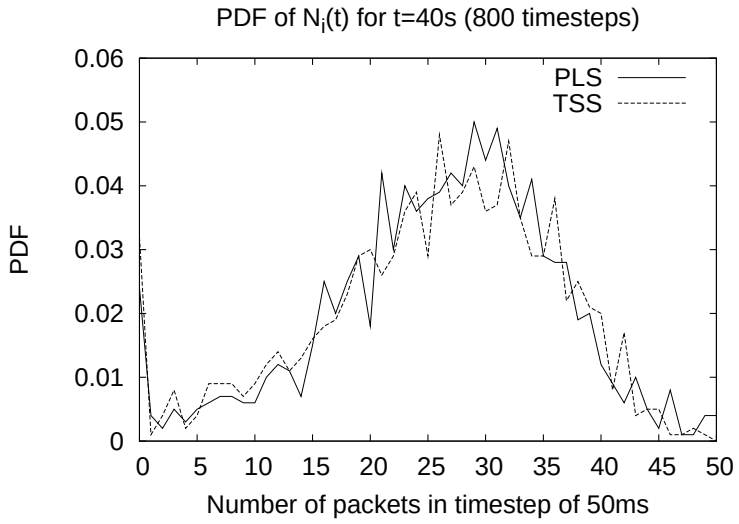


Figure 13.17: Ensemble PDF of the instantaneous per-sta goodput $N_1(t)$ at $t = 40s$ computed over 1000 runs

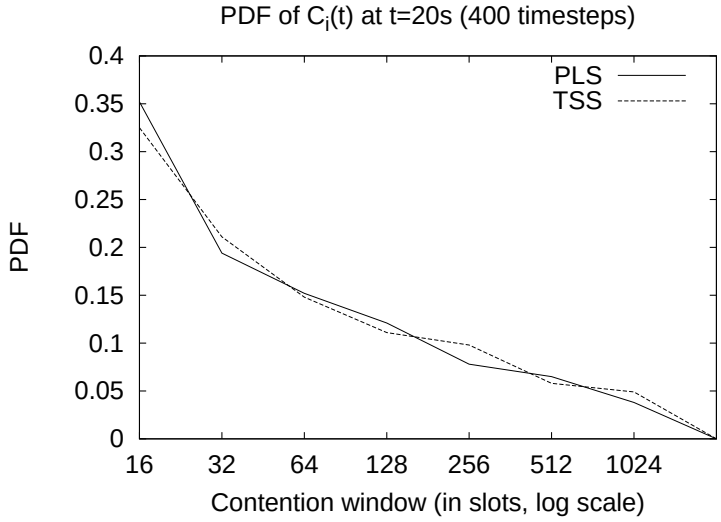


Figure 13.18: Ensemble PDF of the instantaneous contention window $C_1(t)$ at $t = 20s$ computed over 1000 runs.

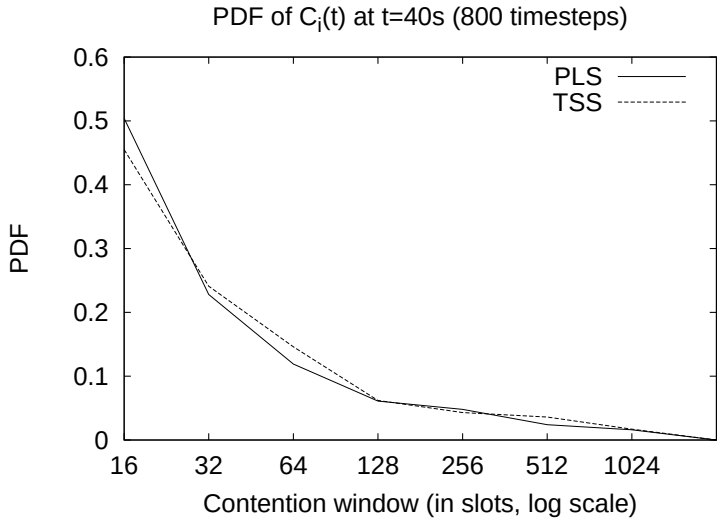


Figure 13.19: Ensemble PDF of the instantaneous contention window $C_1(t)$ at $t = 40s$ computed over 1000 runs.

Chapter 14

Conclusion and future work

Performance evaluation of computer networks is crucial. Analytical methods of performance evaluation are unable to adequately handle state dependent control mechanisms. Packet-level simulation, the *de facto* standard for performance evaluation, does not scale with increasing network size and workloads. Timestepped Stochastic Simulation (TSS) was previously developed as an alternative to packet-level simulation for point-to-point networks. This method generates sample paths of system state and instantaneous metrics over discrete timesteps, rather than at every packet arrival or departure. Because TSS updates the system state at timesteps, it is much faster than packet-level simulation. Because TSS generates sample paths, it can model state-dependent control accurately. This dissertation extends TSS for the case of shared links, specifically, 802.11 DCF based links. The key challenge is the combination of random access and history based scheduling. These factors cause short-term correlations across time and stations that need to be captured for accurate performance evaluation.

This dissertation presented a transient analysis of 802.11 and its application for TSS of WLANs, accounting for all the short-term correlations. The workload assumed that the number of active stations within an timestep remained constant.

First, the distribution of the aggregate goodput $N_A(t)$ was obtained. Next, we considered a tagged station within a timestep and obtained the conditional pdf of its instantaneous goodput $N_i(t)$ in the timestep conditioned on the MAC state $C_i(t)$ at the beginning of the timestep. Then, we obtained the conditional distribution of the new MAC state $C_i(t+\delta)$ conditioned on the old MAC state $C_i(t)$ and $N_i(t)$. All the analysis for these marginal distributions assumed that the rest of the stations' activity can be modeled by a constant collision probability for each attempt determined by the number of active stations irrespective of the history. Because these transient distributions can be easily parametrized in terms of the number of active stations and the timestep size, they can be precomputed or cached across simulation runs. The TSS technique used the pdf's computed by the transient analysis to sample instantaneous goodputs of all stations such that they 1) add up to the instantaneous aggregate goodput and 2) have the required correlation structure. This dependent sampling accounted for the negative correlation by choosing station id's according to a random permutation, and allocating a station's goodput depending on both its marginal distribution and the sum of all the goodputs allocated so far. In sum, the method obtains the sample path evolutions of the contention windows and instantaneous goodputs of all stations with time.

We validated the transient analysis and TSS technique against PLS, and quantified the runtime speedup obtained by TSS over PLS. The metrics that we considered included both those internal to the method (such as the pdf's obtained by the transient analysis) and those external to the method (the distribution, autocorrelation, and crosscorrelation of the contention window and the per-station goodputs). We found that TSS is accurate apart from the minor deviations from

PLS noted in Chapters 12 and 13. Further, TSS scales well with increasing number of stations and is independent of the bit-rate. Specifically, TSS provides up to two orders of magnitude improvement over our custom MAC level packet-level simulator; NS-2 is two orders still slower.

14.1 Future work

Some possible directions for future work include: 1) a timestepped TCP model over 802.11; 2) accounting for the physical layer; and 3) timestepped MAC models for related random access protocols.

References [47] and [70] provide models for TCP over 802.11, but they consider steady state performance and obtain bounds on long-term goodput; we are interested in a timestepped sample path. We believe this is a challenging problem for several reasons. First, existing results for TSS cannot be used directly for a TCP model because a TCP connection that has packets to transmit in a timestep is not typically fully active, i.e., it does not have packets to transmit throughout the timestep. Therefore these (TCP-level) idle intervals have to be modeled in obtaining any sample path metrics. This requires tracking the queuing process of all stations and obtaining the per-station collision probability for skewed unsaturated load regimes. One approach might be to obtain an empirical model for the per-station collision probability as a function of the load in the system. Second, the dependent sampling of the goodputs would be subject to additional constraints. For instance, consider a station performing a TCP upload using an 802.11 access point. The TCP ACKs of the station compete with the TCP DATA of the station for the same channel, and hence their goodputs are negatively correlated. However, whether the DATA is available for transmission in the

timestep itself depends on the ACK flow achieving a goodput. Finally, 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. In past work, the physical layer has been accounted for in references [14, 46]. However, they obtain long-term average goodputs by modeling the average interference in the PHY layer. Modeling a general PHY in a timestepped manner with several interacting WLANs is challenging for the following reasons. First, 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. Next, 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. A PHY-layer with higher fidelity fading models is also possible if the channel conditions can be abstracted within a timestep by a suitable noise-loss probability. However, it is likely that handling fast fading will be challenging within a timestepped model if the mean of the loss probability is not stationary over a timestep duration.

MAC protocols that are based on history are amenable to timestepped simulation on these lines. For instance, the 802.11e Enhanced DCF (EDCF) [50, 64] is one such protocol. It provides for priority classes in DCF by allowing differ-

ing DIFS values for multiple classes, allowing higher priority stations to always precede lower priority stations if they both have traffic. Reference [64] presents an analysis for obtaining the per-station collision probability in a scenario with multiple classes of traffic, and shows that multiple operating points of the protocol exist even in the steady state. We believe that the dependent sampling naturally captures this scenario as the protocol operation can move from one operating regime to another. However, obtaining the marginal distributions for unsaturated regimes would still be challenging as in the regular DCF.

Appendix A

Analysis of collision probability for finite retries

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 $E[X] = \sum_{i=1}^{i=\beta} p^{i-1} \frac{(\gamma 2^{i-1} - 1)}{2}$.

Consider the following two equations:

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

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

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

Reference [3] 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 12.12. Specifically, the solution

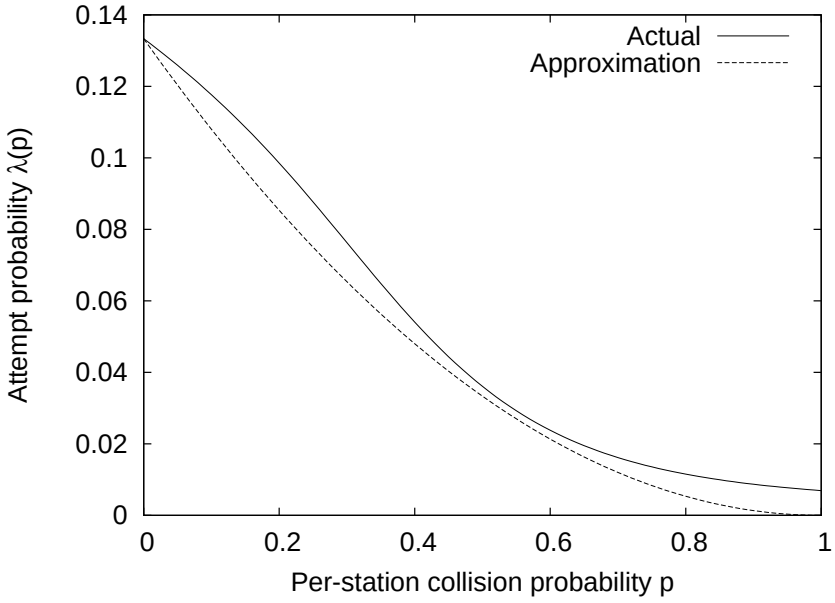


Figure A.1: $\lambda(p)$ approximated as $a(1-p)^2$.

$p(M)$ 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 A.1 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.

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

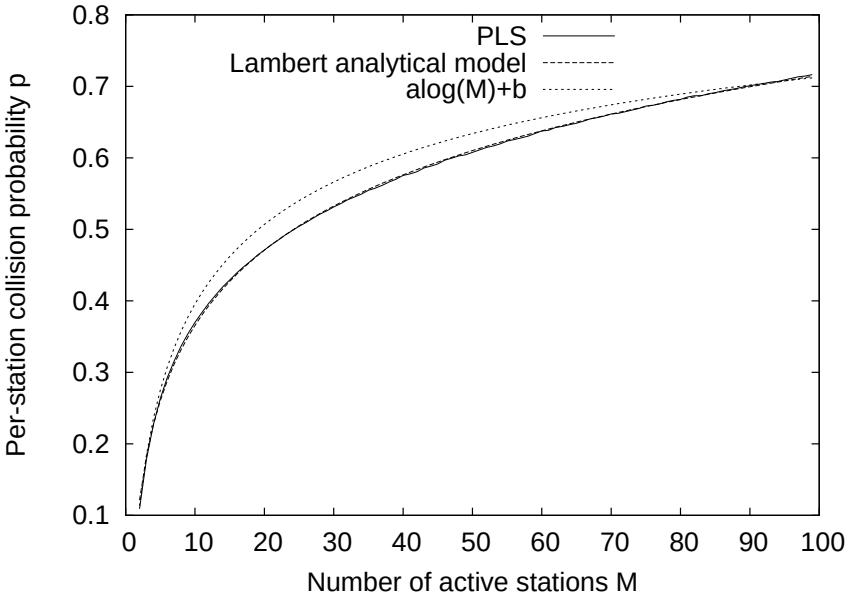


Figure A.2: Confirming the prediction of the model with simulation studies

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,

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

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 approximation,

and $p(M)$ as obtained by simulations in Figure A.2. 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. Interestingly, reference [26] shows an analysis of a log-like behavior for simple binary exponential backoff (in lemma 3 and theorem 4) without freezing backoff counters and infinite retries. Thus an exponential backoff mechanism leads to a logarithmic collision probability increase. However, with finite retries as the number of stations increases, the collision probability also tends to 1.

BIBLIOGRAPHY

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

[2] Qualnet network simulator. http://www.scalable-networks.com.

[3] 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.

[4] A. Papoulis, S. U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 2001.

[5] Imad Aad and Claude Castelluccia. Differentiation mechanisms for IEEE 802.11. In *INFOCOM*, pages 209–218, 2001.

[6] Zakhia Abichar, J. Morris Chang, and Daji Qiao. Group-based medium access for next-generation wireless lans. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 35–41, Washington, DC, USA, 2006. IEEE Computer Society.

[7] Albert Banchs, Pablo Serrano, and Arturo Azcorra. End-to-end delay analysis and admission control in 802.11 DCF WLANs. *Computer Communications*, 29, 2006.

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

[9] G. Bianchi and I. Tinnirello. Kalman filter estimation of the number of competing terminals in an IEEE 802.11 network. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2:844–852 vol.2, 30 March-3 April 2003.

[10] M. Blum. On the sums of independently distributed pareto variates. *SIAM Journal on Applied Mathematics*, 19(1):191–198, jul 1970.

[11] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka. A hybrid systems modeling framework for fast and accurate simulation of data communication networks. In *ACM SIGMETRICS*, pages 58–69, June 2003.

[12] Luciano Bononi, Marco Conti, and Enrico Gregori. Runtime optimization of IEEE 802.11 wireless LANs performance. *IEEE Trans. Parallel Distrib. Syst.*, 15(1):66–80, 2004.

[13] Frederico Calì, Marco Conti, and Enrico Gregori. Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit. *IEEE/ACM Trans. Netw.*, 8(6):785–799, 2000.

[14] M. Carvalho and J. Garcia-Luna-Aceves. A scalable model for channel access protocols in multihop ad hoc networks, 2004.

[15] Harshal S. Chhaya and Sanjay Gupta. Performance modeling of asynchronous data transfer methods of IEEE 802.11 MAC protocol. *Wirel. Netw.*, 3(3):217–234, 1997.

[16] Jaehyuk Choi and Joon Yoo. Eba: An enhancement of the IEEE 802.11 DCF via distributed reservation. *IEEE Transactions on Mobile Computing*, 4(4):378–390, 2005. Member-Sunghyun Choi and Member-Chongkwon Kim.

[17] 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.

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

[19] M. Ammar et. al. Simulation of Large-Scale Communication Networks How Large? How Fast? In *MASCOTS*, 2003.

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

[21] W. Feller. Diffusion Processes in One Dimension. In *Transactions of the American Mathematical Society*, volume 77, pages 1–31, July 1954.

[22] C. H. Foh and M. Zukerman. Performance Analysis of the IEEE 802.11 MAC Protocol. In *Proc. European Wireless Conference*, 2002.

[23] C. H. Foh, M. Zukerman, and J. W. Tantra. A Markovian Framework for Performance Evaluation of IEEE 802.11. *IEEE Transactions on Wireless Communications*, 6:1276–1265, April 2007.

[24] 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.

[25] 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.

[26] Jonathan Goodman, Albert G. Greenberg, Neal Madras, and Peter March. Stability of binary exponential backoff. *J. ACM*, 35(3):579–602, 1988.

[27] Z.J. Haas and Jing Deng. On optimizing the backoff interval for random access schemes. *Communications, IEEE Transactions on*, 51(12):2081–2090, Dec. 2003.

[28] Martin Heusse, Franck Rousseau, Romaric Guillier, and Andrzej Duda. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless lans. *SIGCOMM Comput. Commun. Rev.*, 35(4):121–132, 2005.

[29] 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.

[30] 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.

[31] Chen K. Ho T. Performance analysis of IEEE 802.11 csma/ca medium access control protocol. In *Proc. PIMRC, Taipei*, pages 407–411, Oct 1996.

[32] David P. Hole and Fouad A. Tobagi. Capacity of an IEEE 802.11b wireless LAN supporting VoIP. In *Proc. IEEE Int. Conference on Communications (ICC)*, 2004.

[33] 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.

[34] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison Wesley, 2006.

[35] L. Kleinrock and F. Tobagi. Packet switching in radio channels: Part i–carrier sense multiple-access modes and their throughput-delay characteristics. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 23(12):1400–1416, Dec 1975.

[36] 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.

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

[38] A. Kolmogorov. Ueber die analytischen Methoden in der Wahrscheinlichkeitsrechnung. In *Mathematical Annals*, volume 104, pages 415–458, 1931.

[39] L. Kleinrock. *Queueing Systems, Volume I*. John Wiley and Sons, 1976.

[40] Anders Lindgren, Andreas Almquist, and Olov Schelén. Quality of Service schemes for IEEE 802.11 - a simulation study. In *Proceedings of the Ninth International Workshop on Quality of Service (IWQoS 2001)*, June 2001.

[41] Y. Liu, F. Lo Presti, V. Misra, D. Towsley, and Y. Gu. Fluid Models and Solutions for Large-Scale IP Networks. In *ACM SIGMETRICS*, pages 91–101, June 2003.

[42] M. Ergen and P. Varaiya . Throughput analysis and admission control for IEEE 802.11a. *Mobile Networks and Applications*, 10:705–716, 2005.

[43] M. Gast. *802.11 Wireless Networks - The Definitive Guide*. O'Reilly, 2003.

[44] Benoit Mandelbrot. The pareto-levy law and the distribution of income. *International Economic Review*, 1(2):79–106, may 1960.

[45] Benoit Mandelbrot. The variation of certain speculative prices. *The Journal of Business*, 36(4):394–419, oct 1963.

[46] Mohammad Hossein Manshaei, Gion Reto Cantieni, Chadi Barakat, and Thierry Turletti. Performance analysis of the IEEE 802.11 MAC and physical layer protocol. In *WOWMOM '05: Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*, pages 88–97, Washington, DC, USA, 2005. IEEE Computer Society.

[47] Daniele Miorandi, Arzad A. Kherani, and Eitan Altman. A queueing model for http traffic over IEEE 802.11 WLANs. *Comput. Networks*, 50(1):63–79, 2006.

[48] V. Misra, W. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *SIGCOMM*, pages 151–160, 2000.

[49] 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*.

[50] The Institute of Electrical and Electronic Engineers Inc. IEEE 802.11e, 2005 edition (amendment)). *http://standards.ieee.org/getieee802/download/802.11e-2005.pdf*.

[51] Daji Qiao, Sunghyun Choi, and K.G. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless lans. *Mobile Computing, IEEE Transactions on*, 1(4):278–292, Oct-Dec 2002.

[52] 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.

[53] Colin M. Ramsay. The Distribution of Sums of Certain I.I.D. Pareto Variates. *Communications in Statistics - Theory and Methods*, 35(3):395–405, April 2006.

[54] R.Jain. *The Art of Computer System Performance Analysis*. John Wiley and Sons, 1991.

[55] Roehner, Bertrand and Winiwarter, Peter. Aggregation of independent paretian random variables. *Advances in Applied Probability*, 17(2):465–469, jun 1985.

[56] S. M. Ross. *Introduction to Probability Models.* Academic Press, Inc., 2000.

[57] G. Sharma, A. Ganesh, and P. Key. Performance analysis of contention based medium access control protocols. In *Proceedings of the IEEE INFOCOM*, 2006.

[58] 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.

[59] O. Tickoo and B. Sikdar. Queueing analysis and delay mitigation in IEEE 802.11 random access MAC based wireless networks. In *Proceedings of IEEE INFOCOM*, Hong Kong, China, 03/2004 2004.

[60] Kuo-Chang Ting, Mao yu Jan, Sung huai Hsieh, Hsiu-Hui Lee, and Feipei Lai. Design and analysis of grouping-based DCF (GB-DCF) scheme for the MAC layer enhancement of 802.11 and 802.11n. In *MSWiM ’06: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 255–264, New York, NY, USA, 2006. ACM.

[61] F. Tobagi and L. Kleinrock. Packet switching in radio channels: Part iii–polling and (dynamic) split-channel reservation multiple access. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 24(8):832–845, Aug 1976.

[62] F. Tobagi and L. Kleinrock. Packet switching in radio channels: Part ii–the hidden terminal problem in carrier sense multiple-access and the busy-

tone solution. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 23(12):1417–1433, Dec 1975.

[63] Alberto Lopez Toledo, Tom Vercauteren, and Xiaodong Wang. Adaptive optimization of IEEE 802.11 DCF based on bayesian estimation of the number of competing terminals. *IEEE Transactions on Mobile Computing*, 5(9):1283–1296, 2006.

[64] 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.

[65] Andras Veres, Andrew T. Campbell, M. Barry, and Li-Hsiang Sun. Supporting service differentiation in wireless packet networks using distributed control. *IEEE Journal on Selected Areas in Communications*, 19(10):2081–2093, 2001.

[66] W. Stallings. *Wireless Communications and Networks*. Prentice Hall, 2001.

[67] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma. Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement. In *Proceedings of the IEEE INFOCOM*, 2002.

[68] Yang Xiao and Jon Rosdahl. Performance analysis and enhancement for the current and future IEEE 802.11 MAC protocols. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(2):6–19, 2003.

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

[70] Jeonggyun Yu and Sunghyun Choi. Modeling and analysis of tcp dynamics over IEEE 802.11 WLAN. *Wireless on Demand Network Systems and Services, 2007. WONS '07. Fourth Annual Conference on*, pages 154–161, 24-26 Jan. 2007.

[71] 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.

[72] Hongqiang Zhai, Xiang Chen, and Yuguang Fang. How well can the IEEE 802.11 wireless lan support quality of service? *IEEE Transactions on Wireless Communications*, 4(6):3084–3094, December 2005.