Title
Exploiting Spatial Channel Occupancy Information in WLANs

Author
KRISHNAN, MICHAEL N.

Publication Date
2014

Peer reviewed|Thesis/dissertation

Exploiting Spatial Channel Occupancy Information in WLANs

by

Michael N. Krishnan

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineeing and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Avideh Zakhor, Chair
Professor Jean Walrand
Professor Jim Pitman

Spring 2014

Exploiting Spatial Channel Occupancy Information in WLANs

by

Michael N. Krishnan

Abstract

Exploiting Spatial Channel Occupancy Information in WLANs

by

Michael N. Krishnan

Doctor of Philosophy in Electrical Engineeing and Computer Sciences

University of California, Berkeley

Professor Avideh Zakhor, Chair

Current 802.11 networks do not typically achieve the maximum potential throughput despite link adaptation and cross-layer optimization techniques designed to alleviate many causes of packet loss. A primary contributing factor is the difficulty in distinguishing between various causes of packet loss, including collisions caused by high network use, co-channel interference from neighboring networks, and errors due to poor channel conditions.

In current 802.11 networks, there are two adaptations occurring simultaneously but independently each using packet loss as an indication of a different problem. The first is contention window adaptation, which implicitly assumes all losses are due to collision, and the second is rate adaptation, which implicitly assumes all losses are due to channel errors. Thus, in all high-loss scenarios, at least one of these two adaptations behaves incorrectly, unnecessarily increasing the impact of packet loss.

In this thesis, we propose a novel method for estimating various collision type probabilities locally at a given node of an 802.11 network. The key to our approach is a signal we call the busy-idle (BI) signal, which is a binary-valued record of the channel occupancy over time as observed locally by a given node. We show that if the access point(AP) periodically broadcasts its BI signal to associated nodes at an overhead of less than 2%, the nodes can use this information in conjunction with locally observable quantities to obtain partial spatial information about the network traffic. With this spatial information, nodes can estimate their probabilities of various types of loss and make adaptations to improve throughput and/or network utility.

We provide a systematic assessment and definition of the different types of collision, and show how to approximate each of them using local information and the AP's BI signal. We verify our methods using NS-2 simulations, as well as using the Ath5k open source wireless card driver in an experimental testbed. We show that our proposed methodology accurately estimates overall collision probability to within 5%. This experimental verification

demonstrates the feasibility of our collision probability estimation approach and the resulting throughput gains in practice.

In addition, we describe a suite of methods which exploit these loss probability estimates and the BI signal to improve network throughput and utility. We show via NS-2 simulations that we can achieve gains of up to 400% via modulation rate adaptation, 350% via contention window adaptation, 25% via packet length adaptation, and 50% via carrier sense threshold adaptation.

This thesis is dedicated to my parents, whose hard work provided me with both the opportunities and the inspiration to pursue my PhD.

Contents

List of Figures

List of Tables

List of Abbreviations

ACK	acknowledgement
AP	access point
ARF	auto rate fallback
BEB	binary exponential backoff
BER	bit-error rate
BI signal	busy-idle signal
CMSA/CA	carrier sense multiple access with collision avoidance
COLA	contention-aware link adaptation
CST	carrier sense threshold
CTS	clear to send
CW	contention window
DC	direct collision
DCF	distributed coordination function
GPS	global positioning system
LA	link adaptation
LAN	local area network
MAC	medium access control
NACK	negative acknowledgement
NS-2	network simulator 2
PHY	physical layer
RTS	request to send
SC	staggered collision
SC1	staggered collision of type 1
SC2	staggered collision of type 2
SNR	signal to noise ratio
TCP	transmission control protocol
TX signal	transmit signal
UDP	user datagram protocol
WLAN	wireless local area network

Acknowledgements

I owe a great debt of gratitude to my advisor, Professor Avideh Zakhor. Not only did she give me the tools to be an effective researcher, but she helped me work through the obstacles I faced and provided the guidance I needed to complete this thesis. Avideh has a lot of students working on a wide variety of topics, but I appreciate that she always had the time to meet with me and help me find some direction when I struggled.

I am also eternally grateful to the other members of my thesis committee, Professor Jean Walrand and Professor Jim Pitman. I gained a great deal of insight and ideas from Jean and his students, and Jim's challenging course in probability theory helped me to learn to think like a mathematician.

I must also thank my collaborators, Dr. Sofie Pollin, Dr. Ehsan Haghani, Shicong Yang, Miklos Christine, and Sherman Ng for their contributions to my research. Thanks to Sofie for helping to develop my ideas as I began my thesis. Thanks to Ehsan and Shicong for helping me to think of my research in different ways. Thanks to Miklos and Sherman for helping me deal with the many issues of real world hardware.

The funding for this thesis was provided by ARO MURI W911NF-08-1-0233. I am extremely grateful to the ARO for providing the financial resources to allow me to complete this work.

My thanks also extends to the other members of my group with whom I shared office space and lunch hours over the past few years: Cindy Liu, James Andrews, Matthew Carlberg, Nikhil Naikal, John Kua, Ricardo Garcia, Shangliang Jiang, George Cramer, Jimmy Tang, Stewart He, Nicholas Corso, Eric Turner, and Richard Zheng. I have truly been blessed to work in the company of such brilliant minds.

Last but not least, I would like to thank my family, and especially my wife, for their unending support, even when I had to spend more time with my computer than with them. They provided the inspiration I needed throughout this process.

Chapter 1

Introduction

1.1 Motivation

In 802.11 WLANs, nodes cannot distinguish between packet loss due to fading and collision because the symptoms are the same, namely a missing acknowledgement. The situations which result in each type of loss, however, require different specific actions to maximize throughput. For instance, *channel errors* occur when channel conditions are poor due to large path loss or multipath fading. These errors can be mitigated by using link adaptation (LA) to adapt the modulation and coding levels of each transmission, or by using forward error correction at the application layer. On the other hand, collisions happen when multiple transmissions occur at the same time causing interference and hence low relative signal power. Collision avoidance in the 802.11 distributed coordination function (DCF) is achieved by means of the Binary Exponential Backoff (BEB) scheme, where colliding nodes choose a larger random backoff counter to minimize repeat collision probability when retransmitting the packet.

In this thesis we distinguish between different types of collisions, based on the nature of the way they occur in the network. Our motivation is that by estimating probabilities of each type of collision, it should be possible to arrive at the corrective action to minimize their impact on the future transmissions, and hence improve throughput. Specifically, we focus on two broad classes of collisions, *direct* and *staggered*. A *direct collision (DC)* happens when two nodes simultaneously begin transmitting a packet. These occur in the 802.11 DCF when two nodes finish their backoff at the same time. Co-channel or hidden node collisions occur when transmissions from multiple far-away nodes that cannot sense each other overlap in time. In these collisions, transmissions do not necessarily start at exactly the same time.

As such, we refer to these collisions as *staggered collisions (SCs)*. Traditionally, staggered collisions are dealt with by transmitting Request-To-Send (RTS) and Clear-To-Send (CTS) messages before each data packet, since the hidden node likely receives the CTS message and avoids collision. This scheme however, is rarely used in practice due to the large overhead and delay.

We further subdivide SCs into two types, namely type 1 and type 2. A *staggered collision of type 1 (SC1)* for a given node is one in which the node under consideration begins its transmission first and is then interrupted by another node. A *staggered collision of type 2 (SC2)* for a given node is one in which the node under consideration interrupts the transmission of a hidden node. This distinction is necessary because these two types of staggered collisions each have a different cause, and thus each require a different modification to the link adaptation layer or packet scheduling algorithm. To avoid SC1s, it is possible to transmit with a higher transmit power, or to send shorter packets. On the other hand, to avoid SC2s, a node could decrease its carrier sense threshold in order to sense more nodes, thereby reducing the number of hidden nodes at the cost of deferring channel access more frequently.

Most current LA algorithms are based on Auto-Rate Fallback (ARF) [18], which is based on counting missed acknowledgment packets (ACKs). Specifically, every packet for which an ACK is not received is assumed to be lost due to poor channel conditions. When M consecutive packets are lost, the rate is reduced, and when N consecutive packets succeed, the rate is increased. This class of link adaptation algorithms performs poorly in the presence of collisions, because losses due to collisions are misinterpreted as losses due to poor channel conditions[8, 9]. While lowering the modulation rate reduces the probability of loss due to channel errors, it actually *increases* the probability of loss due to staggered collision, because it results in packets with longer durations.

Collision avoidance in the standard 802.11 MAC using carrier sense multiple access with collision avoidance (CSMA/CA) suffers from a similar problem of misinterpreting packet loss. In CSMA/CA, when a node has a packet to send but senses the channel as busy, it chooses a random number W between 0 and $CW - 1$, where CW is called the contention window size, and waits until it observes the channel as idle for a total of W slots before beginning its transmission. This way, if nodes competing for the channel choose different values of W, they can avoid collision. Ideally CW should be large enough that the probability of multiple nodes completing their backoff in the same slot is sufficiently small, but also small enough to avoid excessive delay. In the BEB protocol, CW is adjusted with every retransmission attempt of a packet. It initially starts at a value $CW_{min} = 32$ and increases by a factor of $\alpha = 2$ for each retransmission attempt, up to $CW_{max} = 1024$. The rationale is that a relatively small CW should be used to limit delay when there is low collision probability, but when the there are many nodes, collision probability increases, and a larger CW is needed. This is based on the often invalid assumption that all losses are due to neighboring nodes competing for the channel; in practice, losses can also be caused by poor channel conditions or the hidden terminal problem. If a node experiences high loss due to poor channel conditions, the BEB algorithm can result in the node increasing its contention window up to 20 ms, during which

time the node could have attempted as many as 10 more transmissions of 2 KB packets at 11 Mbps.

The key to dealing with these differing types of loss is to develop methods to differentiate between them. A number of approaches have been proposed to modify ARF to improve performance in the presence of collisions by differentiating collisions from channel errors [22, 26, 31, 33, 42, 47]. In [22, 26, 42] the authors use the Request to send/clear to send(RTS/CTS) mechanism to avoid collisions at certain times to eliminate the possibility of collisions, and base the choice of modulation rate on the results of packets during this collision-free time. However, turning on RTS/CTS can incur a significant overhead and delay, making it unattractive in practice. In [21], Kim et. al. propose an improvement to ARF assuming knowledge of the probability of collision, but do not suggest a method for obtaining this information in scenarios with hidden nodes.

The techniques in [31, 33, 47] focus on distinguishing collisions from channel errors on a per-packet basis. In [31], Pang, *et. al.* suggest the use of negative acknowledgements (NACKs) for when packets are received with errors, and in [33], Rayanchu *et. al.* suggest feeding back the entire erroneous packet to the sender to allow it to diagnose the type of error. One of the shortcomings of these approaches is that the receiving node is assumed to be able to synchronize to and decode the headers of all non-colliding packets so that it can send a NACK; in practice this is an unrealistic assumption, as Vyas et. al. show that for low modulation rates in 802.11a, most losses are due to failure to synchronize [40]. Additionally, these approaches introduce overhead that scales poorly as traffic and losses increase. In [47], Yun and Seo propose piggybacking timing information for lost packets onto future packets, so that past losses due to collision can be later identified. However, as with the previous solutions, the overhead scaling is quite poor. In [13], Gollakota and Katabi suggest a signal processing method for recovering collided packets, but their approach requires a significant amount of memory, as nodes must maintain high-resolution waveforms of all observed collisions.

In this thesis, we argue that real-time per-packet information is not necessarily needed, and that estimates of probabilities of the various loss mechanisms is sufficient to achieve significant throughput and utility improvements with much lower overhead. Specifically, we propose an approach for estimating various components of collision probabilities for 802.11 networks. by exploiting partial spatial traffic information. Unlike in wired networks, each node in a wireless network observes a different medium depending on its location. As a result, standard local sensing alone cannot allow a sending node to determine network traffic levels at the intended receiver. For a node to estimate the probability that its next packet collides, it needs spatial information about network traffic.

There is a growing body of literature on dealing with the spatial nature of wireless LANs [15, 23, 29]. In [23], Li *et. al.* model the throughput of an 802.11 network using full spatial information. Their approach is from a network point of view, while ours involves individual nodes locally estimating collision probabilities based on limited spatial information. In [29],

Nadeem and Ji suggest nodes piggyback location information based on GPS onto their packets to allow other nodes to build a spatial map of the network, but this only works outdoors, where GPS is available. In [15], a method is proposed to identify causes of "starvation" when a node achieves zero throughput, assuming saturation conditions in the network. However, in order to be useful in practical scenarios, it is desirable to determine the proportion of packets experiencing each type of loss for a broad range of traffic scenarios, not only when nodes are starved.

To our knowledge there has been little to no work on estimating collision probability in the presence of hidden nodes and channel errors. There has been some work [5, 38] on estimating the number of competing nodes in a single collision domain with no hidden nodes and no channel errors, which can allow for estimation of the probability of direct collision, but only in these limited scenarios. In [7], Choi *et. al.* extend this idea to scenarios with channel errors but no hidden nodes, and further assume that nodes can decode MAC headers of most other packets, which is not necessarily the case in multi-rate scenarios. In [35], Sarr *et. al.* suggest estimating collision probability empirically, assuming all losses are due to collision and not channel error.

1.2 Overview of our proposed scheme

Our setup is an 802.11 network in infrastructure mode, with many nodes sending uplink traffic to the AP. This could model a scenario in which many users are video conferencing or file-sharing, or a camera senor network. Our goal is to estimate the uplink collision probabilities for all nodes. As such, we ignore downlink traffic since there are fewer APs than nodes, and they tend to be more spatially spread out; hence there are fewer collisions between downlink traffic as compared to uplink. Downlink traffic from one AP can potentially also collide with uplink traffic to another AP; however, since these collisions involve uplink transmissions, we assume the problem is dealt with by the nodes sending the uplink traffic. We also assume the traffic to be stationary over the period of time over which collision probabilities are being estimated, and that the traffic of all nodes are independent. We do *not* assume the nodes to have any knowledge about packet lengths or traffic shape of other nodes.

The outline of our proposed scheme is as follows: All nodes collect local statistics about their sensed medium occupancy, in the form of a *busy-idle (BI) signal* and *TX signal*. The BI signal is a binary-valued signal over time which takes value 0 when the channel is *idle*, and 1 when it is *busy*. We define busy to mean that there is some transmission with received energy above the carrier sensing threshold of the node or AP, and idle to mean that there is no such transmission. The TX signal is another binary-valued signal taking value 1 when the node is transmitting and 0 otherwise. The AP periodically broadcasts its BI signal to all of its associated nodes. Since this is a periodic broadcast from the AP, the overhead

does not scale with the number of nodes or the number of packets. By comparing its own statistics with those from the AP, a local station is able to obtain a clearer picture of the spatial occupancy of the medium in order to estimate the probabilities of the different types of collisions. In the remainder of the thesis, we refer to the node under consideration for the computation of the collision probability as "the station", and refer to general nodes as "nodes".

To verify the feasibility and accuracy of the collision probability estimation technique, we design an experimental system utilizing standard off-the-shelf hardware and Ath5k[1], an open source driver for wireless cards with certain Atheros chipsets. The BI and TX signals are collected at every node and AP in the network, and need to be temporally aligned for collision probability estimation. In order to implement a full system, busy-idle signals would need to be computed and broadcast in real time; however without access to much of the lower level functionality of the wireless cards, this is not practical for our experiments, which use only off-the-shelf hardware. Thus, we opt to collect the data in real time, and process it offline in order to verify the accuracy of the resulting estimates.

1.3 Exploiting the BI signal and collision probability estimates

Equipped with the BI signal and collision probability estimation technique, we propose a suite of methods to improve throughput and utility performance in 802.11 networks via adaptation of modulation rate, contention window size, packet length and carrier sense threshold.

In Chapter 4, we leverage the estimate to improve link adaptation and thus increase throughput in an 802.11 network via two different approaches. First, we use the estimate in a modified version of ARF, based on [21]. Second, we propose a new LA algorithm, called SNRg, in which nodes estimate the channel conditions by comparing the empirical loss statistics to the expected loss statistics based on the estimated collision probability, and choose the optimal modulation rate for these conditions.

In Chapter 5 we also propose a distributed algorithm whereby each node uses information from the BI signal to adapt its contention window size in order to improve overall network utility. The key to the approach is that it is not necessary for individual nodes to know the contention window sizes or throughput of other nodes; the local and AP busy-idle signal contain enough information for each node to estimate the derivative of total network utility with respect to that node's average backoff length. Nodes can then employ a gradient ascent method to reach the optimal operating point for the network. This is done in a timescale on the order of seconds, and can be combined with packet-level adaptations, including the

traditional BEB or the methods proposed in [4, 24, 36, 44] to adapt to fast timescale changes to network traffic. Our technique shows significant improvements in three major situations in which standard contention window adaptation is ineffective. The first is when the number of nodes is large, the second is when the number of nodes is small, but there are is a poor channel, and the third is in the presence of hidden terminals. Contention Window adaptation literature tends to focus on the first case. This is because contention window adaptation is tailored to avoiding collisions. In [43] and [41], analytical models are derived to determine the optimal CW as a function of the number of nodes assuming all losses are due to collision. In [10], an algorithm with memory is proposed using packet loss counts to keep an updated estimate of the level of traffic, however since only local loss rates are used, it is impossible to distinguish collisions from channel errors. There are also several other adaptation schemes which update the contention window in a Markov manner that differs from the BEB [4, 24, 36, 44]. Our technique shows similar throughput improvements in the high node count scenario, while also improving utility and throughput in the other two scenarios.

Addtionally, in Chapter 6 we propose a local packet length adaptation algorithm whereby each node estimates its current probability of each type of packet loss in order to compute the derivative of throughput with respect to packet length, and to adapt accordingly. While there has been a significant amount of research on packet length adaptation, in most of the packet length adaptation literature, a simple packet loss model is used, assuming the channel to have a constant bit-error rate (BER), and neglecting staggered collisions [14, 45]. This assumes that most packet losses occur due to random bit errors in the packet payload. However, it has experimentally been shown that for lower modulation rates in 802.11a, most packet losses occur due to failure to synchronize to the packet preamble [40]. This type of loss cannot be accounted for using a constant BER model, as it requires a model of channel fading. In [49], Zheng and Nelson use a fading model, but only assume channel coherence over symbols rather than packets and omit staggered collisions. Staggered collisions have also been studied in the absence of channel errors [30, 37, 46]. One barrier to designing a more sophisticated adaptation algorithm is that in a network with multiple causes of packet loss, a node must be able to determine the the proportion of each type of packet loss it experiences in order to choose the optimal packet length. In [37] and [11], algorithms are proposed based on loss statistics in which nodes attempt to use different packet lengths to test whether performance increases or decreases. By avoiding modeling loss mechanisms, they sacrifice convergence speed. In [39], optimal packet length for a wireless sensor network is chosen based on a priori knowledge of network conditions, which cannot adapt to changing network conditions. Using our collision probability estimation technique, we are able to overcome some of the shortcomings of these approaches.

Finally, in Chapter 7 we propose a Carrier Sense threshold (CST) adaptation algorithm in which nodes use the BI signal to adapt their carrier sense thresholds to minimize the impact of hidden and exposed nodes in order to improve throughput. The impact of CST on the performance of the network has been studied by a number of researchers in recent

years. Ma *et al.* [25] propose a centralized algorithm for adjusting the CST based on loss differentiation. In their algorithm a central controller, such as an AP, periodically receives feedback from all stations and calculates the new CST to be used by all the nodes in the network. In this work, since all the nodes use the same CST, the network is unlikely to reach the maximum achievable throughput. Nadeem and Ji [28] propose an enhanced DCF protocol that incorporates the location information to increase the spatial reuse. In their work, it is assumed that each node knows its own location and includes this information in each transmission so that all overhearing stations can predict whether their transmissions affect the ongoing transmission. Zhu *et al.* [50] propose an algorithm to adjust the CST so as to maximize the aggregate throughput given a minimum required Signal to Noise Ratio (SNR). Their proposed adaptive algorithm uses the packet error rate to update the CST. Park *et al.* [32] propose an adaptive algorithm for updating CST. This work also uses packet error rate to update CST, but similar to [50], it does not differentiate between various causes of packet loss i.e., collisions and channel errors; this is problematic because CST adaptation primarily affects collision rates, and not channel errors.

1.4 Thesis organization

The thesis is organized as follows: In Chapter 2, we describe the busy-idle signal and our collision probability estimation technique and demonstrate its accuracy via NS-2 simulation. In Chapter 3, we verify the feasibility and accuracy of our technique experimentally using open source drivers for collecting data for the BI signal in real-time and processing it offline to compute estimates. In Chapters 4-7, we propose methods in which the BI signal and collision probability estimates can be used to improve throughput and/or utility in wireless LANs, providing NS-2 simulation results in each chapter. In Chapter 4, we show how to leverage the estimates to improve link adaptation. In Chapter 5, we propose a contention window adaptation technique using the estimates and BI signal. In Chapter 6, we propose a packet length adaption technique which leverages our improved loss model and collision probability estimation. In Chapter 7, we devise a method to use information from the BI signal to adapt the carrier sense threshold. Finally, we conclude the thesis in Chapter 8.

Chapter 2

Collision Probability Estimation

One of the main goals of this thesis is to distinguish between various types of loss events in WLANs. The primary method of accomplishing this is to use spatial information in the form of the busy-idle signal to estimate the probabilities of each of the three types of collsions: DCs, SC1s, and SC2s. In this Chapter we define the busy-idle signal, which is the key to the remainder of the thesis. We describe the method by which it can be used to locally estimate the probability of each type of collision at each station. The Chapter is organized as follows: In Section 2.1, we define the BI and TX signals and discuss related transmission overhead; in Section 2.2, we detail the estimation technique; we present simulation results in Section 2.3.

2.1 Traffic statistic collection and transmission

The transmission of packets by the station is governed by the state of the medium which it can sense. In particular, it can only send when it senses the local medium as idle. However, the reception of the packets depends on the state of the medium around the destination, the AP. Therefore to estimate the probability of collision, the station must be able to compare the occupancy of the medium locally to that at the AP.

We model the traffic state of the medium at each node or AP as a binary-valued process over time taking the value 0 when the channel is *idle*, and 1 when it is *busy*. We define busy to mean that there is some transmission with received energy above the carrier sensing threshold of the node or AP, and idle to mean that there is no such transmission. We call this the busy-idle process at that node or AP. Because the sensing of transmissions requires a much lower SNR than the decoding, the busy-idle signal is much more robust to fading than

statistics about successfully received packets. In practice, the collection of this signal may be imperfect because there is a non-zero chance of misclassifying the medium as busy or idle at any given time; however, errors caused by random noise are uncorrelated over samples, and can be filtered out using knowledge of the minimum packet length and interframe spacing. For the purposes of this thesis, we assume a perfect signal to be obtainable. In addition to the BI signal, the station also must collect a TX signal which is temporally aligned with the BI signal. The TX signal is also binary-valued and takes value 1 at the times when the station begins its transmission and 0 everywhere else.

The complete signal required for collision probability estimation is the vector valued signal $BI(t) = [BI_{STA}(t), TX_{STA}(t), BI_{AP}(t)]$, where $BI_{STA}(t)$ and $BI_{AP}(t)$ denote the busy-idle signals at the station and AP, respectively, and TX_{STA} denotes the transmit signal of the station. The station has access to $BI_{STA}(t)$ and $TX_{STA}(t)$, but it must acquire $BI_{AP}(t)$ from the AP. Transmission of this complete waveform from the AP to the nodes could result in a significant amount of overhead. However, perfect resolution is not needed for reasonable estimation accuracy. It is only necessary to have at least one sample per backoff slot time, since nodes check the status of the medium at this resolution to determine when to send. Furthermore, since the information needed by all stations is the same, the AP can simply broadcast this information so that the overhead does not scale with the number of nodes or amount of traffic.

For an estimate of the number of bits that must be sent by the AP, we assume there are at most 1000 non-colliding packets per second heard by the AP. This results in 2000 edges in the busy-idle process per second. For 10μs resolution, we can upper-bound the amount of information by a 2-state discrete time Markov chain with transition probability 0.02. The entropy rate of this process is 0.141 bits per sample, or 14.1 kpbs. Because there is also regular transmission overhead for this data, the precise overhead will depend on how frequently it is sent, but if the AP were to send this information every few seconds at the lowest possible modulation rate, i.e. 1 Mbps for 802.11b, the overhead would be on the order of 2%.

2.2 Estimating collision probabilities

To compute the probability that its next packet experiences a collision, each node uses the available statistics to compute its probability of each type of collision, and combines them using the relation:

$$(1 - P_C) = (1 - P_{SC2}) \times (1 - P_{DC}) \times (1 - P_{SC1}) \tag{2.1}$$

where P_{SC2} is the probability the next packet experiences a staggered collision of type 2, P_{DC} is the probability the next packet experiences a direct collision given that it does not

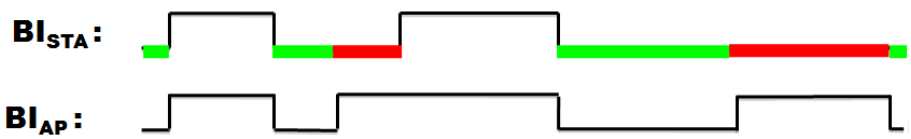


Figure 2.1. Example of P_{SC2} estimation

experience a staggered collision of type 2, and P_{SC1} is the probability that it experiences a staggered collision of type 1 given it does not experience either of the other types of collision. The order in which the probabilities occur in Equation (2.1) matches with the natural order of events. Specifically, an SC2 occurs when the channel is busy at the AP before the station starts to send, a DC occurs when another node starts at the same time, and an SC1 occurs when another node interrupts the station after the transmission has begun. The remaining packet losses are the result of channel errors.

In the subsections that follow, we will describe the estimation of each of the probabilities in Equation (2.1). Once the station estimates P_{SC2}, P_{DC}, and P_{SC1}, it combines them via Equation (2.1) to estimate its total probability of collision. The estimates are verified via NS simulations in Section 4.3.

2.2.1 Probability of staggered collision of type 2

An SC2 for the station occurs when the station starts sending during a time in which the channel is already busy at the AP. If the station has access to the full busy-idle process at the AP, it can identify the times when SC2s occur because these are times when it starts sending while the AP is already busy. Even if the statistics do not cover all times or are imperfect, the station can identify all of the times it could possibly have started sending, and count what proportion of these would have experienced SC2s by examining the state of the busy-idle process at the AP at those times. Specifically, P_{SC2} can be estimated as:

$$P_{SC2} = P(BI_{AP} = 1 | BI_{STA} = 0) \tag{2.2}$$

Figure 2.1 shows an example of a how P_{SC2} is estimated over a short segment of BI signals. The upper waveform is the station's BI signal, and the lower one is the AP's. The times during which $BI_{STA} = 0$ are shaded. Green indicates that $BI_{AP} = 0$ as well and red indicates that $BI_{AP} = 1$. The estimate of P_{SC2} is simply the total length of red divided by the total length of green plus red.

Table 2.1 shows the absolute value of percentage error in estimating P_{SC2} as a function of sampling period of the busy-idle signal at the AP, averaged over 300 nodes from 30 different NS-2 simulations. Each simulation has a single access point, and 10 nodes randomly located

Table 2.1. Absolute error percentage for P_{SC2} as a function of sampling period, averaged over 300 nodes

inter-sample time (μs)	1	2	4	10	15	20	30
mean absolute error (%)	0.4	0.6	0.9	1.6	1.7	3.8	34.07

in one of three regions: right next to the AP, or at the left or right edge of its coverage range. In this way, the nodes on either edge experience SC2s from the nodes on the other edge. As seen, the mean absolute error is below 3% for resolutions higher than 20μs, i.e. one sample per idle slot time. As expected, beyond 20μs resolution, the error becomes increasingly large. For the remaining NS-2 simulations in this thesis, we use a sampling period of 10μs.

Further, if the binary-valued busy-idle signal at the station is replaced by a continuous-valued signal indicating the level of power sensed on the channel, the station can determine its potential busy-idle processes for different carrier-sensing thresholds. Note that finer quantization of this signal does not affect the overhead, since it is not transmitted. With this increased information, the station can determine the sensitivity of its probability of SC2 to its current sensing threshold. For example, if the continuous-valued version of the busy-idle signal at the station takes on a value slightly below the current sensing threshold immediately before the station sends a packet which experiences an SC2, then use of a lower threshold would have resulted in the station sensing the channel as busy immediately before its transmission; in this case the SC2 would have been avoided. Knowledge of this sensitivity can be used for CST adaptation as shown in Chapter 7.

2.2.2 Probability of direct collision

A direct collision for a station occurs when another node begins its transmission at the same time as the station. This event cannot be directly observed by comparing the busy-idle traces of the station and the AP since the transmission of the station itself causes the AP to become busy, and there is no way for it to know whether there is also another node sending.

However, if some simplifying assumptions are made about the behavior of nodes, its is possible to arrive at a reasonable approximation for P_{DC}. For our approximation, we assume that all nodes are equally likely to send at any time in which they sense the channel as idle. In [6], Bianchi shows that nodes in a single collision domain with saturated traffic behave in this way when each time step is the length of a backoff slot. Increasing the time resolution does not significantly change this behavior. However, changing the rate of application layer traffic so that the transmit queue is not always backlogged can affect this behavior. We will show via simulations that our assumption results in reasonable estimates given that the application layer traffic is Poisson.

For a direct collision to occur at time t, the AP must sense the channel as idle at time $t-1$,

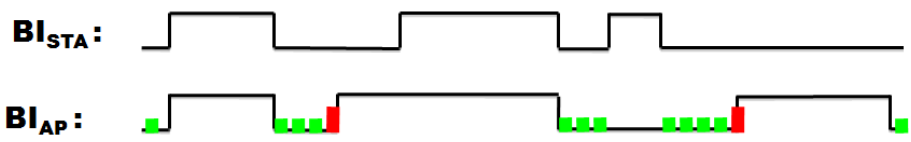


Figure 2.2. Example of P_{DC} estimation

and then hear the start of transmissions by both the station and another node simultaneously at time t. If the station is assumed to be equally likely to send at any time when the channel is idle around it, the probability of direct collision can be computed as the probability that some other node begins to send at time t when the channel is idle at both the station and the AP at time $t-1$. Using the busy-idle signals, the station can estimate P_{DC} as

$$P_{DC} = \frac{\sum_t \mathbb{1}\{BI(t-1) = [0,0,0], BI_{AP}(t) = 1, TX_{STA}(t) = 0\}}{\frac{1}{T}\sum_t \mathbb{1}\{BI(t-1) = [0,0,0], TX_{STA}(t) = 0\}} \tag{2.3}$$

where T is the length of a backoff slot and $\mathbb{1}\{\cdot\}$ is the indicator function. The numerator corresponds to the number of times another node starts to transmit when the station could have started its transmission, and the denominator is the total number of opportunities the station had to transmit. It is required that $TX_{STA} = 0$, since if the node itself is sending, it is impossible to determine if another node is also sending from the BI signal. The $1/T$ is needed in the denominator because the station only attempts to send 1 of every T time steps – those at the start of a backoff slot.

Figure 2.2 shows an example of a how P_{DC} is estimated over a short segment of BI signals. The upper waveform is the station's BI signal, and the lower one is the AP's. The colored blocks correspond to the times in the denominator of (2.3). The taller red blocks are the numerator, namely the times the AP becomes busy while the station does not.

2.2.3 Probability of staggered collision of type 1

P_{SC1} is the most difficult probability to estimate because it depends on the action of hidden nodes with respect to the station while the station is sending. But since the transmission of the station causes the medium to be busy at both the station and the AP, there is no way to observe this behavior.

To develop a method to estimate P_{SC1}, we start by considering a simple scenario consisting of a single AP which hears all traffic, and two sets of grouped nodes which we call "local" and "hidden". The nodes in each set can all hear members of their own set, but no nodes in one set can hear any of the nodes in the other. We then examine the additional phenomena we have to account for in the more general case where there are multiple APs,

and a more complex connectivity graph between nodes, so that the set of hidden nodes to any given node may be unique to that node.

In the simple scenario, we assume there are two sets of nodes which cannot sense packets from each other. Thus, the processes determining the times the nodes from each set send packets are independent. Let τ_l denote the rate at which local nodes begin to send given that they observe the channel to be idle, and let τ_h denote the rate at which hidden nodes begin to send given that they observe the channel to be idle.

Then, τ, the total probability that there is a packet heard by the AP given that the channel was idle in the previous time is given by

$$\tau = 1 - (1 - \tau_l)(1 - \tau_h). \tag{2.4}$$

From its local statistics, a local station can estimate τ_l as

$$\tau_l = \frac{\sum_t \mathbb{1}\{BI_{STA}(t-1) = 0, BI_{STA}(t) = 1\}}{\frac{1}{T}\sum_t \mathbb{1}\{BI_{STA}(t) = 0\}} \tag{2.5}$$

Similarly, from the statistics broadcast from the AP, the local station can estimate τ as

$$\tau = \frac{\sum_t \mathbb{1}\{BI_{AP}(t-1) = 0, BI_{AP}(t) = 1\}}{\frac{1}{T}\sum_t \mathbb{1}\{BI_{AP}(t) = 0\}} \tag{2.6}$$

Combining (2.4), (2.5), and (2.6), the station can estimate τ_h from the pair of busy-idle signals.

The probability that a packet sent by the station avoids an SC1 is the probability that no hidden nodes send during the station's packet. This probability is given by $(1 - \tau_h)^{L^*}$, where L^* is the *effective length* of the packet for the hidden nodes, that is the number of opportunities the hidden nodes have to send during the successful transmission of the packet.

This is strongly related to the notion of *virtual slots* in [6]. In [6], Bianchi shows that a single collision domain 802.11 network with saturated traffic can be thought of as operating in discrete time where the 'virtual' time slots are of variable length, i.e. either a short slot which is the length of a backoff slot when no one is sending, or a long slot which is the length of a full transmission, ACK, and inter-frame spacing when a node sends a packet. In our scenario, there are two separate collision domains, each with independent notions of virtual time. The station's transmission has a length of one virtual slot in its own collision domain, but lasts for more virtual slots in the other collision domain.

In this simple scenario, the station can compute L^* locally as the duration of its transmission – including inter-frame spacing and ACK – divided by length of a backoff slot, since in order for the transmission to be successful, all hidden nodes must remain silent, making all the virtual slots short backoff slots. Thus the station can easily compute P_{SC1} as

$$P_{SC1} = 1 - (1 - \tau_h)^{L^*}. \tag{2.7}$$

In a more general network topology with multiple AP's not hearing everything, and unconstrained node locations, there are multiple hidden nodes which hear different subsets of the other nodes, making the problem more difficult. There is no longer a single number τ_h which characterizes the behavior of all the nodes of interest, nor is there a single L^* for all of these nodes. Further, the presence of nodes which are heard by both the station and its hidden nodes causes a phenomenon we call the "coupling of transmission times". We discuss these three issues in the following three subsections.

2.2.3.1 Time-variation of τ_h

For a general network topology, we define $\tau_h(t)$ as the probability that at least one of the hidden nodes with respect to the station starts to transmit in the next time instance. This quantity is time-varying because the hidden nodes are affected by other nodes which are not in the set of hidden nodes.

The greatest difficulty with estimating $\tau_h(t)$ is that from the busy-idle signals, the station can only directly determine the network behavior while it is not sending. This is because when the node itself is sending, $BI(t) \equiv [1,1,1]$ regardless of what the hidden nodes are doing, since the AP and station both hear the station's transmission. In order to estimate P_{SC1}, the station needs to estimate $\tau_h(t)$ while it is sending. We approximate this value as a constant which we denote by τ_h^*. Similarly, we denote by τ_h^{idle} the average value of $\tau_h(t)$ while the medium is idle at both the AP and the station. This value is straightforward to observe from the busy-idle signals as

$$\tau_h^{idle} = \frac{\sum_t \mathbb{1}\{BI(t-1) = [0,0,0], BI(t) = [0,0,1]\}}{\frac{1}{T}\sum_t BI(t) = [0,0,0]}. \tag{2.8}$$

As seen shortly, our approach for estimating τ_h^*, and hence P_{SC1}, is to first estimate τ_h^{idle}, and then scale it by an appropriate scaling factor, $\bar{R} > 1$ to obtain τ_h^*.

The primary factor which causes τ_h^* to be greater than τ_h^{idle} is what we call the silencing of competing nodes. An example of this is shown in Figure 2.3, where the circles around nodes A and B represent their respective sensing/sensed ranges. We define the sensing range of node A as range of nodes which A can sense, and the sensed range as the range of nodes which can sense node A. For simplicity, in this discussion we assume the sensing and sensed ranges to be the same. If node A is sending, nodes C and D are silenced, since they can sense A. Therefore, the only nodes that can transmit are those in the shaded region in Figure 1. As a result, node B experiences less competition for the channel when A is sending than when A is not sending, since in the latter case B has to compete with all nodes within its sensing range.

We now elaborate on this process for a given hidden nodes B as shown in Figure 2.3. To begin with, assuming a sufficiently large number of nodes spaced uniformly at random,

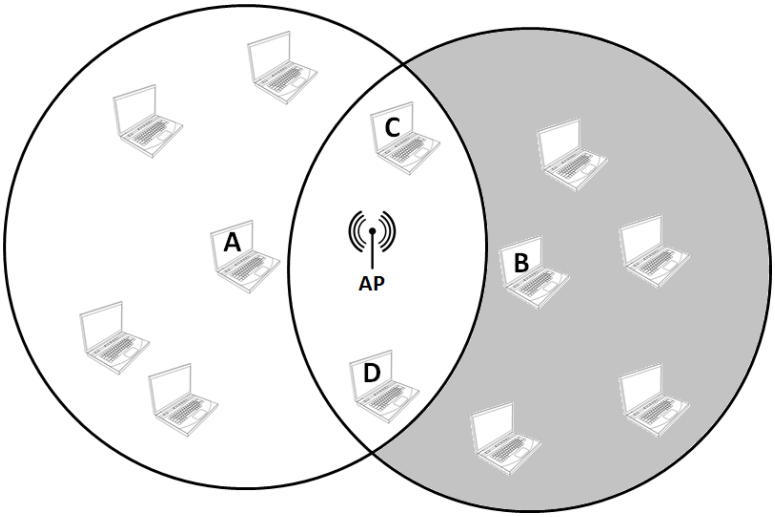


Figure 2.3. The silencing of competing nodes: While node A is sending, only nodes in the shaded region can send, because the other nodes hear node A and are silenced.

the number of nodes competing with node B for the channel roughly scales with the area within B's sensing range for which the channel is currently clear. Furthermore, the sending rate of node B is inversely proportional to the number of nodes it competes with. Putting these together, we conclude that the sending rate of a node B scales inversely with the area around it for which the channel is clear. The crescent-shaped shaded region in Figure 2.3 corresponds to the portion of node B's sensing range for which the channel is potentially clear while the station A is transmitting. When A is not transmitting, the entire circle around B is potentially clear. Let $R(d_B)$ denote the ratio of the area of the circle around B to the area of the shaded region, which is a function of d_B, the distance between A and B. Then the rate at which node B sends while node A is sending is given by:

$$\tau_B^* = \tau_B^{idle} R(d_B) \tag{2.9}$$

where τ_B^{idle} denotes the rate at which node B sends when the station A is not sending.

A similar effect occurs for each of the hidden nodes with respect to the station. Therefore, τ_h^*, the average probability that a hidden node sends in a given virtual slot while the station is sending, is given by:

$$\tau_h^* = \sum_{i \in \mathcal{H}} \tau_i^{idle} R(d_i) \approx \tau_h^{idle} \bar{R} \tag{2.10}$$

where $\mathcal{H}$ is the set of hidden nodes to the station A, and $\bar{R}$ is the average of $R(d)$ over all these nodes.

The approximation in Equation (2.10) assumes that there are no hidden nodes with significantly different distances and sending rates from the others, as this would potentially necessitate scaling one portion of the total transmission probability by a drastically different

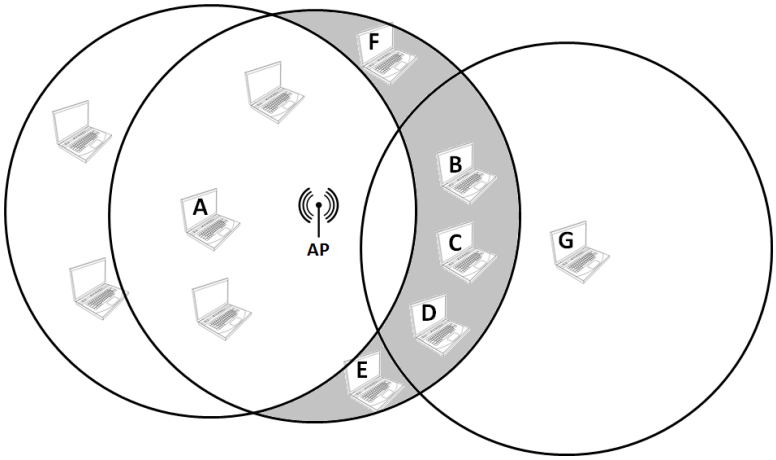


Figure 2.4. The silencing of hidden nodes: If node G starts sending, nodes B through D are silenced and cannot cause a SC1 with A. Effectively, they observe a shorter packet, in virtual slots, than nodes E and F.

value. In practice, this is a reasonable assumption since due to the geometry constraints, hidden nodes have a limited range of possible distances from the station. Specifically, they must be far enough to not sense the station, but close enough to interfere at the AP. Since the station must be within about half of an interference range from the AP, the admissible region for hidden nodes with respect to the station is typically a slim, crescent-shaped region, similar to the shaded region in Figure 2.4.

Since $\bar{R}$ is unknown, the station must estimate it as the expectation of $R(d)$ taken over the distribution of d. For the results in this thesis, we assume that nodes are distributed according to a spatial Poisson process. Depending on the application, it may be possible to have another distribution, but regardless, $\bar{R}$ can be pre-computed and stored.

2.2.3.2 Non-uniformity of L^*

The other key parameter used for the estimation of P_{SC1} is the effective length of the packet for the hidden nodes, L^*. Recall that for a hidden node with respect to the station, the effective length L^* of a packet sent by the station is the number of opportunities for the hidden node to interrupt it, or equivalently, the number of virtual slots that elapse for the hidden node during the transmission of the station's packet. If there are no nodes which are sensed by the hidden nodes but not by the AP, L^* is straightforward to compute as previously mentioned. However, in an actual network, there may be nodes exposed to the hidden nodes which are not sensed by the AP, but which silence some of the hidden nodes for a period of time; this gives these hidden nodes less opportunities to interrupt the packet sent by the station, making L^* effectively shorter for those hidden nodes. An example of this is shown in Figure 2.4 where the circles around nodes A and G represent the range for which they can

be sensed, and the circle around the AP represents its interference range. The shaded region denotes the region of hidden nodes with respect to the station A, namely nodes B through F, which are within the interference range of the AP yet outside the sensed range of A. If node G starts sending, hidden nodes B, C, and D are silenced, and thus experience fewer transmission opportunities than experienced by hidden nodes E and F, making A's packet effectively shorter for those nodes. It is useful to think of each node as having a different perception of virtual time. In this example, the transmission of node G causes nodes B, C, and D to freeze the virtual time in one long slot while virtual time continues to elapse in many short slots for nodes E and F. Thus, depending on the position and activity of exposed nodes such as G, there is a time- and space-varying scaling between effect packet length and real-time packet length at the hidden nodes.

For the station, A, the probability that a given hidden node, B, causes an SC1 can be expressed as

$$P_{SC1}^{(B)} = 1 - (1 - \tau_B^*)^{L^*{}_B(l)} \tag{2.11}$$

where l is the length of the packet in real time, and $L^*{}_B(l)$ is the effective length of the packet sent by the station in virtual slots as observed by node B. The difficulty in directly applying Equation (2.11) is that virtual time, $L^*{}_B(l)$, does not necessarily progress linearly with real time, l. If this were the case, plotting P_{SC1} versus l would yield an exponential according to Equation (2.11). Through NS-2 simulation, we have empirically shown this not to be the case.

Specifically, Figure 2.5 shows a plot of an example cumulative histogram of the "real" time, l, between the start of the station's packet and the start of the next packet sent by any of its hidden nodes. To obtain this plot we run NS-2 simulations for a network with 7 APs at fixed locations covering hexagonal cells, and 50 nodes placed at random according to a spatial Poisson process. We fix the modulation rate of all nodes to 11 Mbps and send packets of maximum length for 802.11b, i.e. 2 kB or 1610μs. For each instant the station starts a transmission, we record the time until its next hidden node starts to transmit. Figure 2.5 is an example cumulative histogram of these values for a particular station.

The significance of Figure 2.5 is that the duration of the packet sent by the station in real time, l, can be used as the value on the horizontal axis in order to look up P_{SC1} for the station on the vertical axis from the shown empirical curve. In particular, P_{SC1} is the probability that the next packet sent by any of the station's hidden nodes arrives before time l. Our overall approach to computing P_{SC1} is for the station to use the busy-idle signals at the station and AP to estimate this curve, and then to look up P_{SC1} based on its packet length. Specifically, we model this curve as a two-piece piecewise linear curve and estimate the slopes using the busy-idle signals.

For a single hidden node, B, the slope of the curve in Figure 2.5 can be estimated as the derivative of P_{SC1} with respect to l in Equation (2.11),

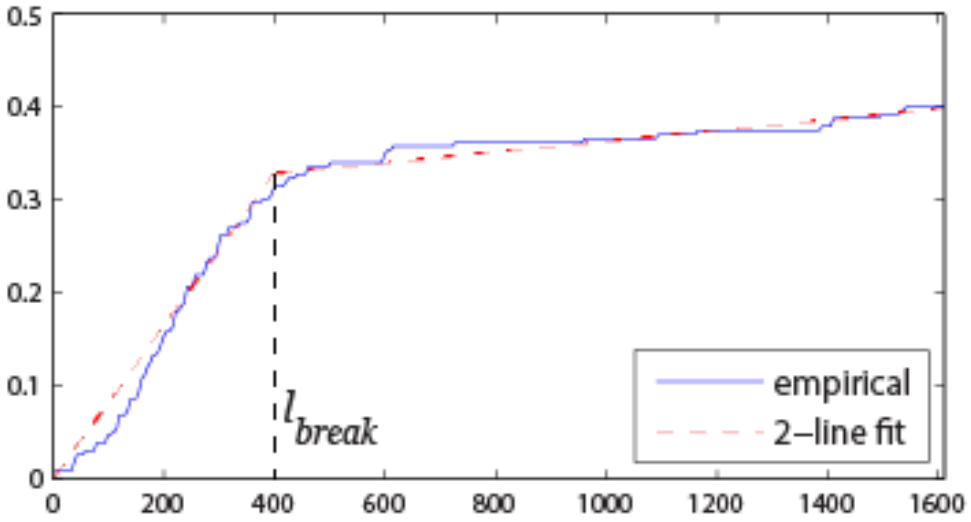


Figure 2.5. Plot of the cumulative histogram of the real time, l, in μs between the start of the station's packet and the start of the next packet sent by any of its hidden nodes.

$$\frac{\partial}{\partial l}P_{SC1}^{(B)} = -[\ln(1-\tau_B^*)](1-\tau_B^*)^{L^*{}_B(l)}\frac{\partial L_B(l)}{\partial l}. \tag{2.12}$$

The key issue is that the rate at which virtual time elapses with respect to real time l, i.e. $\frac{\partial L_B(l)}{\partial l}$, is a not constant. This is because of the nature of the virtual slot lengths. In essence, virtual time can be thought of as consisting of many consecutive short, idle, slots together with isolated long, busy, slots.

It is useful to think of $\frac{\partial L_B(l)}{\partial l}$ as a random function of l. To compute the expectation of $\frac{\partial L_B(l)}{\partial l}$, we split the realizations into two cases, depending on the state of the channel as observed by node B when the station begins its transmission:

1. If the station begins its transmission when node B observes the channel as idle, then for small l, $\frac{\partial L_B(l)}{\partial l}$ is relatively large, namely $1/(length\ of\ backoff\ slot)$. This is because node B is observing idle backoff slots and has $L^*{}_B(l) = 1$ opportunity to transmit per $l = length\ of\ backoff\ slot$ seconds. For larger l, eventually a silencing node, such as G in Fig. 2.4, starts to transmit, decreasing $\frac{\partial L_B(l)}{\partial l}$ to $1/(length\ of\ average\ busy\ slot)$. This is because for node B only $L^*{}_B(l) = 1$ virtual slot elapses during $l = length\ of\ busy\ slot$ seconds. Let l_{break}, shown in Fig. 2.5 denote the expected time at which this decrease occurs, i.e. the expected amount of time before a node similar to G in Fig. 2.4 starts sending when the station, A, is sending.

2. If the station begins its transmission when node B observes the channel as busy, then it is equally likely that the station has started its transmission at any time during the long,

busy slot. In this case, ${L^*}_B$ is initially constant, and eventually increases rapidly when the current busy slot expires and idle slots resume. However, since this increase is equally likely to happen at any time, averaging over all of these cases yields the same average value $\frac{\partial L_B(l)}{\partial l}$ for all l.

Combining the above two cases, we conclude that the random function $\frac{\partial L_B(l)}{\partial l}$ has a higher expected value for small l than for large l. The actual value of this expectation depends on how often the channel is busy on average at the hidden node B, because this gives the weighting of the two cases as well as the actual value of each. A reasonable proxy for how often the channel is busy at the hidden node B is the proportion of time the busy-idle signal at the AP is busy, denoted by $P(B_{AP})$. This quantity can be readily computed at the station, as the mean of BI_{AP}.

The empirical curve of Figure 2.5 shows the aggregate effect of this $\frac{\partial L^*(l)}{\partial l}$ for all hidden nodes for the station. Specifically, taking all hidden nodes into consideration, Equation (2.12) can be approximated as

$$\frac{\partial}{\partial l} P_{SC1} \approx -[\ln(1-\tau_h^*)](1-\tau_h^*)^{L^*(l)} \frac{\partial L(l)}{\partial l} \tag{2.13}$$

where $L^*(l)$ denotes the average length of station's packet in virtual slots as observed by all the hidden nodes. Similar to $\frac{\partial L_B(l)}{\partial l}$, we expect $\frac{\partial L(l)}{\partial l}$ and hence $\frac{\partial}{\partial l} P_{SC1}$ to have a higher expected value for small l than for large l. This is in agreement with the shape of the empirical curve in Figure 2.5.

Based on the above analysis, which predicts $\frac{\partial L(l)}{\partial l}$ to have a higher value for small l than for large l, it is reasonable to approximate the empirical curve in Figure 2.5 with a 2-piece piecewise linear function, with a steeper initial slope for $l < l_{break}$, and a less steep slope for $l > l_{break}$. Via our NS-2 simulations, we have empirically found $l_{break} = 400\mu$s to result in fairly accurate estimates of the curve in Figure 2.5 for our 7 AP, 50 node Poisson topology. For less uniform topologies, such as those with clusters of nodes, this break point will be different. Future work involves investigating ways for the station to estimate the break point based on local and AP statistics.

Having fixed l_{break}, the piecewise linear function in Figure 2.5 is completely determined by 2 parameters: the slope for $0 < l < l_{break}$, denoted by m_1, and the slope for $l_{break} < l < l_{max}$, denoted by m_2. To estimate P_{SC1}, the station must somehow estimate these slopes from the busy-idle signals available to the station. Obtaining a reasonable approximation for the entire curve, rather than for one value of P_{SC1}, has the added advantage of allowing the station to determine the sensitivity of P_{SC1} with respect to packet length from the slope of the curve in Figure 2.5. In general, the smaller the slope is at the current packet length, the less sensitive P_{SC1} is to packet length.

Rather than estimating m_2, we opt to estimate m_{avg}, the average slope over the entire range of l, as

$$m_{avg} = \frac{(m_1 l_{break} + m_2 l_{max} - l_{break})}{l_{max}} \tag{2.14}$$

where l_{max} is the length of the longest possible packet the station can send, i.e. 2KB in our example. m_2 can easily be computed from m_1 and m_{avg}.

We have empirically found a reasonable model for the initial slope to be

$$m_1 = \alpha_0(P(B_{AP})) - \alpha_1(P(B_{AP})) \ln(1 - \tau_h^*) \tag{2.15}$$

and for the average slope to be

$$m_{avg} = \beta_0(P(B_{AP})) - \beta_1(P(B_{AP})) \ln(1 - \tau_h^*). \tag{2.16}$$

α_0, α_1, β_0, and β_1 are functions of $P(B_{AP})$, and are looked up from a table by the station. The first terms in Eqs. (2.15) and (2.16), i.e. α_0 and β_0, are independent of τ_h^*, are absent in Equation (2.13), and are discussed in detail in Section 2.2.3.3.

From Equation (2.13) and Figure 2.5, α_1 and β_1 are the average values of $(1-\tau_h^*)^{L^*(l)} \frac{\partial L^*(l)}{\partial l}$ for $l < l_{break}$ and $l < l_{max}$, respectively. As noted earlier, $\frac{\partial L(l)}{\partial l}$ depends on $P(B_{AP})$. Similarly, it can be argued that $(1 - \tau_h^*)^{L^*(l)}$ depends on $P(B_{AP})$. This is because τ_h^* is the average sending rate of the hidden nodes, and $L^*(l)$ depends on the sending rate of the hidden node neighbors. Putting these together, we conclude that the quantity $(1 - \tau_h^*)^{L^*(l)} \frac{\partial L(l)}{\partial l}$, and thus α_1 and β_1 are also dependent on $P(B_{AP})$.

When the average network traffic, $P(B_{AP})$, is low, nearly all hidden nodes experience idle channel conditions when the station begins to send. Thus, $\frac{\partial L_B(l)}{\partial l} = (1\ virtual\ slot)/(length\ of\ idle\ slot)$ for small l. In this case, it is also likely that τ_h^* is small, so $(1 - \tau_h^*)^{L^*(l)} = 1$ for small l. Therefore, for small $P(B_{AP})$, α_1, which corresponds to small l, i.e. $l < l_{break}$, is given by $(1\ virtual\ slot)/(length\ of\ backoff\ slot)$. As $P(B_{AP})$ increases, there are more long, busy slots, causing both $(1 - \tau_h^*)^{L^*(l)}$ and $\frac{\partial L(l)}{\partial l}$, and thus α_1 to decrease. Similarly, β_1 decreases with increasing $P(B_{AP})$ due to a similar phenomenon. These observations are verified shortly via simulations shown in Table 2.2 of Section 2.2.3.4.

2.2.3.3 Coupling of transmission times

We now justify the existence of α_0 and β_0 in Eqs. (2.15) and (2.16). They are related to a phenomenon which we call the coupling of transmission times. When an intermediate node such as node C or D in Figure 2.3 is sending, it silences both nodes A and B. As a result, while nodes A and B would have transmitted independently in the absence of intermediate nodes, in the presence of such a node, both of their available transmission times are reduced to times when nodes C and D are silent. This increases the number of staggered collisions between

nodes A and B because they are more likely to send around the same time. The amount by which this increases P_{SC1} is dependent on how often nodes such as C or D are sending. Assuming fairly symmetric behavior by all nodes, $P(B_{AP})$ should be a reasonable proxy for this in our uniform random topology. For lower-traffic networks, this effect is negligible, but for higher traffic networks with larger values of $P(B_{AP})$ the effect becomes more pronounced. Thus α_0 and β_0 should increase with $P(B_{AP})$ as verified shortly via simulations shown in Table 2.2 of Section 2.2.3.4.

2.2.3.4 Estimating α_0, α_1, β_0, and β_1

In general, computing closed-form analytical expressions for α_0, α_1, β_0, and β_1 as functions of $P(B_{AP})$ is a non-trivial task. Instead, we pre-compute them for various ranges of $P(B_{AP})$ via regression from simulation data using the the NS-2 simulation package, which we have modified to collect the busy-idle signal at the AP and nodes as well as record the fate of each packet – DC, SC1, SC2, or no collision. We use the topology described in Section 2.2.1, with all nodes sending Poisson application layer traffic at a fixed rate, which varies over different simulations. We use 13 different rates, ranging from 2 kB/s to 120 kB/s, and repeat each rate 5 times with different random topologies. From each of the 65 trials, we collect the measured data, namely the busy-idle waveform, at the center AP and all the nodes in its cell. We also record the empirical cumulative histograms for each station, i.e. the solid curve in Figure 2.5 to use as ground-truth. We then choose a random subset of the over 400 stations from the 65 trials to be used for training data. We optimize m_1 and m_2 for each station to achieve the minimum squared distance between the 2-line approximation and the empirical curve. For each of the training stations, we use the optimal m_1 and m_2, as well as the local estimates of $\ln(1-\tau_h^*)$ to perform a linear regression using Eqs. (2.15) and (2.16) in order to determine the values of α_0, α_1, β_0, and β_1. The resulting values are shown in Table 2.2. As expected, α_1 and β_1 decrease with $P(B_{AP})$ while α_0 and β_0 increase with it. Also, for $P(B_AP) < 0.9$, $\alpha_1 = 1/(20\mu s) = (1\ virtual\ slot)/(length\ of\ backoff\ slot)$. Since $\ln(1-\tau_h^*)$ is on the order of 10^{-2}, all the terms in Eqs. (2.15) and (2.16) are of comparable size. To estimate P_{SC1}, a station observes $P(B_{AP})$ from the AP's BI signal, and looks up values for α_0, α_1, β_0, and β_1 from a pre-stored table.

Figure 2.6 shows the resulting average normalized error in the estimate of the curve in Figure 2.5, defined as $|y^{estimate} - y^{empirical})/y^{empirical}|$, for two bins of values of $P(BI_{AP})$, namely 0.6 to 0.7 and 0.92 to 0.94. The averaging is done over multiple stations in multiple topologies, as described above, with similar mean values of BI_{AP}. As seen, the estimation accuracy improves with packet length. The performance for lengths shorter than 200μs is unimportant, since this is below the minimum packet length including inter-frame spacing for 802.11b. For longer packets, errors become less than 10% as the effects of inaccurate estimates of L_{break} decrease. This increased accuracy at longer packet lengths is desirable as higher traffic applications and newer standards tend to use longer packets.

Table 2.2. Sample values of α's and β's

range of $P(B_{AP})$	α_0	α_1	β_0	β_1
0-.3	0	5e-2	0	2.48e-2
.3-.5	0	5e-2	0	2.43e-2
.5-.6	0	5e-2	0	2.38e-2
.6-.7	0	5e-2	0	1.96e-2
.7-.75	0	5e-2	0	1.93e-2
.75-.8	0	5e-2	0	1.91e-2
.8-.85	0	5e-2	0	1.78e-2
.85-.88	1e-4	5e-2	1e-4	1.29e-2
.88-.9	1e-4	5e-2	1e-4	1.30e-2
.9-.92	2e-4	3.96e-2	2e-4	9.48e-3
.92-.94	3e-4	3.23e-2	2e-4	8.10e-3
.94-.96	6e-4	2.41e-2	3e-4	4.75e-3
.96-1	9e-4	2.00e-2	4e-4	2.17e-3

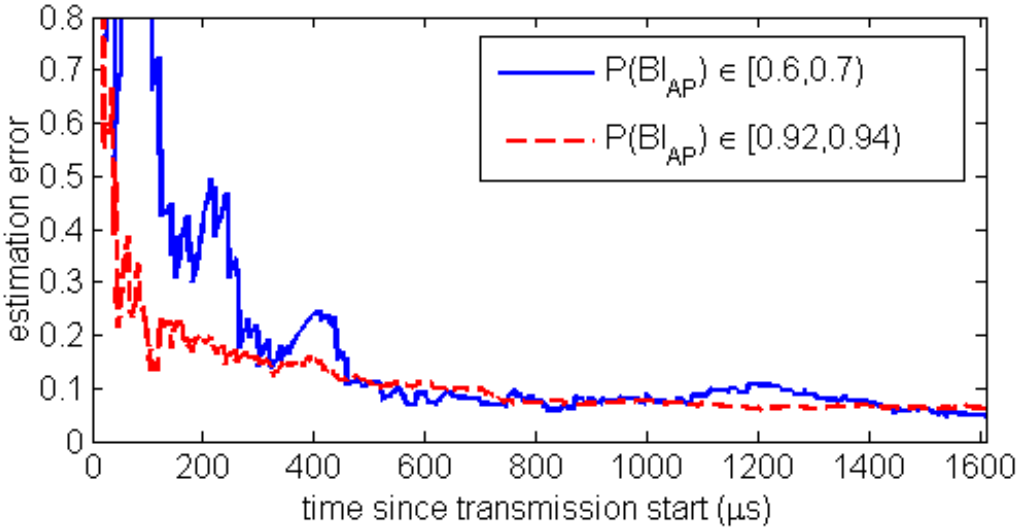


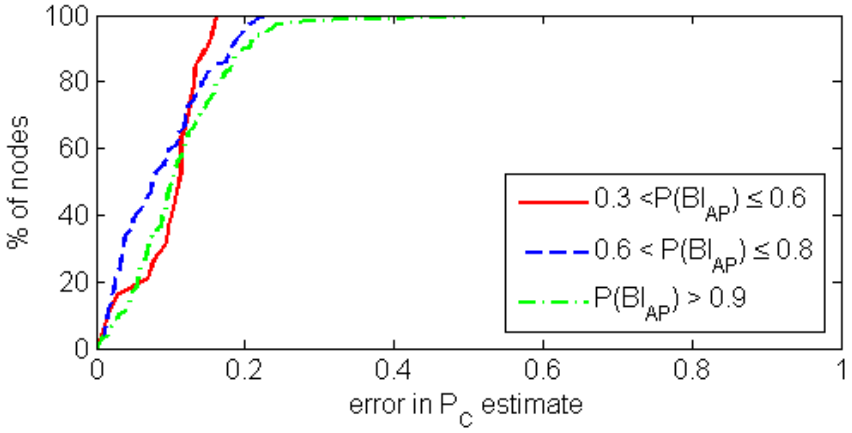
Figure 2.6. Average absolute value of normalized estimation error, $|(P_{SC1}^{estimate} - P_{SC1}^{actual})/P_{SC1}^{actual}|$, for two traffic levels.

Having estimated the probabilities of each type of collision, we can now combine them to find the total P_C using Equation (2.1).

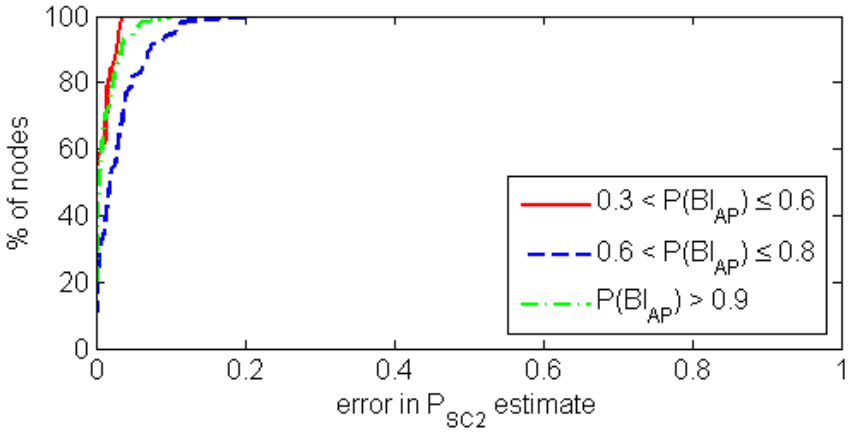
2.3 Results

To verify our analysis, we use the NS-2 simulation package, which we have modified as described above. We post-process the busy-idle signals generated by NS in MATLAB to arrive at estimates of the various collision probabilities. Our topology consists of 7 APs covering hexagonal cells, with 50 nodes placed randomly over the covered area via spatial Poisson process. Traffic is generated via a Poisson process at each node. 13 different rates are used for each of 5 different topologies, differing from those used in Section 2.2.3.4. Each run lasts 30 seconds. The estimates are computed using the busy-idle signals from the first 10 seconds, while ground truth is obtained by empirically counting the number of packets incurring each fate over the full 30 seconds.

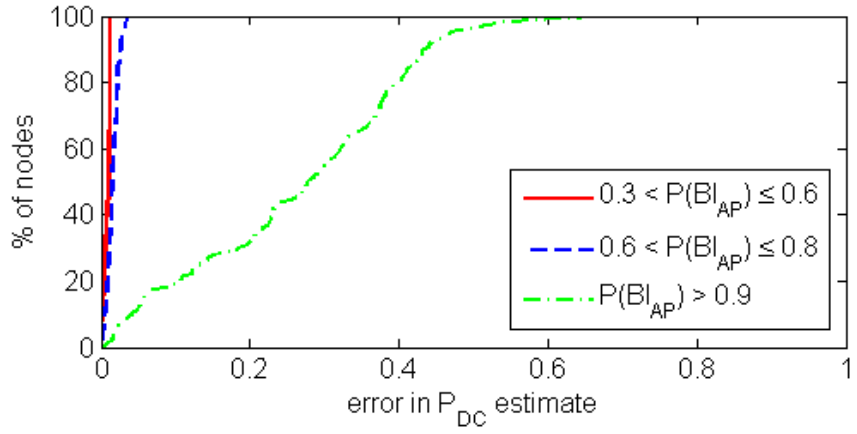
Figure 2.7(a) shows a cumulative histogram of the absolute error in estimating P_C for three different values of $P(B_{AP})$. Data for other traffic levels is not shown, but performance is similar. Figs. 2.7(b)-(d) show the error in each component of P_C for the same three traffic levels. As seen, P_{SC2} has a consistently small error. Due the way collisions are counted, SC2s are the dominant type of collision for higher traffic scenarios, making precise estimation of P_{DC} and P_{SC1} more difficult and less important as mean(BI_{AP}) increases. In particular, for $P(BI_{AP}) > 0.9$, P_{SC2} is almost 1, and there is not a sufficient number of packets which are not classified as SC2s for even the empirical count of P_{DC} or P_{SC1} used as ground truth to be accurate. It can be seen that while the error in estimating P_{DC} and P_{SC1} grows as traffic increases, the total error in P_C does not. In general, 90% of the time, the P_C estimate is within 20%, and 50% of the time, it is within 10%.


100
80
60
40
20
0
% of nodes
0
0.2
0.4
0.6
0.8
1
error in P_C estimate
0.3 <P(BI_AP) ≤ 0.6
0.6 < P(BI_AP) ≤ 0.8
P(BI_AP) > 0.9

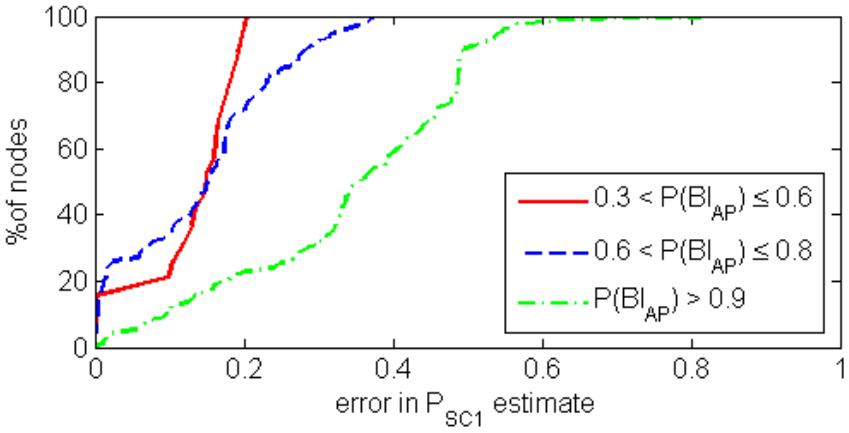
(a)


100
80
60
40
20
0
% of nodes
0
0.2
0.4
0.6
0.8
1
error in P_SC2 estimate
0.3 < P(BI_AP) ≤ 0.6
0.6 < P(BI_AP) ≤ 0.8
P(BI_AP) > 0.9

(b)



(c)



(d)

Figure 2.7. Cumulative histogram of the estimation errors in (a)P_C, (b)P_{SC2}, (c)P_{DC}, and (d)P_{SC1} for three traffic levels.

Chapter 3

Experimental Verification

In the previous chapter we developed a method whereby the BI signal of the AP is used in conjunction with local information, namely the BI and TX signals of the station, in order to locally estimate the probabilities of the 3 types of collisions. We also demonstrated the accuracy of our proposed estimation method via NS-2 simulations. In this chapter, we experimentally verify the feasibility and accuracy of this collision probability estimation technique. This centers around generating the BI and TX signals at a resolution on the order of 10μs over a 5 second period.

The BI and TX signals are collected at every node and AP in the network, and need to be temporally aligned for collision probability estimation. In order to implement a full system, busy-idle signals would need to be computed and broadcast in real time; however, without access to much of the lower level functionality of the wireless cards, this is not practical for our experiments, which use only off-the-shelf hardware. Thus, we opt to collect the data in real time, and process it offline using MATLAB[27] in order to verify the accuracy of the resulting estimates.

Collection of the BI data is possible using Ath5k [1], which is an open source driver for wireless cards with certain Atheros chipsets. This driver allows a user to access hardware registers in the wireless card. While the BI signal is not explicitly computed and stored, we show in this chapter that it can be inferred from data stored in accessible registers. We collect experimental data using the Ath5k driver and process it offline to generate BI signals and collision probability estimates for a variety of network conditions.

The computation of collision probability estimates consists of 4 steps:

1. Collect available carrier sense data from wireless card
2. Process this data to generate BI and TX signals

3. Align BI and TX signals of station and AP

4. Compute estimates.

Each of the 4 steps are outlined in the following subsections.

3.1 Collecting carrier sense data

All stations and APs in our experiments are laptops running Ubuntu Linux version 9.10, Kernel version 2.6.31-22. We use the Ath5k Open Source wireless card driver, which we have modified to provide the functionality to generate the BI signals. We use external Proxim Wireless Cards Model: 8470-FC, which utilize the AR5212 Atheros chipset.

While the actual carrier sensing mechanism is handled in hardware, there are few registers on the wireless card, called the "profile count" registers, which can be accessed using Ath5k; these registers store statistics about recent channel activity. The AR5K_PROFCNT_CYCLE register behaves like a clock, incrementing each clock cycle. The AR5K_PROFCNT_TX register increases whenever the card is transmitting packets. The AR5K_PROFCNT_RXCLR register increases when the energy detected on the channel measures above the carrier sense threshold. In the remainder of the chapter, we omit the "AR5K_PROFCNT_" prefix for simplicity.

We have carried out a number of experiments to verify the behavior of these registers. Verification is done with 2 laptops, nodes A and B, which use the Ath5k driver, and a standard AP. The two nodes are placed next to each other while node A transfers a file to the AP. Node B is close enough to overhear the traffic, but does not contribute any significant traffic. Profile count register values are collected at both nodes. Examining the register values, we see that while node A is transmitting, both the TX and the RXCLR registers of node A increase linearly. Because node B hears the same traffic as node A, its RXCLR register increases at the same rate as that of node A; however, because it is not transmitting any traffic, its TX register does not increase significantly. [1] In the next experiment, a single node is placed next to a microwave oven, and no traffic is sent. In this scenario, the RXCLR register increases at a greater rate when the microwave is turned on as compared to when it is off. This is because the microwave operates around the same frequency as the network card. This demonstrates that even interference that cannot be decoded as packets affects the RXCLR register.

These experiments confirm that the RXCLR register increases when the channel is occupied and remains constant when it is idle, while the TX register increases while the node is transmitting and remains constant otherwise. This means that the derivative of the RXCLR

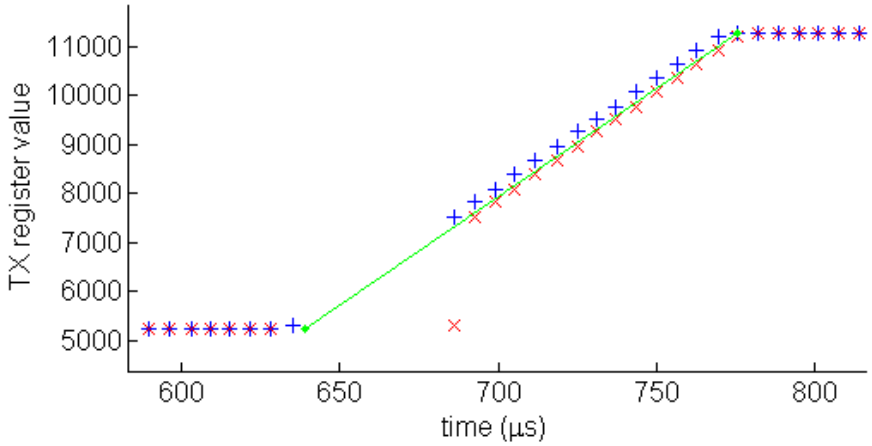
[1] While these increases appear linear at a large time scale, when they are observed at a small time scale, it becomes apparent that the increases occur in small bursts with each packet.

register with respect to the CYCLE register is the desired BI signal, and the derivative of TX with respect to CYCLE is the desired TX signal.

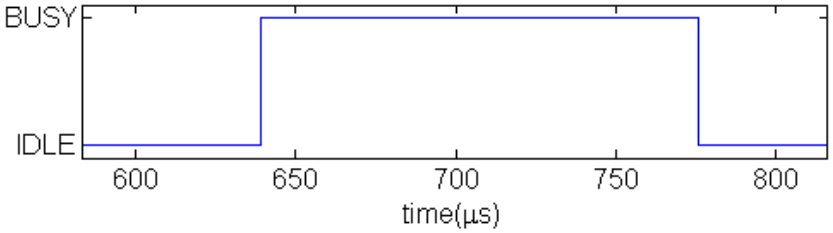
If it were possible to observe the register values in real-time, it would be trivial to generate the BI and TX signals. However, we can only access the register values via sending a read request to the card, which is not necessarily served in a real time fashion. Additionally, we cannot read multiple registers simultaneously, resulting in uncertainty about the precise times at which each register value is increasing. However, the CYCLE register value increases at a constant rate of 1, and the other two registers are either increasing at the same rate, or remaining constant, switching between these two behaviors only at the start and end of packets. We can use this observation to approximate the desired busy-idle signal.

We also have to combat a few other practical issues with regard to the register behavior. There are times when the read requests are not served for a long period of time due to the non-real time nature of the operating system. Since this creates excessive ambiguity in the BI signals, we omit any sections with longer than 1ms delay between samples from collision probability estimation. Another issue is that these registers are currently also used for an adaptive noise floor computation module in the Ath5k driver. This module occasionally resets the registers, interfering with the BI estimation process; thus, we disable it during our experiments. In actual implementation of our approach within a wireless card in the future, we would need dedicated registers for BI computation or coordination with the noise floor computation module.

In order to generate the BI and TX signals, we need to plot RXCLR and TX vs CYCLE; however, since we can only sample one register value at a time, it is not possible to obtain exact points on this plot. Rather, we cyclically sample all three registers in the following order: CYCLE, TX, RXCLR. For each y-value of RXCLR or TX, we can bound the corresponding x-value for CYCLE, i.e. time, to be between the previously sampled CYCLE value, and the subsequently sampled CYCLE value. An example of a resulting plot is shown in Figure 3.1(a), where for each value of TX on the y-axis, a blue '+' represents the lower bound of the corresponding x-value and a red 'x' represents the upper bound on the x-value. Thus, the true TX vs CYCLE curve must lie to the right of all blue '+'s and to the left of all red 'x's. As seen, in this example there is a large gap between the sample times at the start of the increasing slope; so it is possible to misinterpret the data to conclude that the value of the TX register does not begin to increase until much later. For this reason, care must be taken in interpolating the BI signal from these points, as described in Section 3.2



(a)



(b)

Figure 3.1. *(a) TX vs CYCLE. For each value of TX, the value of CYCLE is bounded by the x-position of the blue '+' and following red 'x'. The interpolated actual value is shown as the green line; (b) the resulting estimated BI signal.*

3.2 Generating the BI signal

Once we obtain bounds for the position of the RXCLR vs CYCLE curve, we must use the known constraints about the register behaviors in order to interpolate the actual curve whose derivative is the desired BI signal.

Since the RXLCR and TX vs CYCLE curves alternate between having slopes 0 and 1, they can be uniquely identified by the points at which they switch between these two slopes. Our approach to generating the BI signal is to identify the start and end point of each increasing section, which we call *busy sections*, since they correspond to times when the channel is busy. An example busy section is denoted by the green line segment in Figure 3.1(a), and the resulting BI signal is shown in Figure 3.1(b). Figure 3.2 shows a flow diagram of our proposed process to identify these busy sections from the raw data.

The first step is to identify sections of the data that could potentially be busy sections.

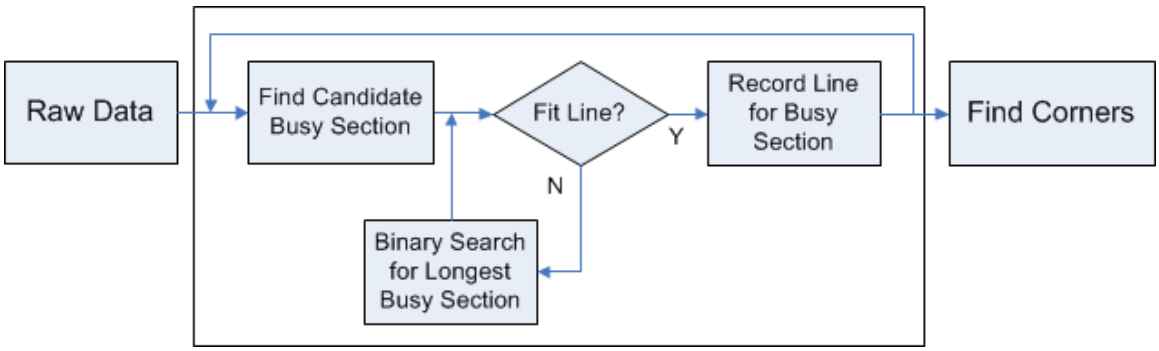


Figure 3.2. *Flow diagram of BI signal generation process.*

This is done by selecting the largest possible consecutive set of y-values which are strictly increasing. We denote this vector of y values by $\vec{y}$. We denote the vector of previously and subsequently sampled CYCLE values by $\vec{x_1}$ and $\vec{x_2}$, respectively. We then determine if a line of slope close to 1 can be fit to the right of all lower bounds and to the left of all upper bounds. If such a line exists, it is identified as a busy section, which is represented by a line $y = m(x - a) + b$.To find such a line, we solve the following optimization problem:

$$\begin{aligned} \min_{m,a} \quad & |m - 1| \\ s.t. \quad & (\vec{y} - b \cdot \vec{1})/m > \vec{x}_1 - a \cdot \vec{1} \\ & (\vec{y} - b \cdot \vec{1})/m < \vec{x}_2 - a \cdot \vec{1} \\ & 0.95 < m < 1.05 \end{aligned} \tag{3.1}$$

where $\vec{1}$ is the all-ones vector, m is the slope, and (a, b) is a point on the line corresponding to the busy section where b is chosen as the y-value of the last pair of previous consecutive samples with equal y-values, as shown in Figure 3.1(a). Since there can be some random imprecision due to clock jitter, we allow m to differ from the ideal value of 1 by up to 5%.

If there is a feasible solution to (3.1), then the resulting line is stored as representing a busy section. The green line of Figure 3.1(a) is such a solution, where $m = 1$, and the point (a, b) is the leftmost point of the line segment.

If there is no solution, the vectors $\vec{y}$, $\vec{x}_1$, and $\vec{x}_2$ must be adjusted so that there is a solution. To accomplish this, we perform a binary search for the maximal length section for which there is a solution. Once a line has been fit to the maximum number of RXCLR or TX samples, the line is stored and the points are removed from consideration. The algorithm then continues on the remaining points until all RXCLR or TX samples are processed

Once all of the busy sections have been accounted for, their end points, which are the transition points of the BI signal, can be identified by finding the intersection of the set of representative lines with the set of horizontal lines, $y = b$, which are trivially fit to the surrounding idle sections. [2]

[2]Additional minor approximations are done in cases where busy or idle sections contain fewer than 2 data points; the details have been omitted for brevity.

Once the corners between lines of slope 0 and 1 have been found, the RXCLR vs CYCLE curve is complete and the BI signal can be generated. The TX signal is generated via the same method, using TX rather than RXCLR.

3.3 Aligning station and AP signals

In a real-time implementation, the AP would periodically broadcast its BI signal to its associated stations to enable the stations to perform computations to estimate the collision probability. To do so, the AP's BI signal and the station's BI and TX signals must be temporally aligned. Note that it is not necessary for nodes to achieve clock synchronization to the level of 20μs time slots. In practice, nodes merely need approximate time synchronization of around the size of a packet, because more precise alignment can be obtained using the signals themselves, as there are known correlations between the signals. In our current setup, we do not have any clock synchronization because we cannot control the CYCLE register; therefore the alignment problem is actually more involved for our setup than it would be in a complete implementation by a card manufacturer. Still we are able to achieve reasonably good results, as shown later, at the cost of greater amount of computation.

Since the BI and TX signals for a single node are collected with respect to the same clock, they are already aligned; thus aligning all of the signals can be accomplished by aligning either of the station's signals with either of the AP's signals. Even though the AP's TX signal is not used in the estimation process, it may be used for alignment purposes.

Alignment consists of scaling and shifting the AP's signals to correspond to the station's signals. Due to a mismatch in clock speeds of the wireless cards, the signals must be scaled before they can be aligned. We have experimentally verified that the scaling factor does not significantly change for sets of data collected in a single hour-long data collection period, so in practice the scale factor can be computed as infrequently as once per hour. Once the AP's signal has been scaled, it must be shifted to properly align the starts of packets with those in the station's signal.

We compute the appropriate scaling by using data from a simple scenario in which we successfully transmit 2000 packets from a station to the AP over a 20 second period. We then generate the resulting BI signals and align the start of the first packet. We plot the difference in the start times of each packet in Figure 3.3. From this plot we can clearly determine the slope and required scaling, which in this case is around 0.003%.

Once the scaling is done, we need to compute the offset between the AP's signals and the stations signals. Depending on the scenario in which the data is collected, we perform the shifting using one of two methods:

1. Align AP's BI to station TX

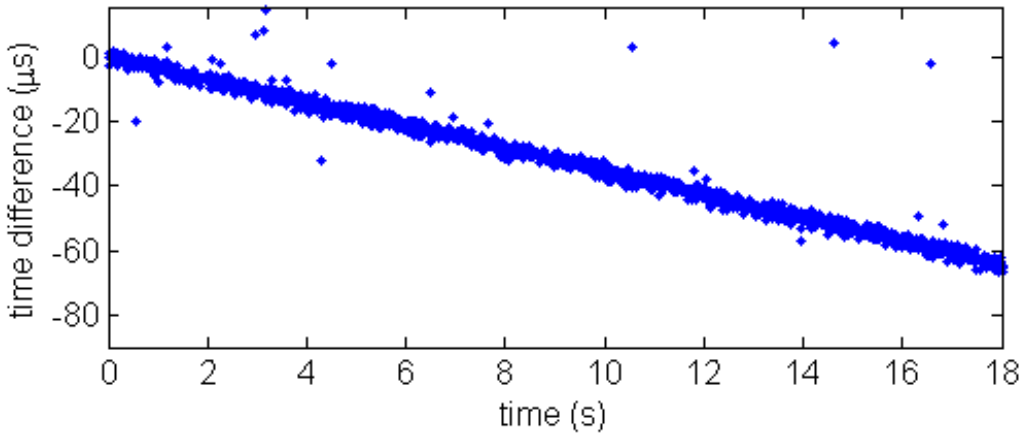


Figure 3.3. *Difference between packet start times in the AP's BI signal versus the station's BI signal over time.*

2. Align AP's TX to station TX.

In the first method, we exploit the fact that the station's TX signal should be a subset of the AP's BI signal if the AP is capable of sensing all of the stations packets; this is typically the case barring a deep fade. The alignment is accomplished by using the first few milliseconds of the station's TX signal, including the first 5 transmitted packets. The offset is chosen to maximize the overlap with the APs BI signal. This method works best if most of the packets are correctly received, because the pattern of packets should look identical in the AP's BI and station's TX signals.

The second method relies on the fact that the AP transmits ACKs immediately after it successfully receives packets from the station. We can thus take a set of ACK packets identified from the AP's TX signal and find the corresponding packets in the station's TX signal. We do this by examining the sequence of inter-arrival times for ACKs observed in the AP's TX signal, and searching for a set of packets in the station's TX signal with the same set of inter-arrival times. The packets need not be consecutive as some packets may fail and go unacknowledged. An example of this is shown in Figure 3.4. The top row shows the packets sent by the station and the bottom shows the packets sent by the AP. The start of each packet is represented by a 'o', and the end is represented as a 'x'. They are colored according to the type of packet they correspond to, which can be determined by length. The long blue packets are data, the short red packets are ACKs, and the green packet is a management packet. Since the ACKs are so short, the 'x' appears to overlap the 'o' in this figure. As seen, the ACKs transmitted by the AP occur immediately after the data packets transmitted by the station, with the same inter-packet time. This method is more robust to collisions and poor channel conditions than the first one, because it does not rely on consecutive packets being received correctly. Therefore, we use this second method to compute the offset between the AP's and station's signals in scenarios with higher loss.

Note that we do not use the station's BI signal for alignment, because its relationship with the AP's signals is not as straightforward due to the potential existence of exposed nodes.

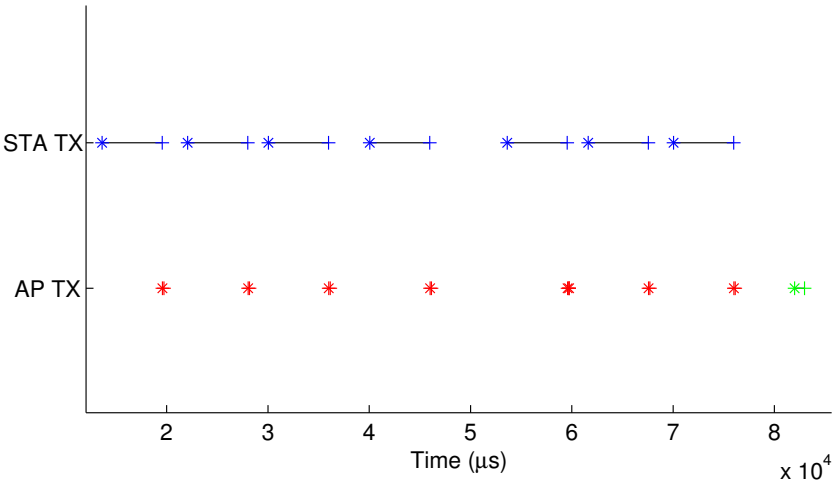


Figure 3.4. *Alignment of TX signals of station (top) and AP (bottom). Blue line segments correspond to data packets, red to ACKs, and green to management packets.*

After the alignment, the start of most packets common to the station's and AP's BI signals are within 40 μs of one another. Figure 3.5 shows a histogram of the difference in start times of all the packets, including management packets, in the scenario where 2000 packets are sent without errors after appropriate scaling and shifting. This alignment process must be done periodically because the clock drift may change over time, but consecutive broadcasts can use the same alignment computations.

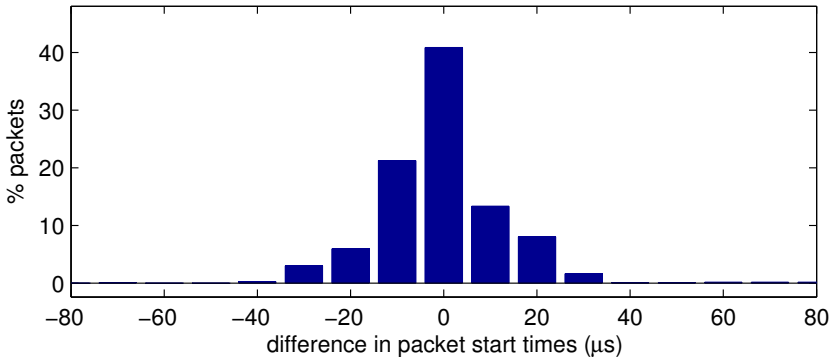


Figure 3.5. *Difference in start times between corresponding packets in station's and AP's BI signals for experiment with 2000 successful packets.*

3.4 Computing collision probability estimates

Once the signals are aligned, we estimate the probability of collision for each scenario via the method described in Chapter 2. Since we do not have perfect resolution, reconstruction, or alignment between the signals, we allow for edges as far as 40 μs apart to be classified as simultaneous. This must be done to avoid a station from mistaking a slightly misaligned packet for an indication that a hidden node exists and is generating these packets slightly offset from those observed by the station. As seen in Figure 3.5, most corresponding packets are within this range of each other.

3.5 Results

To verify the effectiveness and accuracy of our approach, we set up a simple experiment with a station sending traffic to an AP, and another distant node pair hidden to the station causing staggered collisions. The topology is shown in Figure 3.6, and the parameters are listed in Table 3.1. In this experiment, we can only verify the overall P_C estimate, not the components, since it is not possible to identify the individual collisions for the ground truth.

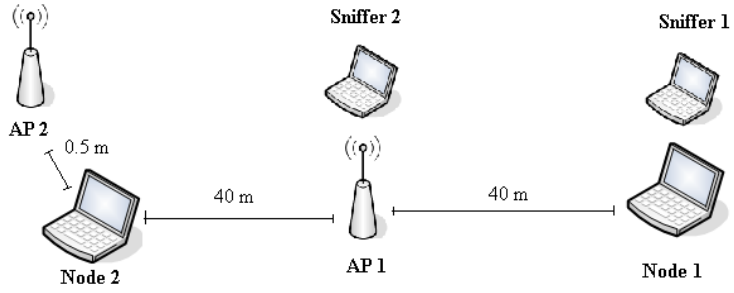


Figure 3.6. *The topology of the experiments. Node 2 is hidden from Node 1.*

The experiment topology is made up of 2 networks, each consisting of 1 access point and 1 node. Node 1 sends 2000 packet of 1400 Bytes each to AP1 over 15 seconds. Collision probabilities are estimated for this node. Node 2 sends varying amounts of traffic to AP2 to cause collisions with packets of Node 1 at AP 1. Traffic is generated using Iperf [2]. The Iperf software allows us to generate UDP traffic with fixed packet size and average inter-packet time at the application layer, but cannot be used to send a precise pattern of packets. Additionally, since failed packets are re-transmitted, the actual number of transmitted MAC layer packets may vary.

To count the actual number of transmitted MAC layer packets, 2 laptops run Kismet[3] to sniff the generated traffic, while not contributing to the traffic on the network. The sniffed traffic is used to generate ground truth in each experiment.

Table 3.1. Experimental Parameter Settings

Parameter	Value
Protocol	802.11b
Mode	Infrastructure
Channel	1
Modulation Rate	2Mbps
Transmit Power	27 or 15 dBm
Foreground Packet Length	1400 B
Background Packet Length	1300 B

We modify 2 parameters over our various experiments:

1. The rate of packet transmission for Node 2, which affects collision probability. Specifically, it takes on 5 different levels, resulting in collision probabilities ranging from 0 to 55%.

2. The transmission power of Node 1, which affects channel error probability. Specifically, it takes on 2 different values resulting in channel error probabilities of less than 1% and 20-40%, respectively. Due to variations in the environment, it is difficult to achieve a consistent empirical loss probability over 2000 packets; nevertheless, we show that our estimates are largely unaffected by this varying channel error probability.

The traffic generated by Node 1 is kept approximately constant at 100 UDP packets per second for each scenario, while the traffic of Node 2 is varied to achieve 5 different collision probabilities. For each collision probability, we run 5 experiments of 20 second duration, 2 at 27dBm transmit power to avoid channel errors and 3 at 15dBm to allow for channel errors. We use the sniffer data from the high power experiments to obtain the ground truth collision probability. We then use the technique described in this chapter to generate the BI signal and estimate collision probability over 15 different 5-second intervals of the data, 3 from each experiment.

We verify that Nodes 1 and 2 cannot sense each other with respect to their physical carrier sensing functionality. This is done by observing the RXCLR signal of each node when the other node is sending traffic. We have verified that each node only senses power 5% of the time that the other hidden node is transmitting.

We also use the sniffer data to count the number of packets transmitted, to verify the correctness of the BI signal. To accomplish this, we run an experiment in a low loss scenario so that the sniffer would be able to detect and decode all packets. We then use the sniffer data to generate a predicted BI signal by inserting busy sections of appropriate lengths at the times the packets were recorded by the sniffer and compare this to the BI signal generated using the approach in this chapter. We find that the number of data packets observed in the BI signal, defined as busy sections with length approximately equal to that of a data packet,

is within 1% of the number of data packets observed by the sniffer. The sniffer detects significantly more ACKs, because the ACKs are often combined with the data packets in the BI signals. This is due to the fact that the time interval between the data packet and ACK is short relative to the resolution of the BI signals. Similarly, some management packets are combined with one another accounting for the difference in total management packets. When we compare the actual BI signal with the predicted BI signal on a sample-by-sample basis at 10μs resolution, we observe that the signals are identical 97.3% of the time. 1.4% of the time, the predicted signal is busy while the actual BI signal is idle, and 1.3% of the time, the reverse is true. This can be attributed to minor misalignments in packets and slightly different sensing due to the fact that the station and sniffer are not perfectly co-located.

Figure 3.7 shows the resulting estimates of collision probability – each generated using 5 seconds of BI signal – versus ground truth empirical counts. Solid blue dots correspond to estimates for scenarios with high power and low channel error probability, while red 'x's correspond to estimates for scenarios with low power and high channel error probability. For each empirical value of collision probability, there are 15 points, including 6 blue and 9 red.

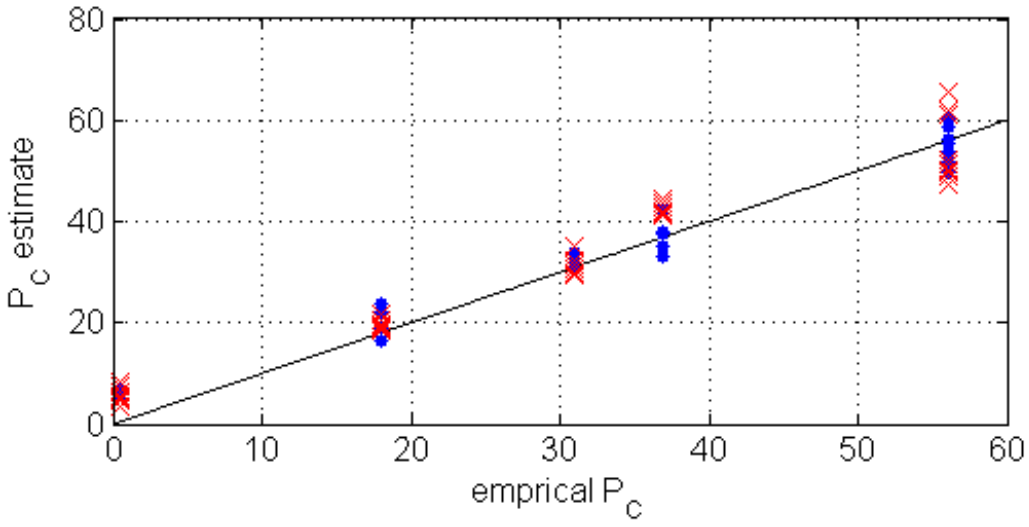


Figure 3.7. *Collision probability estimate vs empirical value.*

As seen, the estimates are distributed in the appropriate neighborhood. Even though they vary between each 5-second period, this is to be expected since over a 5 secondtime period, even the empirical error count is highly variable. The estimates are slightly high for the scenarios with no collisions due to imperfections in the BI signal generation; however, overall the estimates tend to be within 5% of the empirical count. The distribution of estimation errors is shown in Figure 3.8. As seen, the estimates are always within 10%, and within 5% over 70% of the time.

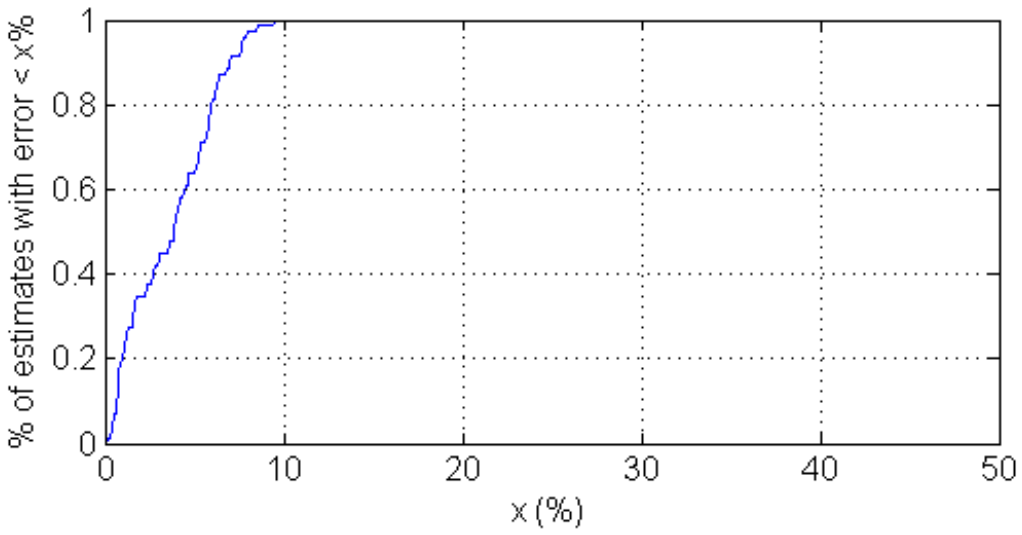


Figure 3.8. *Cumulative histogram of errors in collision probability estimates.*

3.6 Conclusion

In this chapter we have demonstrated an implementation of the BI signal generation and collision probability estimation technique using off-the-shelf hardware. Our results validate the approach. There are some imperfections within our data, which causes some variation in our estimates. However, if more of the process were to be implemented at a lower level by card manufacturers in hardware, the estimation accuracy should improve, since many of the obstacles we had to deal with could be avoided. Currently, all of the necessary information is already being collected by the wireless chip so it should be possible to store it and perform the necessary computations at the driver level. Such an implementation would be extremely valuable for experimental verification and potentially real implementation of the adaptation algorithms described in the remainder of this thesis.

Chapter 4

Improving Link Adaptation

The 802.11 standard includes several modulation rates, each of which is optimal for a different set of channel conditions. However, there are no simple and reliable methods for nodes to determine their current channel conditions. Existing link adaptation techniques use packet losses as an indication of poor channel conditions; however, when there is a significant probability of collision, this assumption fails, leading to degraded throughput. Specifically in a high collision scenario, typical link adaptation techniques based on ARF reduce the modulation rate to the lowest level regardless of channel conditions. In this chapter we describe two methods to leverage the P_C estimation technique described in Chapter 2 to improve link adaptation and thus increase throughput in an 802.11 network via two different approaches. First, we use the estimate in a modified version of ARF, based on [21]. Second, we propose a new LA algorithm, called SNRg, in which nodes estimate the channel conditions by comparing the empirical loss statistics to the expected loss statistics based on the estimated collision probability, and choose the optimal modulation rate for these conditions.

The chapter is organized as follows: Section 4.1 describes the packet loss model, the link adaptation algorithms are explained in Section 4.2, simulation results are presented in Section 4.3, and conclusions are discussed in Section 4.4.

4.1 Packet Loss Model

As described previously, losses in Wireless LANs can be classified into two types: collisions, which are the result of unfavorable traffic conditions, and channel errors, which are the result of unfavorable channel conditions. The total packet loss probability P_L can thus

be computed as

$$P_L = 1 - (1 - P_C)(1 - P_e) \tag{4.1}$$

where P_C is the probability of collision, and P_e is the probability of channel error, which is assumed to be independent of P_C. In this analysis, we assume that all collided packets are lost, not captured, and that the probability of ACK loss is negligible compared to other losses.

Knowledge of the probabilities of each of the components of P_C is useful for adaptation of parameters such as contention window, packet length, and carrier-sense threshold. However, for LA, nodes are primarily concerned with distinguishing channel-based losses, which are combated by LA, from all other losses. As such, in this chapter, we only use the composite P_C to diagnose the proportion of losses caused by collisions and that caused by channel errors; this way, nodes can estimate their channel conditions based on their total loss statistics and their estimates of P_C.

For channel errors, we consider a sequence of k bits being sent at a constant modulation rate R over a channel with SNR σ. The bit-error rate, as a function of R and σ, is denoted by $BER_R(\sigma)$, and the probability of successfully transmitting all k bits is given by $(1 - BER_R(\sigma))^k$. Since an 802.11 packet consists of a preamble and PLCP header sent at low modulation rate, and a payload possibly sent at a higher rate, the probability of channel error for a single packet experiencing an SNR of σ can be computed as

$$\begin{aligned} P_e^p(\sigma) &= 1 - (1 - P_{e,h}^p(\sigma))(1 - P_{e,p}^p(\sigma)) \\ &= 1 - (1 - BER_{R_h}(\sigma))^{L_h}(1 - BER_{R_p}(\sigma))^L \end{aligned} \tag{4.2}$$

where $P_{e,h}^p(\sigma)$ and $P_{e,p}^p(\sigma)$ are the header and payload error probabilities respectively, L_h and L are the lengths of the header and payload respectively, R_h and R_p are the modulation rates of the header and payload respectively, and $BER_R(\cdot)$ is assumed to be a known function, which depends on the the signal constellation for each rate. In this chapter, we fix L_h, R_h, and R_p, and adapt L.

Since the SNR is unknown and varies between packets, it is modeled as a random variable. Thus the probability of packet error over all packets, P_e is the expectation of the expression in Equation (4.2) taken over the distribution of SNR, denoted by $f_\sigma(s)$:

$$P_e = 1 - \int (1 - BER_{R_h}(s))^{L_h}(1 - BER_{R_p}(s))^L f_\sigma(s) ds \tag{4.3}$$

For the simulations in this chapter, we assume SNR to have a log-normal distribution, where the mean is dependent on the path loss, and the variance is dependent on the type of environment [34].

Figure 4.1(a) shows BER as a function of SNR for each modulation rate used in 802.11b, based on the data sheet of the Intersil HFA3861B [16]. Inserting these values into Equation

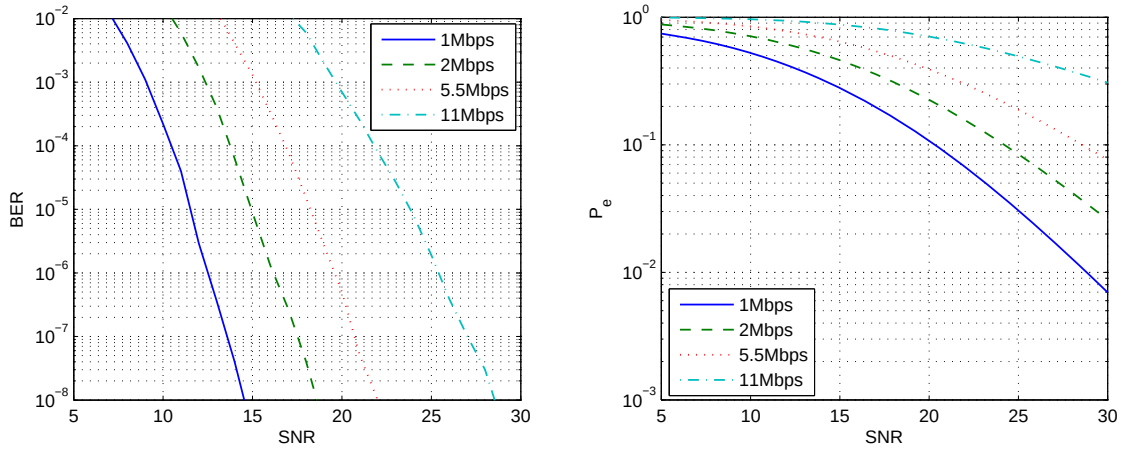


Figure 4.1. *(a)BER vs. SNR, and (b) probability of packet loss due to channel error vs. mean SNR for each modulation rate. SNR standard deviation is fixed at 7 dB.*

(4.3) and fixing the SNR variance yields probability of packet error, P_e, as a function of mean SNR, for each rate, shown in Figure 4.1(b).

Based on this probability of packet loss, the throughput for each rate can be computed as

$$Throughput = \frac{(1 - P_L)L_p}{\frac{L_p}{R_p} + \frac{L_h}{R_h} + T_{ov}} \tag{4.4}$$

where T_{ov} is additional overhead associated with packet transmission, including backoff, inter-frame spacing, and ACK. The denominator is the total time spent for each packet transmission, including the time it takes to send the modulated payload and header as well as general packet overhead. The total bits successfully transmitted per packet is L_p if the packet is successful, and 0 if it is lost. Thus the numerator is the expected number of bits received per packet transmission.

From Equations (4.3) and (4.4), we can plot throughput as a function of mean SNR for each modulation rate as shown in Figure 4.2. It can be seen that the optimal rate is a monotonic function of SNR. As a result, there is a contiguous region of SNRs for which a particular rate is optimal.

4.2 Link Adaptation Algorithms

In this section, we examine the potential throughput improvement achievable using our P_C estimate in a LA algorithm. To do this, we implement three LA algorithms in NS-2. The first is standard ARF, which is used as a baseline reference; the second is a modified version

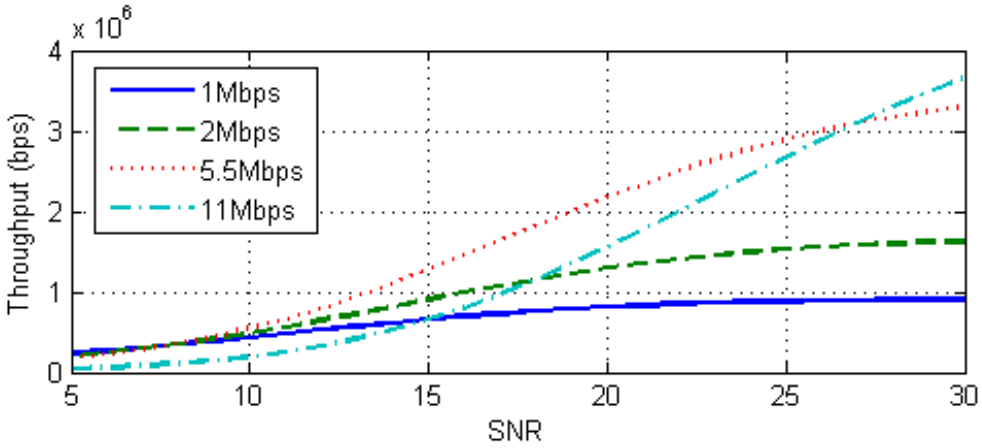


Figure 4.2. *Throughput vs. mean SNR for each modulation rate. SNR standard deviation is fixed at 7 dB.*

of the algorithm proposed in [21]; the third is a new algorithm proposed in this chapter, called SNRg.

4.2.1 ARF

The reference LA algorithm, variations of which are commonly implemented in commercial wireless cards, is ARF. The basic idea behind ARF is that consecutive packet losses is an indicator of poor channel conditions. Thus ARF decreases the modulation rate after M – usually 2 – consecutive unacknowledged packets. Conversely, after N successful packets, it increases the modulation rate. For our simulations, we use $M = 2$ and $N = 10$, which are the parameters used in [18].

4.2.2 Modified COLA

The Congestion-aware Link Adaptation (COLA) algorithm proposed in [21] is similar to ARF, except for a few modifications to account for the effect of loss due to collisions. The primary difference is that the thresholds for the number of success and failures needed to change rates are adaptive, depending on P_C and past packet transmission results. Additionally, the successes or failures need not be consecutive since a high P_C can make it unlikely to have a large number of consecutive successes even in perfect channel conditions.

For COLA, the state maintained by each node is (a) the number of transmission attempts, n_t, successes, n_s, and failures, n_f, at the current rate; and (b) a vector $\vec{N}$, in which the mth

entry, N_m is a threshold on n_s such that when $n_s > N_m$ at rate m, the rate is increased to $m+1$.

To account for the effect of P_C on packet losses, COLA differs from ARF in two ways. First, n_s need not be integer-valued. Rather than being a count of the number of actual successes, n_s is the expected number of packets that would have been successful in the absence of collisions, given the observed total loss statistics. When a packet is transmitted and an ACK is not received, not only are n_t and n_f incremented by one, but additionally n_s is incremented by P_C, which is the probability that the packet was lost due to collision and not due to channel error. If n_s were to only count actual successes, a high P_C would unnecessarily increase the number of transmission attempts needed to increase the modulation rate in favorable channel conditions. Second, rather than requiring a constant number of consecutive failures to decrease the rate, as in ARF, COLA decreases the rate when the number of failures, n_f, exceeds the number of failures expected for n_t packets in the absence of channel errors, i.e. $n_t \cdot P_C$, by a specified amount, k, which is set to 1. In this way, higher values of P_C cause the nodes to require more losses to drop the rate.

In order to avoid oscillation between two rates when the algorithm converges, $\vec{N}$ is updated every time the rate is changed. Specifically, each time the rate is decreased from $m+1$ to m, if all packets at rate $m+1$ fail, N_m is doubled so that the node becomes less aggressive in attempting to increase the rate back to m. However, whenever a packet succeeds at rate $m+1$, N_m is reset to 1; this way, if channel conditions change to make successful transmission at rate $m+1$ possible, there are no lingering effects of the increased N_m.

In [21], it is assumed that P_C is known, and further that it is the same for all nodes, so it cannot be used in our scenario, which includes hidden nodes. We therefore further modify COLA to use our P_C estimate in a two-stage algorithm. In the first stage, the modulation rate is held constant for 5 seconds, at the end of which P_C is estimated as described in Chapter 2. We have found via simulations that using 5 seconds of busy-idle data results in reasonably accurate estimates of P_C. In the second stage, the node counts successful and failed packets until it determines that the rate needs to be changed, according to the algorithm in [21] using the P_C estimate from the first stage. Once the rate has been changed, the node returns to the first stage to re-estimate P_C, since it may change with time as other nodes adapt their rates, and may vary between modulation rates due the difference in packet duration. We call this modified algorithm mCOLA, with pseudo-code shown in Algorithm 1. Note that this overall adaptation happens at a timescale of seconds, rather than tens of milliseconds as ARF or COLA because the first stage must last long enough to re-estimate P_C. In this paper we disable adaptation during this stage, but it may be possible to allow for stage 2 to continue for a prolonged period of time to continue to adapt at the fast timescale if the channel does not stabilize.

Algorithm 1 The mCOLA algorithm

```
1: Initialize n_t, n_s, n_f ← 0; ∀m, N_m ← 1
2: Transmit at current rate for 5 seconds
3: Receive BI signal from AP
4: Compute estimate of P_C from BI signal
5: for each transmission do
6:    n_t ← n_t + 1
7:    if no ACK then
8:       n_f ← n_f + 1
9:       n_s ← n_s + P_C
10:      if n_f ≥ n_t · P_C + k then
11:         n_s ← 0
12:         if m ≠ m_min then
13:            m ← m − 1
14:            if n_t == n_f then
15:               N_m ← 2N_m
16:            end if
17:            n_t, n_f ← 0
18:            Goto 2
19:         end if
20:      end if
21:   else
22:      n_s ← n_s + 1
23:      if m ≠ m_min then
24:         N_{m−1} ← 1
25:      end if
26:      if n_s > N_m and m ≠ m_max then
27:         m ← m + 1
28:         N_m ← 1
29:         n_t, n_s, n_f ← 0
30:         Goto 2
31:      end if
32:   end if
33: end for
```

4.2.3 SNR guess (SNRg)

The mCOLA algorithm is somewhat inefficient in that it consists of two separate stages, one in which the traffic conditions are estimated via P_C, and the other in which the channel conditions are estimated via loss counting. However, these stages need not be separate. In particular, during the five seconds over which P_C is estimated, an estimate of P_e can also be obtained. The decision to change rates can then be based on this estimate rather than

counting subsequent packet successes and failures. Specifically, nodes estimate P_C using the technique in Chapter 2, estimate P_L based on empirical measurements, and insert these into Equation (4.1), to obtain an estimate of P_e. They then use this estimate of P_e to estimate the mean SNR and choose the appropriate modulation rate.

In this chapter, we assume the SNR variance to be known, possibly set based on knowledge of the type of environment. For a known SNR variance and fixed modulation rate, P_e is a one-to-one function of mean SNR. Thus, the mean SNR can be approximated from the P_e estimate and the known modulation rate. It may also be possible to estimate both the mean and variance of the SNR distribution using additional information such as the received signal strength indicator; we plan to investigate this as part of our future work.

The complete SNRg algorithm executes on a 5 second loop, which can be timed to coincide with the broadcast busy-idle signals sent by the AP for P_C estimation. Every five seconds, each station estimates P_C using the busy-idle signal broadcast by the AP along with its local busy-idle signal. It then approximates P_L based on the empirical proportion of packets lost, n_f/n_t, and uses this estimate along with P_C to estimate P_e. The station then estimates SNR based on P_e and the current rate via lookup from a table based on Figure 4.1(b). Once the mean SNR has been estimated, the optimal modulation rate for that SNR can then be chosen by comparing the estimated mean SNR against several thresholds, namely the crossover points where the optimal rate changes as shown in Figure 4.2. Pseudo-code is shown in Algorithm 2. Note that like mCOLA, SNRg operates at a timescale on the order of seconds. This cannot be changed to a faster timescale, so it is not appropriate for rapidly changing channel conditions.

Algorithm 2 The SNRg algorithm

1: Transmit at current rate for 5 seconds
2: Receive BI signal from AP
3: Compute estimate of P_C from BI signal
4: $P_L \leftarrow n_f/n_t$
5: **if** $P_L == 1$ **then**
6: $\quad m \leftarrow m - 1$
7: $\quad$ Goto 1
8: **end if**
9: $P_e \leftarrow 1 - (1 - P_L)/(1 - P_C)$
10: Lookup SNR based on P_e and current rate
11: Set current rate to best rate for SNR
12: Goto 1

A timing diagram comparing COLA and SNRg is shown in Figure 4.3. The dotted arrows correspond to times when P_C is estimated, and the solid arrows correspond to times when the new modulation rate is chosen. Since COLA requires a variable number of transmission attempts before changing the modulation rate, the times when P_C is estimated, represented by the dotted arrows, do not occur at regular time intervals. As a result, it is not possible

for the AP to broadcast its busy-idle signal at one fixed period to satisfy all the nodes. In contrast, SNRg avoids this problem because the new rate is chosen at the same instant as P_C is estimated.

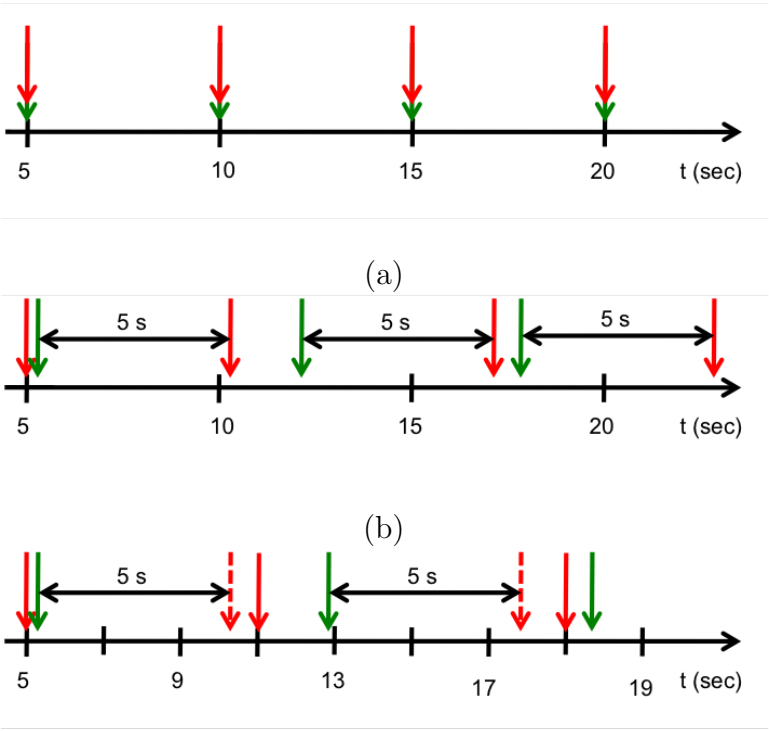


Figure 4.3. *Timing diagrams for (a) SNRg; (b) idealized COLA used in simulations; (c) COLA in a practical implentation. The dotted arrows correspond to times when P_C is estimated, and the solid arrows correspond to times when the modulation rate is chosen. In (c), the dashed arrows correspond to ideal P_C estimation times.*

For the simulations in this chapter, we assume nodes have access to a busy-idle signal from the AP corresponding to the most recent 5 seconds on demand; however, in practice, the busy-idle broadcasts from the AP must occur at fixed intervals, hence requiring the time between rate change and P_C estimation to be variable. Figure 4.3(c) shows an example timing diagram for a more practical version of mCOLA in which the AP broadcasts its busy-idle signal every 2 seconds, at odd integer values of t. While the node would ideally estimate P_C at the times marked by dashed arrows, it must delay its estimation to the times marked by dotted arrows in order to use the most recent busy-idle signal. This results in a trade-off in selecting the frequency of AP broadcasts: more frequent broadcasts allow for more rapid rate adaptation at the expense of increased transmission overhead. Investigation of this trade-off, and the resulting throughput is part of our future work.

4.3 Simulation results

To test the throughput gains achievable by the mCOLA and SNRg algorithms using our collision probability estimates, we use the NS-2 simulation package. In addition to the modifications described in Chapter 2 to allow for collection of the BI signal and estimation of collision probabilities, we have modified it to count the actual results of each packet transmission: collision, channel error, or success and report the results to each node with the same frequency as the BI signal. This allows nodes to estimate P_C in two different ways: using an empirical count, which will henceforth be referred to as *count*, and using our estimation technique , henceforth called *est*. The results in this thesis are for 802.11b, but they can easily be extended to 802.11a or g, or to any other carrier-sense multiple access MAC with multiple available modulation rates.

The test topology consists of 7 APs arranged to cover hexagonal cells, and 10 to 50 nodes placed at random over the area by a spatial Poisson process sending saturated traffic to the nearest AP. Transmission power is set so that each AP covers a radius of 100m. We fix $L_h = 1152$ bits, $R_h = 1$ Mbps, and $L_p = 18432$ bits, which are typical values for 802.11b. We generate 5 random topologies and simulate 3 minutes of traffic, using each of 5 different LA techniques – mCOLA using *count*, mCOLA using *est*, SNRg using *count*, SNRg using *est*, and ARF. An example trace of the changing rate using the SNRg method is shown in Figure 4.4. As seen, when P_e gets sufficiently high, the rate drops, and when P_e decrease, the rate increases. Note again that the timescale is slow relative to potential adaptations from ARF.

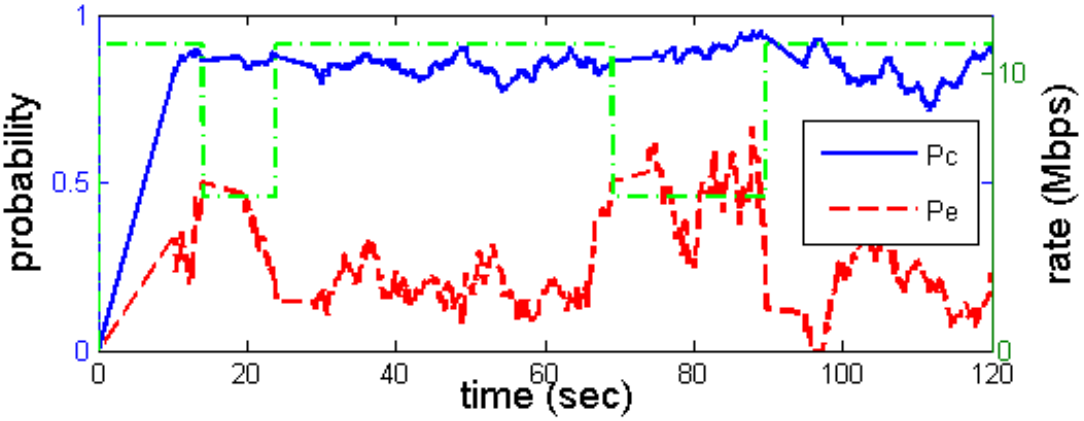


Figure 4.4. *P_e, P_C, and modulation rate over time for one node using SNRg.*

Figure 4.5 shows the throughput improvement, with respect to ARF, for each of the four techniques using P_C estimates, as a function of noise power. Figures 4.5(a), 4.5(b), and 4.5(c) correspond to scenarios with 50, 30, and 10 nodes respectively. The blue lines with squares and circles denote mCOLA, the red lines with triangles denote SNRg, the algorithms using *count* are dotted, and the algorithms using *est* are solid. As seen, when the noise power

is low, the throughput gain over ARF can be as much as a factor of 5. This is due to the fact that in this situation, very few losses are due to channel error, so the maximum rate is optimal for all nodes. However, because of the high incidence of collisions due to hidden nodes, ARF causes the nodes to lower their rates unnecessarily. As the noise power increases, the optimal rate for most of the nodes decreases as there are more losses due to channel errors. As P_e grows, relative to P_C, the assumption that losses are due to channel errors becomes increasingly more accurate. However, even as the noise gets very high, the nodes closest to the AP still have a sufficiently strong channel to send at a higher rate than that resulting from ARF, which overestimates the effect of channel relative to traffic. As a result, even at -95 dBm, there can be as much as a 15-20% throughput improvement over ARF. It is worth noting that while -135 dBm may be an unrealistically low noise power, if the coverage range is reduced from 100m to a more typical 30m, the same phenomenon occurs for higher noise powers.

In most situations, there is little difference between using *count* and *est* for either algorithm. This indicates that, in general, our P_C estimate is sufficiently accurate for these algorithms. However, the gap between the performance of SNRg *count* and SNRg *est* for low noise increases with the number of nodes. This is due to the fact that when P_C is large relative to P_e, as is the case when there is a large number of nodes and low noise power, a small relative error in estimating P_C can lead to a larger relative error in estimating P_e. Since the SNRg algorithm depends strongly on the P_e estimate, these errors are significant. mCOLA, on the other hand is robust to this problem since it does not use an estimate of P_e, and uses only P_C.

The per-node throughput improvements of SNRg compared to ARF for an example topology with noise power -125 dBm for *count* and *est* are shown spatially in Figures 4.6(a) and 4.6(b), respectively. Green circles represent nodes with increased throughput, red circles represent nodes with decreased throughput, and the size of the circles correspond to the change in throughput compared to ARF. When *est* is used, nodes near the periphery with less accurate estimates of P_C, choose incorrect rates. If they choose a modulation rate lower than the optimal rate, they not only achieve decreased throughput for themselves, but also decrease the throughput of their neighboring nodes because their lower-rate transmissions occupy the channel at the AP for a longer period of time than would higher-rate transmissions.

At high noise levels, such as -95dBm, SNRg achieves a 15% average throughput improvement over ARF; however, mCOLA achieves lower throughput than ARF. This is because nodes using mCOLA tend to use overly-aggressive modulation rates. Specifically, since N_m is reset to 1 every time a packet succeeds at rate $m+1$, mCOLA causes nodes to repeatedly attempt to send at any modulation rate for which they can successfully transmit a packet with non-zero probability, not necessarily with high probability. Due to large random variations in the channel, successful transmission is possible at high rates even when the average SNR is very low. This results in the use of high rates even when P_e is very high, resulting in lower throughput. Since SNRg takes into account the actual value of P_e, rather than simply

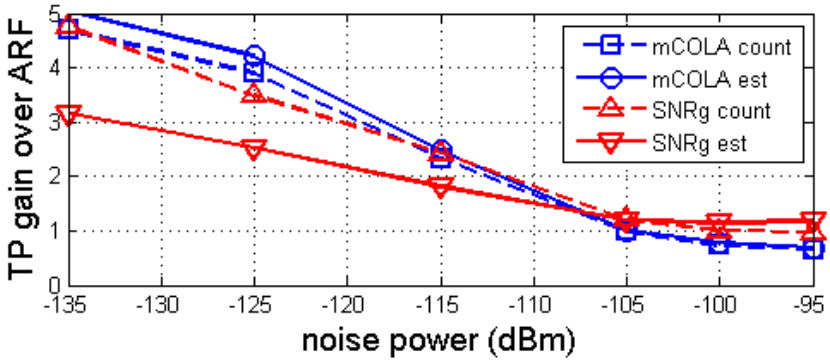
the indicator that $P_e \neq 1$, it is better able to choose the modulation rate which maximizes throughput.

Figures 4.6(c) and 4.6(d) show mCOLA *est* and SNRg *est*, respectively, for noise power -95 dBm on an example topology. As mentioned earlier, the selection of overly aggressive modulation rates lowers throughput for most nodes in Figure 4.6(c) as compared to ARF. However, in Figure 4.6(d), the magnitudes of throughput decreases with respect to ARF are smaller for nodes with decreased throughput, and some nodes with sufficiently strong channels achieve increased throughput with respect to ARF, causing the overall throughput to increase. It is interesting to note that in Figure 4.6(c), while the selection of higher modulation rates decreases the overall throughput and that of most nodes, it does allow for greater throughput than SNRg for the nodes closest to their APs. In particular, the node at (400, 150) achieves greater throughput in part due to the use of shorter higher-rate transmissions by its neighboring nodes, which allows it to access the channel for a greater proportion of the time.

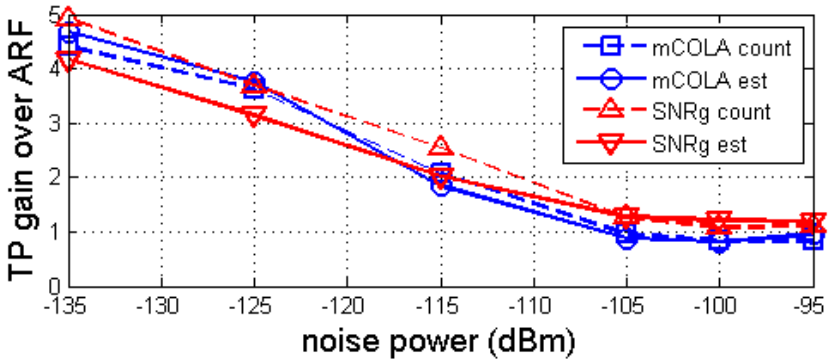
4.4 Conclusions

In this chapter we have shown that the P_C estimates from Chapter 2 can be used in LA to significantly improve throughput in 802.11 wireless LANs. We have proposed a new link adaptation algorithm, called SNRg, which provides up to a factor of five 400% throughput gain over standard ARF in scenarios where all losses are due to collision. In scenarios in which the probability channel error is larger, it results throughput gains of 15% or more. Additionally, with newer wireless LAN standards, there is a much greater range of modulation rates than in 802.11b, so the impact of these adaptations is potentially even greater.

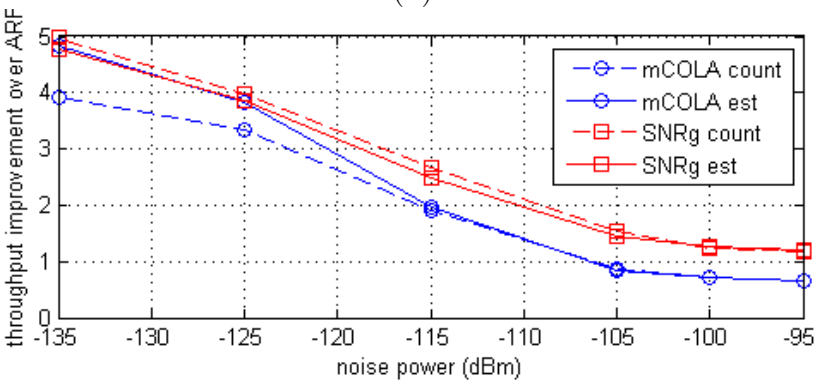
As mentioned earlier, directions for future research include (a) development of methods to use P_C estimates along with other information to estimate both the SNR mean and variance; and (b) investigation of the performance of mCOLA in realistic scenarios where busy-idle signals are only available at specific times. Another direction is development a hybrid algorithms or a modified SNRg algorithm which uses mCOLA-like elements to achieve the performance of mCOLA in scenarios where it outperforms SNRg, e.g. when P_C is high and noise power is low. Based on P_C and SNR estimation, nodes may be able to choose the better of the two algorithms for the current scenario.



(a)

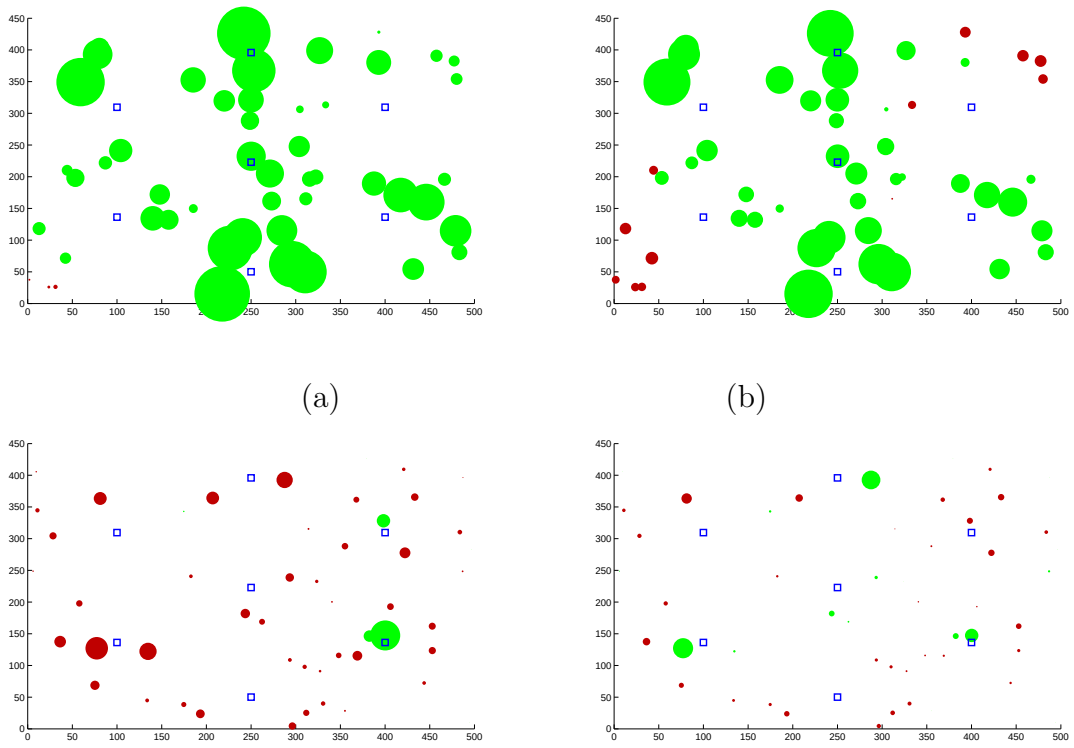


(b)



(c)

Figure 4.5. *Average throughput improvement over ARF for scenarios with (a) 50 nodes; (b) 30 nodes; (c) 10 nodes.*



(a) (b)

(c) (d)

Figure 4.6. *Throughput improvement over ARF of each node over space in representative topologies with 50 nodes for (a) SNRg count with noise power -125 dBm; (b) SNRg est with noise power -125 dBm; (c) mCOLA est with noise power -95 dBm; and (d) SNRg est with noise power -95 dBm. The small squares represent the position of the APs, and the circles represent the nodes. The lighter green circles indicate nodes with increase throughput relative to ARF, and darker red circles indicate nodes with decreased throughput. The size of the circle is proportional to the node's change in throughput.*

Chapter 5

Improving Contention Window Adaptation

In the 802.11 standard, the contention window is used for collision avoidance. In the BEB protocol, CW is adjusted with every retransmission attempt of a packet. It initially starts at a value $CW_{min} = 32$ and increases by a factor of $\alpha = 2$ for each retransmission attempt up to $CW_{max} = 1024$. The BEB protocol suffers in three types of scenarios. The first is when there is a sufficiently large number of nodes such that $CW_{min} = 32$ is too small to prevent collisions. This is addressed by several others in the literature[4, 10, 24, 36, 41, 43, 44]. The second is when the number of nodes is small, but there are is a poor channel, which causes channel-based losses that are incorrectly interpreted as collision-based losses. The third is in the presence of the hidden terminal problem, where staggered collisions are incorrectly treated as direct collisions.

In this chapter, we propose a distributed algorithm whereby each node uses information from the local and AP BI signal to adapt its contention window size in order to improve overall network utility. In Section 5.1, we formulate an expression for total network throughput and show that the derivative of utility with respect to the average backoff length of node i depends only on quantities which can either be observed by node i or are contained in the busy-idle signal of its AP. In Section 5.2, we outline an algorithm in which nodes collect the relevant information, compute their derivatives, and adapt their contention window size. In Section 5.3 we present simulation results showing improvements in utility with significant throughput gains for the three major cases where standard contention window adaptation fails: high node count, high channel error, and hidden nodes. We offer come conclusions in Section 5.4

5.1 Analytical Framework

In this section we present an expression for total network utility and derive an expression for its derivative with respect to the average backoff length of node i. It is important to maximize utility, rather than individual throughput, because the latter results in the undesirable trivial solution of all nodes setting their CW_{min} to zero. As such, we assume that all nodes seek to maximize this global objective, namely the utility.

To enforce fairness, we consider the utility of each node to be the log of its throughput [19] and maximize the total network utility given by:

$$U = \sum_j \log TP_j. \tag{5.1}$$

The throughput, TP_i, of node i can be expressed as

$$\begin{aligned} TP_i &= L_i S_i (1 - P_{Li}) \\ &= L_i S_i (1 - P_{SCi})(1 - P_{DCi})(1 - P_{ei}) \end{aligned} \tag{5.2}$$

where L_i is the number of bits per packet, S_i is the number of packets sent per time slot, i.e. $1/S_i$ is the average number of timeslots that elapse between the beginning of consecutive transmissions of node i, P_{Li} is the packet loss rate for node i, and P_{SCi}, P_{DCi}, and P_{ei} are the probabilities of staggered collision with hidden nodes, direct collision with neighboring nodes, and channel error for node i, respectively.

S_i can be thought of as the probability of node i sending in any given slot. This can be broken down via the chain rule into:

$$\begin{aligned} S_i = P(\text{node } i \text{ sends}) = P(\text{channel idle in previous slot}) \\ \times P(\text{node } i \text{ sends}|\text{channel idle in previous slot}) \end{aligned} \tag{5.3}$$

If we define $\overline{W}_i$ as the average backoff chosen for node i, we have

$$P(\text{node } i \text{ sends}|\text{channel idle in previous slot}) = \frac{1}{\overline{W}_i}. \tag{5.4}$$

Since the channel alternates between busy and idle states, we also have that

$$P(\text{channel idle in previous slot}) = \frac{T_i}{B_i + T_i}, \tag{5.5}$$

where B_i and T_i are the average durations of busy and idle periods, respectively.

We can thus express S_i as

$$S_i = \frac{T_i}{B_i + T_i} \frac{1}{\overline{W}_i} \tag{5.6}$$

Let $\mathcal{N}_i$ be the set of "neighboring" nodes to node i, i.e. the set of nodes whose transmissions can be sensed by node i. Then, assuming that the length of the idle periods is roughly geometric, which is true for Poisson traffic as well as saturated traffic [6], it is clear that

$$\begin{aligned} T_i &= 1/P(\text{any node in } \mathcal{N}_i \cup \{i\} \text{ starts sending in given slot}) \\ &= \left(1 - \prod_{k \in \mathcal{N}_i} \left(1 - \frac{1}{\overline{W}_k}\right)\right)^{-1} \end{aligned} \tag{5.7}$$

For each node $k \in \mathcal{N}_i$, the probability of collision is the probability that node i completes its backoff at the same time slot as node k, which is $1/\overline{W}_k$. This event occurs independently for each k. Thus, the probability of direct collision can be expressed as

$$P_{DCi} = 1 - \prod_{k \in \mathcal{N}_i} \left(1 - \frac{1}{\overline{W}_k}\right) \tag{5.8}$$

The staggered collisions for node i are caused by a set of hidden nodes $\mathcal{H}_i$ which can interfere at the AP but cannot be sensed by node i. Although these nodes may share some neighbors with i, for the sake of simplicity, they are assumed to behave independently of node i. The total rate at which the nodes in $\mathcal{H}_i$ send is $\sum_{j \in \mathcal{H}_i} S_j$. Assume they are sent according to a Poisson process, and that all packets are of duration D backoff slots. This assumption is reasonable for the saturated case, where the hidden node problem is most severe, since with saturated traffic, nodes use maximum length packets. In practice, there may also be shorter control packets, which cause the adaptation to be conservative, choosing slightly larger contention windows than necessary. However, we have empirically found that including them in our simulations still results in significant improvements in overall network utility. Under these assumptions, the probability that a packet sent by node i does not collide with a packet from a node in $\mathcal{H}_i$ is the probability that there is no packet sent within $D-1$ slots before or D slots after. Basic Poisson theory states that this is the probability a Poisson random variable with parameter $(2D-1)\sum_{j \in \mathcal{H}_i} S_j$ is equal to zero, which results in the following expression for P_{SCi}:

$$\begin{aligned} P_{SCi} &= 1 - e^{-(2D-1)\sum_{j \in \mathcal{H}_i} S_j} \\ &= 1 - \prod_{j \in \mathcal{H}_i} e^{-(2D-1)S_j} \end{aligned} \tag{5.9}$$

Substituting Equations (5.6), (5.8), and (5.9) into Equation (5.2) yields:

$$TP_i = \frac{L_i T_i (1 - P_{ei})}{(B_i + T_i)\overline{W}_i} \prod_{k \in \mathcal{N}_i} \left(1 - \frac{1}{\overline{W}_k}\right) \prod_{j \in \mathcal{H}_i} e^{-(2D-1)S_k} \tag{5.10}$$

In order to maximize the utility in a distributed manner, each node i must be able to estimate the derivative of Equation (5.1) with respect to $\overline{W}_i$ using only locally available information.

This derivative can be broken up into 3 terms: one related to the node itself, one to its neighbors, and one to its hidden nodes:

$$\frac{\partial}{\partial \overline{W}_i} U = \frac{\partial}{\partial \overline{W}_i} \log TP_i + \sum_{k \in \mathcal{N}_i} \frac{\partial}{\partial \overline{W}_i} \log TP_k + \sum_{j \in \mathcal{H}_i} \frac{\partial}{\partial \overline{W}_i} \log TP_j \tag{5.11}$$

We evaluate each of these terms in Appendix A to obtain:

$$\begin{aligned} \frac{\partial}{\partial \overline{W}_i} U = & -\frac{1}{\overline{W}_i} + \left(\frac{1}{T_i} - \frac{1}{B_i + T_i} \right) (1 - P_{DCi}) \frac{T_i^2}{\overline{W}_i^2} \\ & + |\mathcal{N}_i| \left[\frac{B_i}{T_i (B_i + T_i)} (1 - P_{DCi}) \frac{T_i^2}{\overline{W}_i^2} - \frac{1}{\overline{W}_i^2 - \overline{W}_i} \right] \\ & + |\mathcal{H}_i| \frac{(2D-1) T_i}{(B_i + T_i) \overline{W}_i^2} \end{aligned} \tag{5.12}$$

We use this equation in Section 5.2 to derive a contention window adaptation algorithm.

5.2 Adaptation Algorithm

We assume an 802.11 network in infrastructure mode, potentially with multiple APs. APs periodically broadcast the BI signal to all associated nodes, which the nodes can use to estimate their probabilities of each type of loss. Additionally, from this same information, nodes can estimate several quantities necessary for computation of the derivative in Equation (5.12).

Notice that all the quantities on the right-hand side of Equation (5.12) can be observed or estimated by node i. Specifically, $\overline{W}_i$ can be observed by recording the chosen backoffs; B_i and T_i can be observed from the busy-idle signal; P_{DCi} can be estimated from the local and AP busy-idle signal; $|\mathcal{N}_i|$ can be observed from headers of overheard packets; and $|\mathcal{H}_i|$ can be obtained if the AP additionally broadcasts a list of the nodes it hears or it can be estimated as in Chapter 2.

By estimating the derivative of network utility with respect to their average backoff, nodes can tune their contention window in order to improve network utility. We begin by examining the case where $\alpha = 1$, so that it is sufficient to find the optimal $\overline{W}_i$, then we extend to other values of α, where it is necessary to find both $\overline{W}_i$ and CW_{min}.

There are two ways to exploit the derivative in Equation (5.12). The first is to use a gradient ascent method, whereby each node changes its contention window size by an

amount proportional to its derivative. We call this the "gradient" approach. The second is to set the derivative equal to zero, and solve the resulting quadratic equation. Nodes can then immediately change their contention window size to this value. We call this the "zero" approach.

The zero approach cannot find the actual zero of the derivative when the current average contention window size, W_0, is far from the optimal $\overline{W}_i$ because the observed quantity T_i depends on the current contention window size. More precisely, if we use the notation $T_i(\overline{W}_i)$ to denote the dependency of T_i on $\overline{W}_i$, the observed quantity is $T_i(W_0)$. An example can be seen in Figure 5.1 where the solid blue curve plots $\frac{\partial}{\partial \overline{W}_i} U(T_i(\overline{W}_i), \overline{W}_i)$, and the dashed red curve plots $\frac{\partial}{\partial \overline{W}_i} U(T_i(W_0), \overline{W}_i)$. As seen, at $\overline{W}_i = W_0$, these are equal; however, for $\overline{W}_i > W_0$, $|\mathcal{N}_i| \geq 1$, and $W_0 \geq 1$, Equation (5.12) implies that $T_i(\overline{W}_i) > T_i(W_0)$. Therefore, at the zero of the dashed curve, namely $\overline{W}_i = W_0^*$, the solid curve is still positive. The zero of the solid curve must therefore occur at $\overline{W}_i^*$ such that $|\overline{W}_i^* - W_0| > |W_0^* - W_0|$. Thus if the algorithm were to jump to the estimated zero, W_0, it would systematically undershoot the actual optimal and converge slowly. To increase the speed of the convergence, it is useful to slightly overshoot the target W_0^*. We have empirically found via NS-2 simulations that an overshoot of 25% works well.

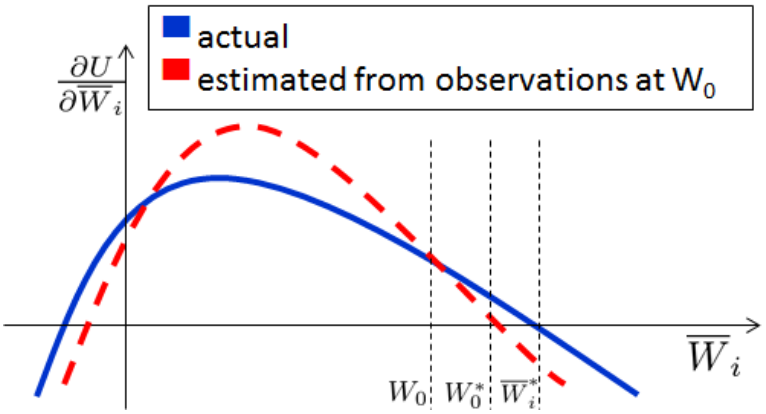


Figure 5.1. *An example plot of* $\frac{\partial}{\partial \overline{W}_i} U(T_i(\overline{W}_i), \overline{W}_i)$ *as a function of* $\overline{W}_i$*, represented by the solid blue curve, and* $\frac{\partial}{\partial \overline{W}_i} U(T_i(W_0), \overline{W}_i)$ *as a function of* $\overline{W}_i$*, represented by the dashed red curve.*

The gradient approach is guaranteed to converge to the optimal solution assuming that all the relevant quantities are estimated accurately, but the convergence can be slow. The zero approach comes with no such guarantees, and in fact it is possible for the function $\frac{\partial}{\partial \overline{W}_i} U(T_i(\overline{W}_i), \overline{W}_i)$ computed in a given step to be convex, rather than concave, for very small values of T_i, potentially causing $\overline{W}_i$ to change by a large amount in the wrong direction. However, by using a combination of the current derivative estimate along with the approximated zero, it is possible to assure convergence at a faster rate than using the gradient approach alone. Specifically, our proposed approach is the use the zero method when its direction of the change is in agreement with the derivative. Otherwise, the change in

contention window size is adjusted by an amount proportional to the value of the derivative. The complete algorithm, for each node i, is shown in Algorithm 1. The values in step (9) are found via empirical tuning within NS-2.

Algorithm 3 The CW adaptation algorithm

Loop:

1. Transmit at current rate for 5 seconds and observe:
   - $\overline{W}_i$
   - $|\mathcal{N}_i|$
   - busy-idle signal
2. Receive busy-idle signal from AP
3. Estimate P_{DCi} and $|\mathcal{H}_i|$ using BI signal
4. Estimate $\texttt{dUdW} = \frac{\partial}{\partial \overline{W}_i} U$ via Equation (5.12)
5. Solve for $\texttt{target\_W} = \{\overline{W}_i : \frac{\partial}{\partial L} TP = 0\}$
6. If $\alpha \neq 1$, compute `target_Wo` via Equation (5.18)
7. If $(sign(\texttt{target\_Wo} - \texttt{Wo}) = sign(\texttt{dUdW}))$
   - $\texttt{Wo} = \texttt{Wo} + 0.25(\texttt{target\_Wo} - \texttt{Wo})$

   Else
   - $\texttt{Wo} = \texttt{Wo} + 0.25(\texttt{target\_Wo} - \texttt{Wo}) + c \cdot \texttt{dUdW}$
8. Truncate changes of over 50%
9. Update c as follows:
   - If change is in same direction as last step increase c by 25%
   - Else decrease c by 50%

In the situation where $\alpha = 1$, it is sufficient to set $CW_{min,i} = 2\overline{W}_i$, but for $\alpha \neq 1$ or in the case of any Markovian packet-level adaptation, it is not trivial to select the appropriate $CW_{min,i}$ from the optimal $\overline{W}_i$. For a given backoff procedure, it is possible to derive an expression for the relationship between CW_{min} and $\overline{W}_i$, but this expression typically depends on the probability of packet loss, P_L. Fortunately, an estimate of P_L can be obtained from an empirical count assuming relatively stationary traffic.

We now derive the expression for $CW_{min,i}$ for the standard backoff procedure. To do so,

we can consider a Markov chain with M states, where M is the retransmit limit. The state transition probabilities are:

$$p_{m,m+1} = P_L \qquad \text{for } m = 0, ..., M-1 \tag{5.13}$$

$$p_{m,0} = 1 - P_L \qquad \text{for } m = 0, ..., M-1 \tag{5.14}$$

$$p_{M,0} = 1 \tag{5.15}$$

All other transitions have zero probability. From steady-state equations, we can solve for the steady-state proportion of backoffs chosen from each state, π_m. The average value of the chosen backoff in state m is $\frac{1}{2}\min(CW_{min,i} \cdot \alpha^i, CW_{max,i})$. Thus the average contention window size can be expressed as

$$\overline{W}_i = \sum_{i=0}^{M} \pi_i \frac{1}{2} \min(CW_{min,i} \cdot \alpha^i, CW_{max}) \tag{5.16}$$

$$= \frac{1}{2} CW_{min} \sum_{i=0}^{M} \pi_i \min\left(\alpha^i, \frac{CW_{max}}{CW_{min}}\right) \tag{5.17}$$

If we define $m = CW_{max}/CW_{min,i}$ as a constant, then there is a linear relationship between $\overline{W}_i$ and $CW_{min,i}$:

$$CW_{min,i} = \frac{2\overline{W}_i}{\sum_{i=0}^{M} \pi_i \min(\alpha^i, m)} \tag{5.18}$$

In order to select the optimal $CW_{min,i}$, node i estimates the optimal $\overline{W}_i$ and observes P_L. It then uses P_L to solve for π_m and uses these values along with $\overline{W}_i$ to select $CW_{min,i}$ via Equation (5.18).

5.3 Simulation results

To characterize the throughput gains for our contention window adaptation algorithm, we use the NS-2 simulation package. We have made modifications to allow for collection of the BI signal, estimation of collision probabilities, and execution of the the adaptation algorithm. Simulations are carried out for 802.11b, but the results are generally extensible to any MAC which uses carrier-sense multiple access.

Figure 5.2 shows the relationship between throughput and contention window for various channel error rates. The topology consists of two nodes and a single AP in a single collision domain with no hidden nodes. Since the topology is effectively symmetric, increasing utility is equivalent to increasing throughput, and thus it is sufficient to examine throughput. The

nodes send saturated traffic, and use the fixed contention window size indicated on the x-axis. The process is repeated for 10 different contention window sizes and 3 different channel error probabilities. Throughput is plotted as a function of contention window size as the solid lines, with the upper blue curve corresponding to $P_{ei} = 0$, the middle red curve corresponding to $P_{ei} = 0.3$, and the lower green curve corresponding to $P_{ei} = 0.6$. It can be seen that the optimal contention window size is less than $CW_{min} = 32$ in all cases.

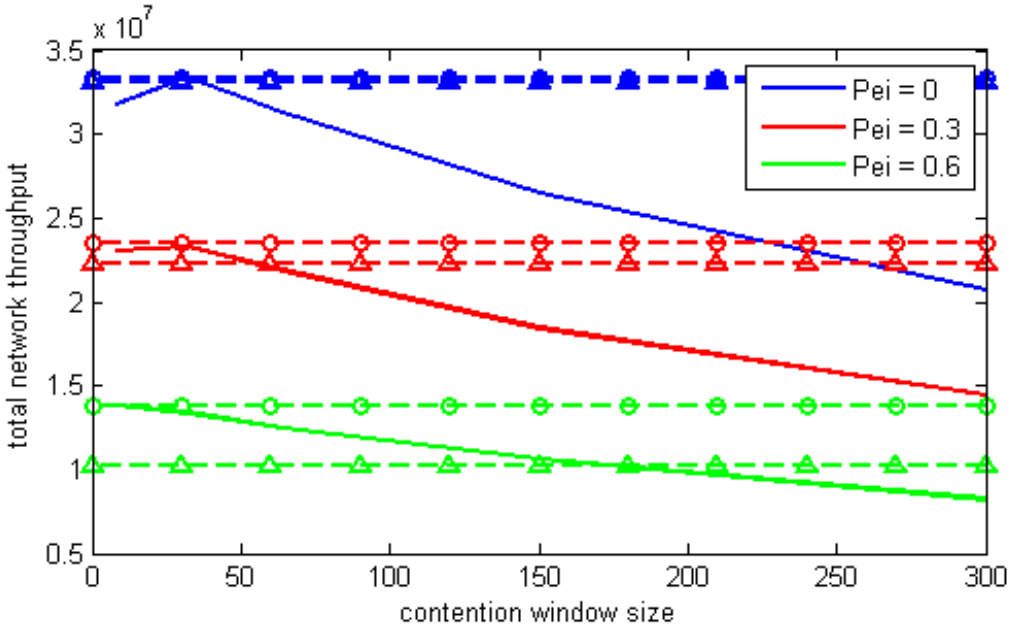


Figure 5.2. *Throughput as a function of contention window, for 2 nodes.*

The throughput achieved by the standard BEB algorithm is shown as the horizontal dashed lines with triangles. It is close to optimal for $P_{ei} = 0$, but suffers as P_{ei} increases. This is because the standard BEB algorithm assumes all losses are due to collisions, thus causing nodes to misinterpret the losses caused by channel error as losses caused by high channel contention. The throughput achieved by our proposed algorithm is shown as the horizontal dashed lines with circles. It can be seen that it achieves near-optimal throughput regardless of P_{ei}, achieving a 35% throughput improvement for $P_{ei} = 0.6$.

While the standard BEB protocol works reasonably well for small numbers of nodes in the absence of channel errors or hidden nodes, performance drops off rapidly as the number of nodes increases, since for a large number of nodes $CW_{min} = 32$ is small enough that the collision probability of the first attempt is too large. This drop in performance can be seen in Figure 5.3, which shows throughput as a function of the number of nodes. The blue curve is for the standard BEB algorithm, and the red and green are for the adaptive algorithm with $\alpha = 1$ and $\alpha = 2$, respectively. While the throughput for all methods decreases with number of nodes, the slope is significantly smaller for the adaptive algorithms, resulting in 14% improved throughput for 20 nodes and 24% for 40 nodes as compared to the standard.

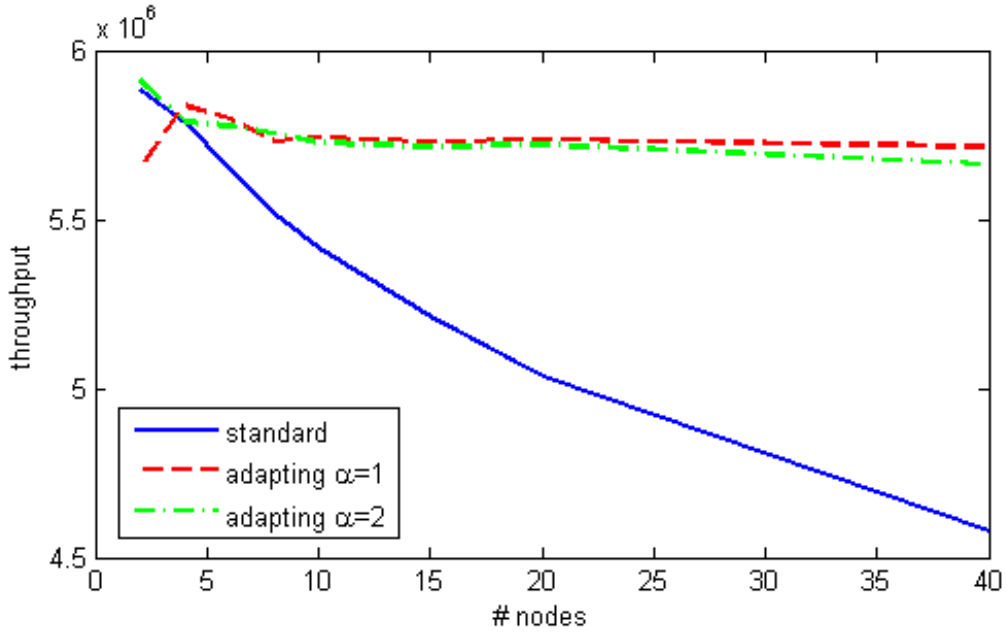


Figure 5.3. *Throughput vs number of nodes for standard and adaptive with* $\alpha = 1$ *and* $\alpha = 2$.

Figure 5.4 plots CW_{min} as a function of time for one node in the same scenario as Figure 5.3 for 5, 10, and 15 nodes. It can be seen that the convergence time increases with the number of nodes, taking about 40 seconds, or 8 iterations, for 10 nodes.

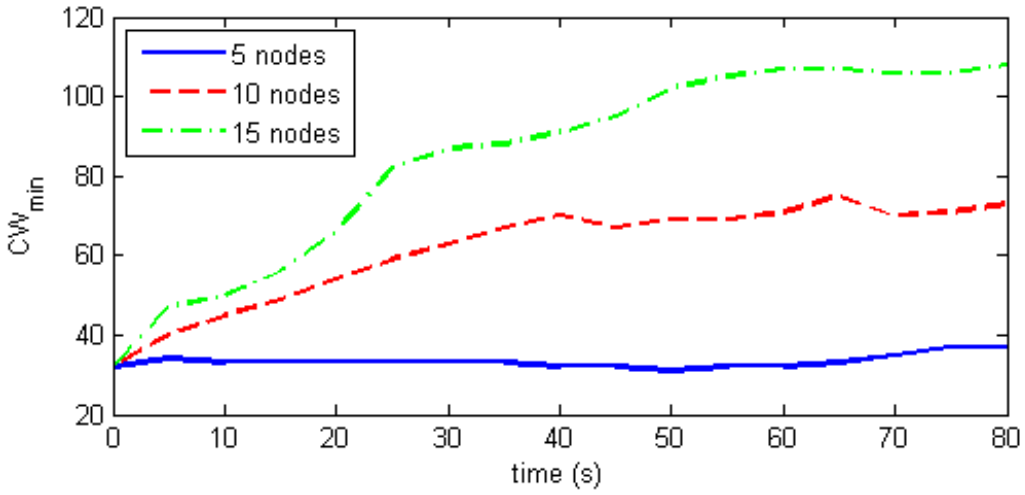


Figure 5.4. CW_{min} *vs time for single AP with no hidden nodes and 5-15 nodes.*

Figure 5.5 shows the total throughput for a scenario with 20 nodes sharing a single access point with no hidden nodes and various frame error rates (FERs) for standard BEB

and adaptation for various values of α. It can be seen that the value of α does not make a significant difference in the throughput performance. For $FER = 0$, regardless of α, the adaptive algorithm outperforms the standard by 14%. As the FER increases, there is less to gain via adaptation.

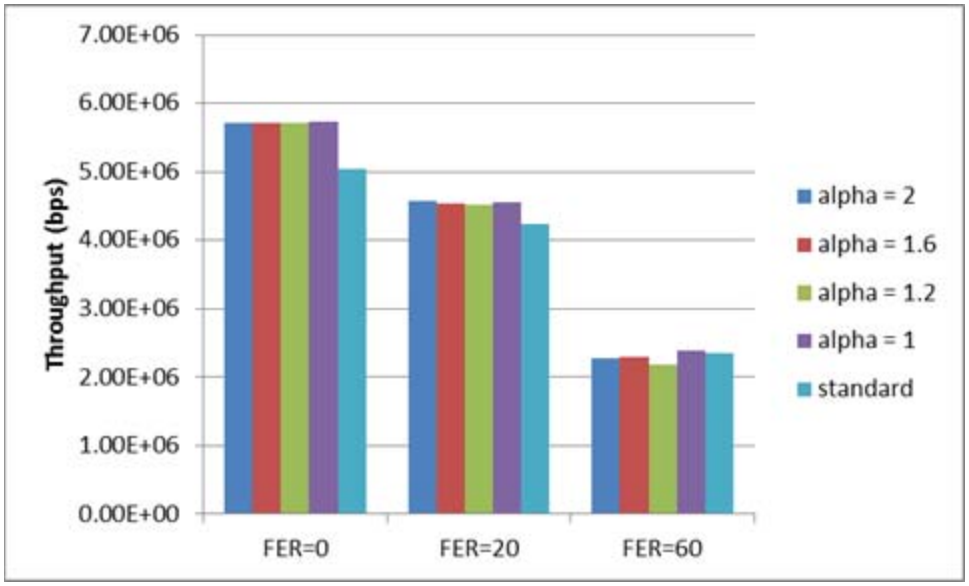


Figure 5.5. *Throughput vs standard for various FER and alpha for 20 nodes with no hidden nodes.*

Figure 5.6 shows a scenario with 20 nodes associated with a single AP arranged uniformly in a large circle with a large radius such that all nodes have the same non-zero number of neighboring and hidden nodes. As seen, even for large values of P_{ei}, there is an improvement of as much as 350% compared to standard. This improvement decreases with P_{ei}, but increases with α. The decrease with P_{ei} is due to the fact that as P_{ei} increases, the standard algorithm increases the contention window not because it recognizes the hidden node problem, but because it assumes there is a greater number of collisions which improves the behavior despite the incorrect reasoning. The increase with α is due a phenomenon which causes bursts of packets to be sent when hidden nodes choose long backoffs. With large values of α, a given $\overline{W}_i$ results in a smaller CW_{min}. Under these conditions, when a transmission by node i is successful, suggesting that nodes in $\mathcal{H}_i$ are currently in a long backoff, node i is more likely to transmit again immediately, allowing better utilization of the time during long backoffs of hidden nodes.

In a multi-access point network with asymmetric hidden node conditions, fairness becomes an issue. Nodes which do not suffer from the hidden node problem can dominate channel access and starve out nodes with less favorable conditions. Figure 5.7(a) show a histogram of the throughput of 50 nodes placed randomly over an area covered by 7 APs with hexagonal cells. It can be seen that in this scenario, 31 of the 50 nodes have throughput under 20 kbps. However, when adaptation is applied with $\alpha = 1$ or $\alpha = 2$, as shown in

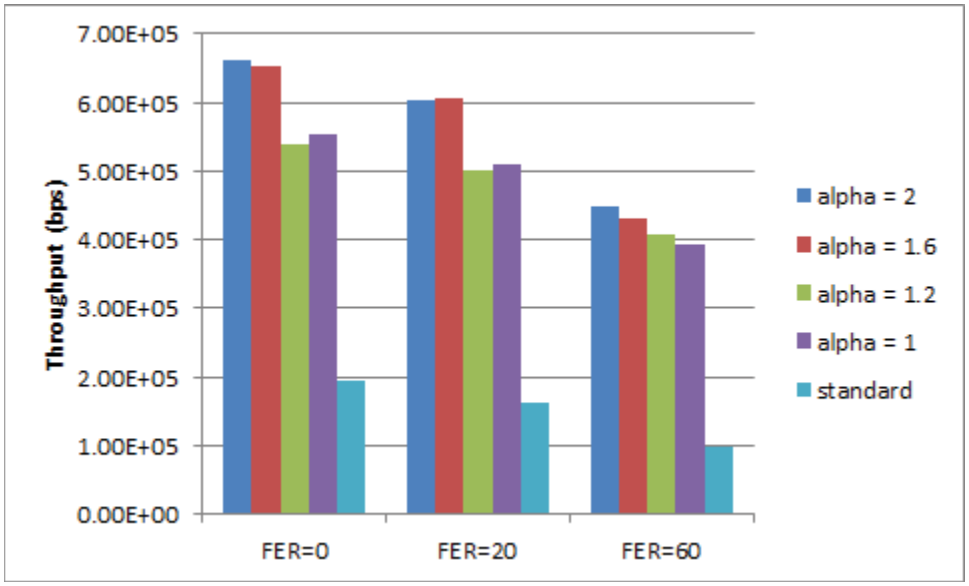


Figure 5.6. *Throughput vs standard for various FER and alpha for 20 nodes with hidden nodes.*

Figures 5.7(b) and 5.7(c), respectively, these nodes have improved throughput, resulting in much greater fairness. Total throughput is decreased, but overall utility is improved.

To quantify the improvement in fairness and utility, we introduce 2 metrics. The first is Jain's fairness index [17], which quantifies the fairness on a scale of $\frac{1}{n}$ to 1:

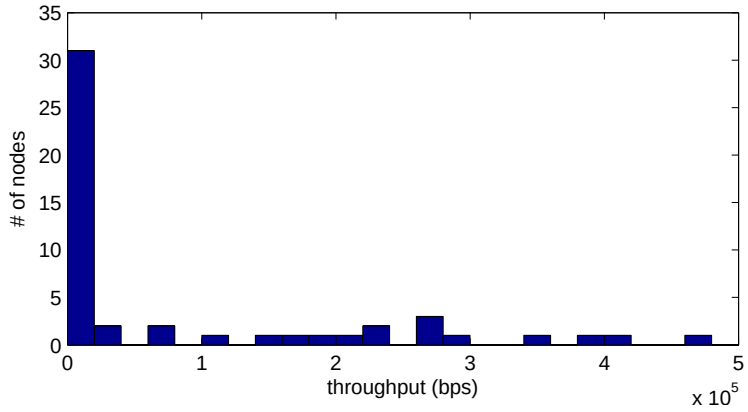
$$J = \frac{(\sum_i TP_i)^2}{n \cdot \sum_i TP_i^2} \tag{5.19}$$

The second we call the "equivalent equal throughput":

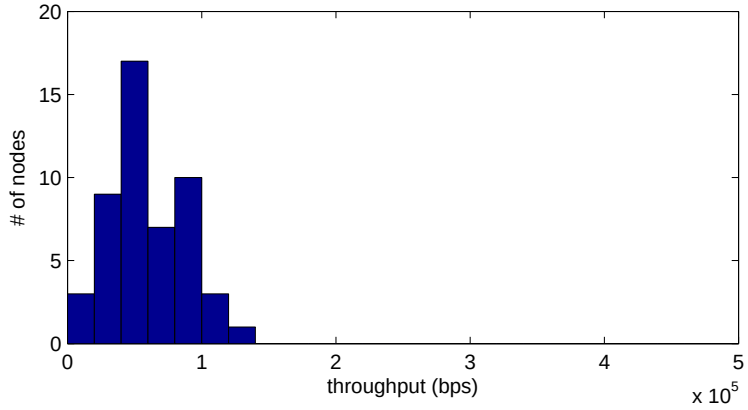
$$eeTP = e^{\frac{1}{n}\sum_i log(TP_i)} \tag{5.20}$$

This expresses the average throughput n nodes with equal throughput would have to achieve to obtain the given utility, $U = \sum_i log(TP_i)$. This quantity is easier to compare than the actual utility, since it has units of bits/sec.

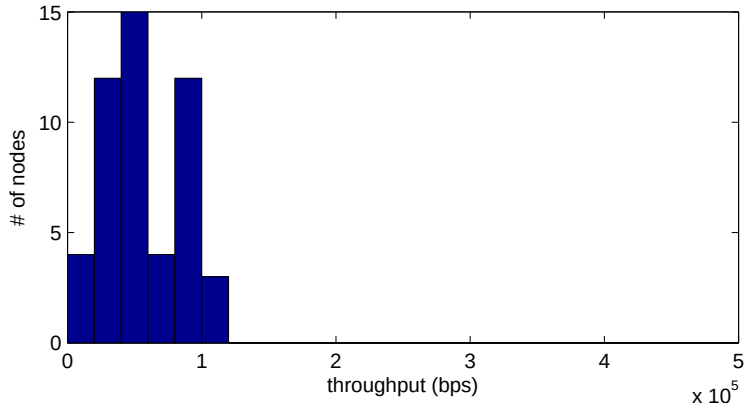
We ran 15 simulations on various random topologies with 7 APs 40-60 nodes and found the average throughput decreased by 44%, but the average utility increased by 7%, corresponding to an equivalent equal throughput gain of 165% and a 52% improvement in the Jain fairness index.



(a)



(b)



(c)

Figure 5.7. *Histogram of throughputs of each of 50 nodes for (a) standard BEB, (b) adaptation with* $\alpha = 1$*, and (c) adaptation with* $\alpha = 2$*.*

5.4 Conclusions

In this section, we proposed a novel contention window adaptation scheme in which nodes use information shared by the AP to optimize contention window sizes in a distributed fashion in order to improve network utility. We have examined three cases in which the standard BEB algorithm performs poorly and shown via NS-2 simulations that our method can improve throughput by as much as 24% in the high traffic scenario, 35% in the high channel error scenario and 350% in the presence of the hidden terminal problem.

Our results indicate that performance can vary depending on the chosen value of α. Determining the optimal alpha remains an open problem. Additionally, it may be the case that exponential backoff is inferior to other per-packet adaptation approaches such as those in [4, 24, 36, 44]. Since our algorithm works at a slower time scale, it can potentially be combined with these other methods to further improve performance.

Chapter 6

Packet Length Adaptation

Packet length is an intriguing parameter for optimization because it interacts with multiple types of loss. Not only are longer packets are subject to greater channel error rates due to the increased number of bits, but they are also subject to a higher probability of SC1. While there has been a significant amount of research on packet length adaptation, many of the loss models are greatly simplified, focusing on a single type of loss. In order to design a more sophisticated adaptation algorithm for a network with multiple causes of packet loss, a node must be able to determine the proportion of each type of packet loss it experiences in order to choose the optimal packet length. Using our collision probability estimation technique, we are able to overcome some of the shortcomings of existing approaches.

In this chapter, we use a packet loss model which includes random channel fading across packets as well as direct and staggered collisions to analyze the impact of MAC layer packet length on throughput. We propose a local packet length adaptation algorithm whereby each node estimates its current probability of each type of packet loss in order to compute the derivative of throughput with respect to packet length, and to adapt accordingly. The chapter is organized as follows: Section 6.1 describes the impact of the packet loss model; mathematical analysis of throughput and its derivative with respect to packet length is included in Section 6.2; the adaptation algorithm is described in Section 6.3; simulation results are presented in Section 6.4; conclusions are discussed in Section 6.5

6.1 Packet Loss Model

A major shortcoming of existing work on packet length adaptation is the use of an oversimplified loss model. Often the channel is modeled as a binary symmetric channel not

accounting for packet-to-packet variations in fading or collisions. In this section, we use a packet loss model which includes random channel fading across packets as well as direct and staggered collisions to analyze the impact of MAC layer packet length on throughput. We propose a local packet length adaptation algorithm whereby each node estimates its current probability of each type of packet loss in order to compute the derivative of throughput with respect to packet length, and to adapt accordingly.

As described in Chapter 4, the total packet loss probability P_L can be computed as

$$P_L = 1 - (1 - P_C)(1 - P_e) \tag{6.1}$$

where P_C is the probability of collision, and P_e is the probability of channel error, which is assumed to be independent of P_C.

The probability of packet error over all packets, P_e is the expectation of the expression in Equation (4.2) taken over the distribution of SNR, denoted by $f_\sigma(s)$:

$$P_e = 1 - \int (1 - BER_{R_h}(s))^{L_h}(1 - BER_{R_p}(s))^{L} f_\sigma(s) ds \tag{6.2}$$

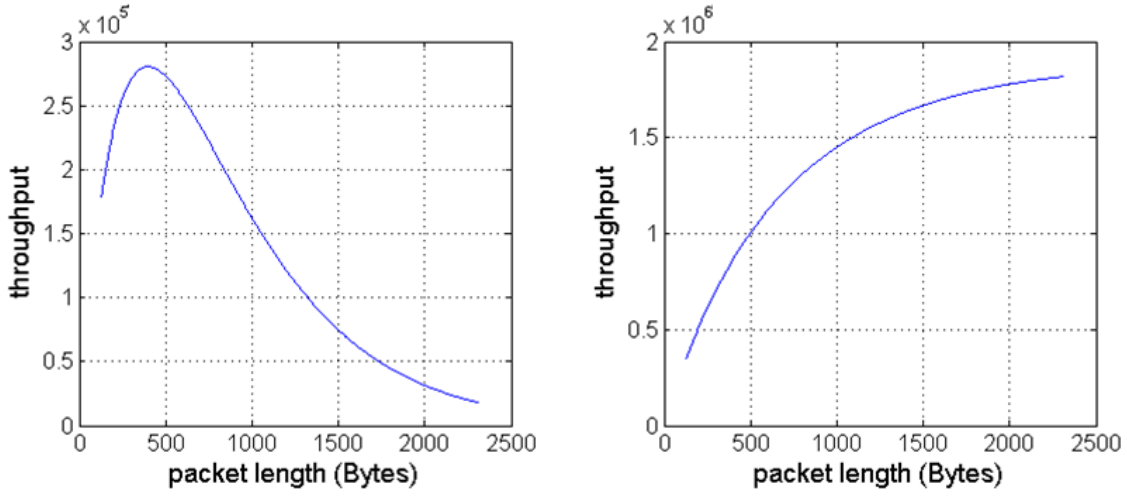
For the simulations in this chapter, we assume SNR to have a log-normal distribution, where the mean is dependent on the path loss, and the variance is dependent on the type of environment [34].

It is important to discuss the impact of using this channel error model compared to the commonly used constant BER model. A constant BER model treats $f_\sigma(s)$ as a delta function, so that the BER is the same for all packets. As a result, it tends to overestimate the impact of packet length on loss probability because it ignores the impact of the SNR distribution. To demonstrate this, we plot theoretical throughput as a function of packet length for the two different loss models with the same average SNR of 9dB in Figure 6.1. Figure 6.1(a) assumes a constant BER, and Figure 6.1(b) assumes SNR to have a log-normal distribution. The shapes of the curves are noticeably different. This is because in the case where SNR is modeled probabilistically, the actual value of SNR has a much higher impact on an individual packet's successful transmission than the packet's length. While a constant BER model might suggest using a packet length of only 400 bytes, a more accurate model including SNR distribution shows that maximum packet length would be superior.

6.2 Analytical Throughput Computation

The throughput, TP, of a wireless node can be modeled as

$$TP = L \cdot S \cdot (1 - P_L) \tag{6.3}$$



(a) (b)

Figure 6.1. *Throughput vs packet length assuming no collisions for (a) SNR fixed at 9dB, and (b) SNR with mean of 9dB with standard deviation of 3dB.*

L is the length of the packet payload in bits, S is the number of packets sent per second, and P_L is the overall packet loss probability. S can be expanded into

$$S = \frac{1}{\overline{W}(P_L) + \overline{T}_B + T_{ov} + L/R} \tag{6.4}$$

where $\overline{W}(P_L)$ is the average amount of time spent in exponential backoff per packet, which depends on P_L; $\overline{T}_B$ is the average amount of time the channel is busy at the station; T_{ov} is the amount of per-packet overhead including headers, interframe spacing, and ACK time, and R is the modulation rate. $(1 - P_L)$ can be expanded into

$$(1 - P_L) = (1 - P_{SC2}) \cdot (1 - P_{DC}) \cdot (1 - P_{SC1}) \cdot (1 - P_e) \tag{6.5}$$

where P_{SC2}, P_{DC}, P_{SC1}, and P_e are the probabilities of staggered collision of type 2, direct collision given no staggered collision of type 2, staggered collision of type 1 given no staggered collision of type 2 or direct collision, and channel error given no collision, respectively.

Since we are only adapting the packet length, the terms for P_{SC2} and P_{DC} are unimportant, and can be lumped into a constant C, because they do not depend on packet length to first order. Specifically, they only depend on the probability the channel is busy prior to or at the same time as a transmission start, regardless of the length of the transmission.

We can thus reduce Equation (6.3) to the product of five factors, four of which depend on L:

$$TP = C \cdot L \cdot S \cdot (1 - P_{SC1}) \cdot (1 - P_e) \tag{6.6}$$

where S is computed as in Equation (6.4), and P_e is computed as in Equation (6.2). Many of the variables in these expressions are unknown and vary with time, particularly as other nodes adapt their packet lengths. Thus it is difficult to determine a single value of L to maximize this function analytically. However, it is possible to estimate the derivative of TP with respect to L for the current network conditions given the BI signal.

For a function $F(x) = \prod_{i=1}^{n} f_i(x)$, the derivative with respect to x can be computed as:

$$F'(x) = \sum_{i=1}^{n} \left(f_i'(x) \prod_{j \neq i} f_j(x) \right) = F(x) \sum_{i=1}^{n} \frac{f_i'(x)}{f_i(x)} \tag{6.7}$$

Therefore, to compute the derivative of the throughput with respect to L, we must compute the derivatives of each of the four terms in the product in Equation (6.6), and substitute them into the following expression:

$$\frac{\partial}{\partial L} TP = TP \cdot \left(\frac{1}{L} + \frac{\frac{\partial}{\partial L} S}{S} + \frac{\frac{\partial}{\partial L}(1 - P_{SC1})}{1 - P_{SC1}} + \frac{\frac{\partial}{\partial L}(1 - P_e)}{1 - P_e} \right). \tag{6.8}$$

The first term is easily computed because L is a known quantity, and its derivative is simply 1. In the second term, S can be estimated empirically at each node by counting the number of packets transmitted over a fixed period of time. Its derivative can be computed as

$$\begin{aligned} \frac{\partial}{\partial L} S &= -\frac{\frac{\partial}{\partial L} \overline{W}(P_L) + \frac{1}{R}}{(\overline{W}(P_L) + \overline{T}_B + T_{ov} + L/R)^2} \\ &= -S^2 \cdot \left(\frac{\partial}{\partial L} \overline{W}(P_L) + \frac{1}{R} \right). \end{aligned} \tag{6.9}$$

where $\frac{\partial}{\partial L} \overline{W}(P_L)$ can be computed from the chain rule as

$$\frac{\partial}{\partial L} \overline{W}(P_L) = \frac{\partial}{\partial P_L} \overline{W} \cdot \frac{\partial}{\partial L} P_L. \tag{6.10}$$

$\overline{W}$ is a deterministic function of the number of attempts, which is in turn a function of P_L. It is straightforward to empirically estimate this function $\overline{W}(P_L)$ as well as its derivative $\frac{\partial}{\partial P_L} \overline{W}$, which we call $p(P_L)$. An empirical count of P_L can be plugged into $p(P_L)$ to obtain an estimate of $\frac{\partial}{\partial P_L} \overline{W}$.

For the second factor in the right-hand side of Equation (6.10), we also have

$$\begin{aligned} \frac{\partial}{\partial L} P_L &= -\frac{\partial}{\partial L}(1 - P_L) \\ &= -(1 - P_L) \cdot \left(\frac{\frac{\partial}{\partial L}(1 - P_{SC1})}{1 - P_{SC1}} + \frac{\frac{\partial}{\partial L}(1 - P_e)}{1 - P_e} \right) \end{aligned} \tag{6.11}$$

where the second equality comes from applying Equation (6.7) to Equation (6.5) and noting that P_{SC2} and P_{DC} are independent of L. The two terms in the sum are the same as the

last two terms in Equation (6.8), so it is possible to substitute Equations (6.9), (6.10), and (6.11) into (6.8) and re-arrage to obtain

$$\begin{aligned}\frac{\partial}{\partial L}TP &= TP \cdot \Big[\tfrac{1}{L} + \tfrac{S}{R} \\ &\quad +(S \cdot p(P_L) \cdot (1 - P_L) + 1) \\ &\quad \cdot \Big(\tfrac{\frac{\partial}{\partial L}(1-P_{SC1})}{1-P_{SC1}} + \tfrac{\frac{\partial}{\partial L}(1-P_e)}{1-P_e}\Big)\Big]\end{aligned} \tag{6.12}$$

Now all that remains is to estimate the last two terms. The second to last term can also be computed using the chain rule:

$$\frac{\partial}{\partial L}(1 - P_{SC1}) = \frac{\partial}{\partial l}(1 - P_{SC1}) \cdot \frac{\partial}{\partial L} l \tag{6.13}$$

where $l = L/R$ is the length of the packet in seconds. It is shown in Chapter 2 that $\frac{\partial}{\partial l}(1-P_{SC1})$ can be approximated by a quantity denoted by m_2, and which can be estimated as part of the collision probability estimation process. Thus Equation (6.13) can be reduced to

$$\frac{\partial}{\partial L}(1 - P_{SC1}) = -\frac{m_2}{R}. \tag{6.14}$$

To obtain the final term in Equation (6.12), we must take the derivative of Equation (6.2) with respect to L. This results in:

$$\begin{aligned}&\tfrac{\partial}{\partial L}(1 - P_e) \\ &\quad = \textstyle\int (1 - P^p_{e,h}(s)) \tfrac{\partial}{\partial L}(1 - BER_{R_p}(s))^L f_\sigma(s) ds \\ &\quad = \textstyle\int (1 - P^p_e(s)) \ln(1 - BER_{R_p}(s)) f_\sigma(s) ds \\ &\quad \approx -\textstyle\int (1 - P^p_e(s)) BER_{R_p}(s) f_\sigma(s) ds.\end{aligned} \tag{6.15}$$

The first equality comes from switching the derivative into the integral and noting that $P^p_{e,h}(s)$ does not depend on L; the second equality comes from taking the derivative of the exponential term; the final approximation comes from a Taylor series expansion of $\ln(1-x)$ for x close to zero. This can be justified for large s, because BER is low for high SNR. Even though the Taylor approximation becomes less accurate as s decreases, the terms in the integral for smaller s have lower weight because as s decreases, $(1 - P^p_e(s)) \to 0$. Therefore the approximation in Equation (6.15) is reasonable for all values of SNR. The final expression of Equation (6.15) depends only on the distribution of SNR. The node can thus estimate $\frac{\partial}{\partial L}(1 - P_e)$ using only an estimate of the current SNR distribution.

If the SNR distribution $f_\sigma(s)$ is a single-parameter distribution such as Rayleigh or a log-normal with known variance, then it can be estimated by each node, based on the estimate of P_e, and via a lookup table based on Equation (6.2). If it is a two-parameter distribution, it requires observation of another variable dependent on the SNR distribution, such as a received signal strength indication (RSSI). In the simulations of this chapter we assume a log-normal distribution with a known variance. This variance depends on the type

of environment and is assumed to be either pre-set by the deployer or learned from the variation in signal strength of beacon packets.

P_e is estimated via Equation (6.1) where P_L is estimated via empirical counting and P_C is estimated via the technique described in Chapter 2. A lookup table generated from Equation (6.2) is then used to estimate $f_\sigma(s)$, which is used to estimate $\frac{\partial}{\partial L}(1-P_e)$ via Equation (6.15).

Substituting Equation (6.14) into Equation (6.12), we get a final expression:

$$\begin{aligned} \frac{\partial}{\partial L}TP &= TP \cdot \Big[\frac{1}{L} + \frac{S}{R} \\ &\quad +(S \cdot p(P_L) \cdot (1-P_L) + 1) \\ &\quad \cdot \Big(-\frac{m_2}{R \cdot (1-P_{SC1})} + \frac{\frac{\partial}{\partial L}(1-P_e)}{1-P_e}\Big)\Big] \end{aligned} \tag{6.16}$$

where L is known, S and P_L are estimated via empirical counting, m_2 is estimated as a part of the collision probability estimation, P_e is computed from Equation (6.1) where P_C is estimated via the technique in Chapter 2, and $\frac{\partial}{\partial L}(1-P_e)$ is computed via lookup table based on P_e.

6.3 Adaptation Algorithm

We assume an 802.11 network in infrastructure mode with multiple APs, each with several associated nodes, similar to the topology in Chapter 2. APs periodically broadcast medium occupancy statistics to all associated nodes, which the nodes use to estimate their probabilities of each type of loss and adapt their packet lengths. Specifically, nodes transmit at a constant packet length L for 5 seconds, and over this period the nodes and APs collect their medium occupancy statistics. The AP broadcasts its statistics to the nodes, which then use this information along with local observations to estimate S, P_L, P_C, P_e and $\frac{\partial}{\partial L}TP$. The nodes then choose the new packet length L_{next} to be used for the next 5 second period to be

$$L_{next} = L + \alpha \cdot \frac{\partial}{\partial L}TP \tag{6.17}$$

where α is a step size factor updated according to the following schedule:

- if $sign(\frac{\partial}{\partial L}TP)$ changed, decrease α by factor of λ
- if $sign(\frac{\partial}{\partial L}TP)$ not changed, increase α by factor of γ

By using this schedule, nodes use a large coefficient to rapidly adjust packet length into the appropriate neighborhood, and then decrease the step size to converge to a precise final packet length. Appropriate choices of λ and γ can increase convergence rate and avoid oscillation. There is an additional caveat that if L is equal to L_{min} or L_{max}, α should not

change. This is because, for example, when $L = L_{max}$ and the $\frac{\partial}{\partial L}TP$ is positive, the node cannot increase L to the point where the derivative's sign changes. Thus, as long as no other nodes change their behavior, $\frac{\partial}{\partial L}TP$ continues to be positive at every iteration, causing α to grow exponentially. With such a large value of α, the node would no longer be able to appropriately adapt to future changes in network conditions; in particular, if the optimal packet length were to change, the node would have excessive oscillations until α decreases to a reasonable value. We additionally set a maximum step size max_step, and require L to remain between L_{min} and L_{max}.

Algorithm 4 The packet length adaptation algorithm

loop
  Transmit at current rate for 5 seconds
  Observe S and P_L over last 5 seconds
  Receive BI signal from AP
  Compute estimates of P_C and m_2 using BI signal
  Estimate P_e from P_L and P_C via Equation (6.1)
  Compute estimate of $\frac{\partial}{\partial L}TP$ via Equation (6.16)
  if $sign(\frac{\partial}{\partial L}TP) \neq last\_dir$ **then**
    $\alpha \leftarrow \alpha/\lambda$
    $last\_dir \leftarrow sign(\frac{\partial}{\partial L}TP)$
  else if $L \in (L_{min}, L_{max})$ **then**
    $\alpha \leftarrow \alpha \cdot \gamma$
  end if
  $\Delta L \leftarrow sign(\frac{\partial}{\partial L}TP) \cdot min(|\frac{\partial}{\partial L}TP| \cdot \alpha, max\_step)$
  $L \leftarrow median(L_{min}, L + \Delta L, L_{max})$
end loop

6.4 Simulation results

To test the throughput gains achievable using our length adaptation algorithm, we use the NS-2 simulation package with modifications as described in Chapter 2 as well as modifications to execute the adaptation algorithm. The results in this chapter are for 802.11b, but they can easily be extended to 802.11a or g, or to any other carrier-sense multiple access MAC with multiple available modulation rates. The test topology consists of 7 APs arranged to

cover hexagonal cells, and 50 nodes placed at random over the area by a spatial poisson process sending saturated traffic to the nearest AP. SNR standard deviation is assumed to be 7dB, which is standard for an office environment with hard partitions [12].

To verify the accuracy of our derivative estimation, we fix the packet length of all but one node in the network, and vary the packet length of the single node in 2000 bit increments for the single node and measure total throughput over 4 minutes of simulation time for each packet length, thus obtaining a throughput vs. packet length curve. We repeat this for 16 different nodes in each of 10 different topologies. A plot for a representative node is shown in Figure 6.2 as the blue dash-dot curve. From our simulation data, we also compute the $\frac{\partial}{\partial L}TP$ estimates for each scenario and include it in the same figure. The estimates are shown as the dotted red curve, while the solid green curve is computed as the average slope over three consecutive data points in the throughput curve. The important feature to note in this figure is the point at which the curves for $\frac{\partial}{\partial L}TP$ cross zero, which is represented by the solid black line, because the algorithm converges to this packet length. Even though the estimate of the derivative is inaccurate for small packet lengths, it is quite accurate around the optimal throughput. In this example, the optimal packet length is less that the maximum packet length. However, there are many cases in which maximum packet length is optimal. In particular, for nodes closer to their APs at lower noise levels, probabilities of channel errors and staggered collisions are both low, making longer packet lengths preferable. In all cases where throughput strictly increases with length, the algorithm converges to maximum packet length.

Having verified that the algorithm works for single-node adaptation, we now allow all nodes to adapt simultaneously. Here the dynamics of our algorithm come into play. We have empirically found that initializing α to 2000, and setting $\lambda = 2$, $\gamma = 1.25$, and $max\_step = 4000$ allows nodes to converge rapidly. We run the algorithm on all nodes in 7 different random topologies, each at 5 different level of ambient noise for a total of 35 simulations.

Figure 6.3 shows the results of each of the 35 simulations. Each symbol represents one topology at one noise level. The shape and color represent the noise level. The x position represents the number of nodes selecting non-maximal packet lengths for that scenario, and the y position represents the total throughput gain of the network compared to the situation where all nodes send at maximum packet length. It can been seen that as ambient noise increases, the number of nodes choosing shorter packet length increases. This is consistent with intuition, as higher noise leads to higher loss probability, which can be combatted by choosing shorter packet lengths. It is also observed that as noise power increases, throughput gain increases. This is because as noise and the packet loss rate increase, maximum packet length becomes increasingly sub-optimal for an increasing number of nodes. At lower noise powers, our algorithm results in most nodes choosing maximum packet length, which is optimal. Thus the total throughput it very close to when packet length is fixed at maximum for all nodes.

Figure 6.4 shows the converged packet lengths of various nodes for one scenario with

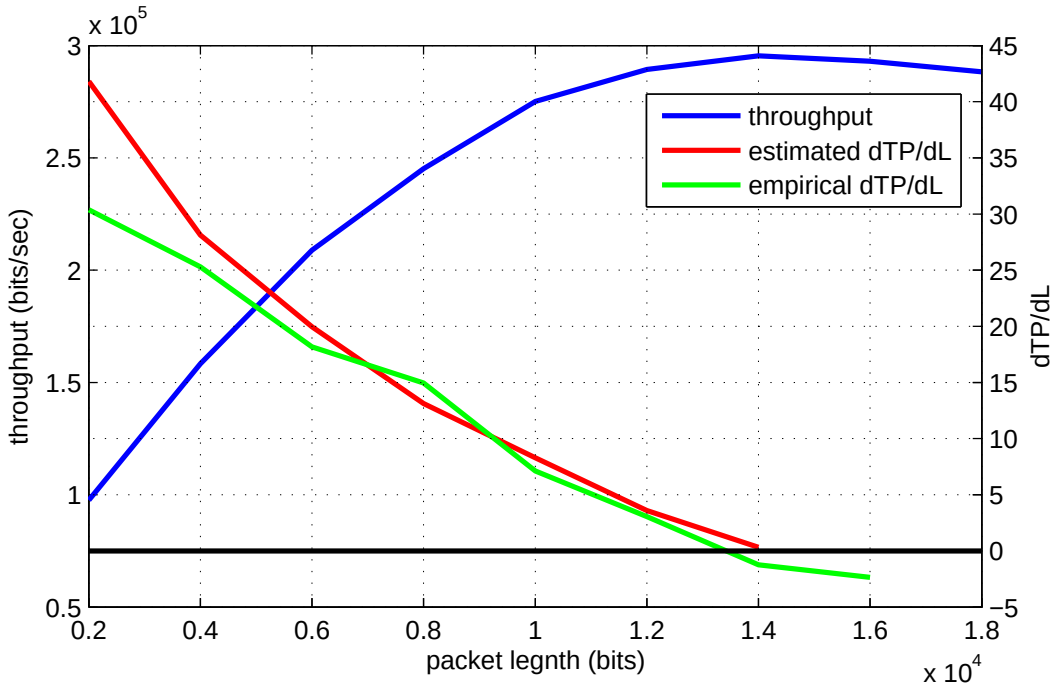


Figure 6.2. *Throughput and its derivative as a function of packet length, for a single node with all other nodes sending at constant packet length.*

a noise power of -95dBm. The black squares represent the location of the APs, and the circles represent the locations of the nodes. The color of the center of the circle indicates the final converged packet length after 5 minutes of simulation time. Dark red is maximum packet length, while blue and green represent shorter packet lengths. The border of the circle represents the average packet length over the simulation time. Significant difference between this color and the color of the center of the circle, this indicates slower convergence, i.e. 10-15 iterations. Depending on their specific locations, some nodes rapidly converge to maximum packet length while other nodes converge to other packet lengths. As seen, 68% of the nodes converge to maximum packet length in 1-2 iterations. The nodes which converge to shorter packet lengths tend to be closer to the borders of the topology. This is because these nodes suffer the most from the hidden node problem as well as weak signal to the AP, and hence have higher packet loss rates, thus making it more beneficial to select shorter packet lengths.

Figure 6.5(a) shows the converged packet lengths for the same topology for a higher noise power of -89dBm. As seen, more nodes choose non-maximal packet lengths in the higher noise scenario, with only 46% choosing near-maximum packet length. This is because at the higher noise power, packet loss becomes a greater concern than overhead for a greater proportion of the nodes. Figure 6.5(b) shows the percent throughput gain compared to the scenario using fixed maximum packet length for each node. Black squares represent the locations of the APs. The center of each circle corresponds to the location of each node,

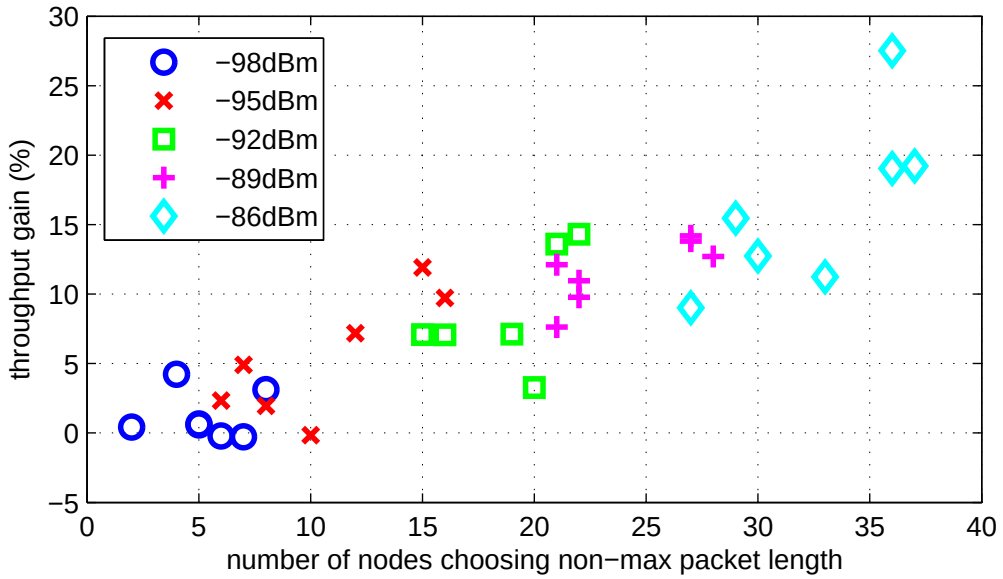


Figure 6.3. *Throughput gain and number of nodes choosing non-max packet length for 7 different topologies, each at 5 different noise powers.*

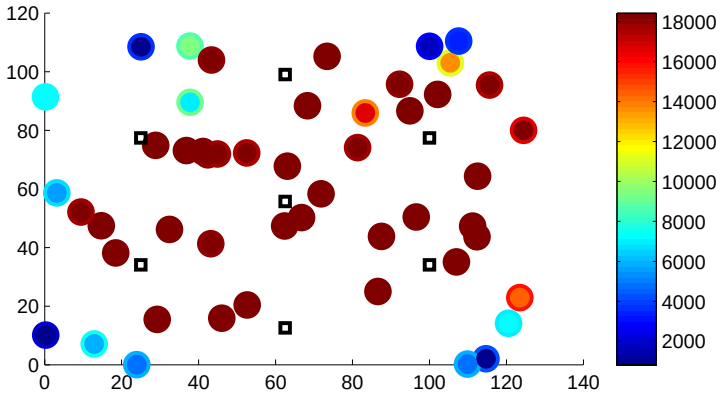


Figure 6.4. *Spatial plot of adapted packet length for each node in a scenario with -95dBm noise.*

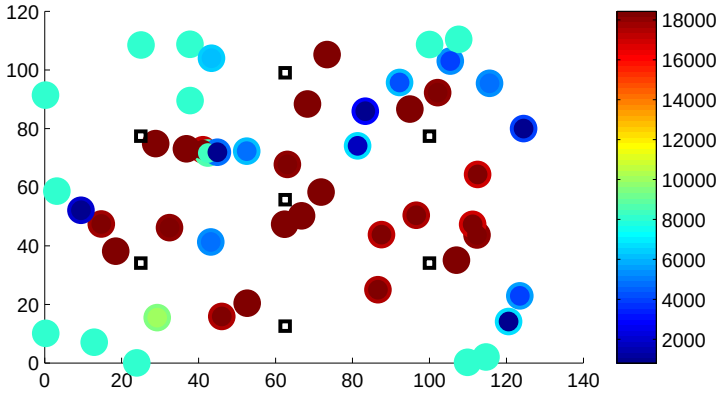
which is the same as in Figure 6.5(a). A node is colored green if it gains throughput and red if it loses. The size of the circle is proportional to the percent throughput change. As seen, some of the nodes choosing short packet lengths experience tremendous throughput improvements, while others have more moderate gains or even losses. Nodes which select maximum packet lengths are also affected by the changed packet lengths of their neighboring nodes, because they acquire the channel more frequently and experience fewer collisions with the nodes sending shorter packets. This coupling behavior between neighboring nodes could potentially explain the fact that gains and losses tend to be localized to certain areas of the

network. For example, the nodes in the lower-right of this topology experience large gains, while the nodes in the middle experience moderate losses. Figure 6.5(c) shows the absolute throughput of each node in the network. There are two squares plotted for each node. The size of the red square is proportional to the throughput of the node when all nodes use maximum packet length, and the size of the green square is proportional to the throughput of the node when all nodes adapt. As seen, the 5 nodes with the greatest throughputs all have moderate throughput improvements. The 3 nodes with the largest percentage gains seen in Figure 6.5(b) are all in the bottom 40% in terms of overall throughput.

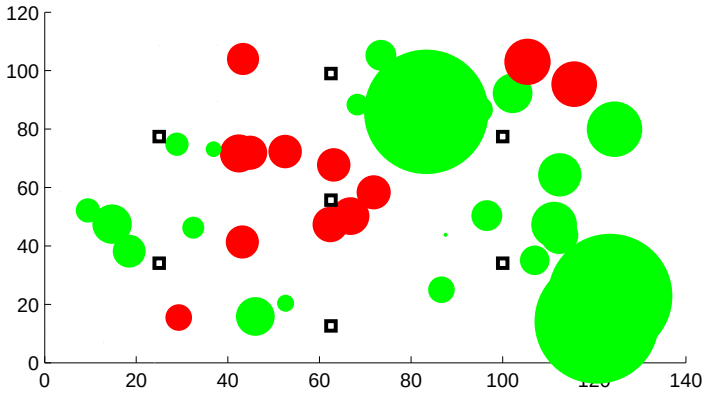
6.5 Conclusions

In this chapter we have demonstrated the importance of using a detailed packet loss model for 802.11 networks, including channel fading and staggered collisions. We have derived analytical expressions for throughput and its derivative as a function of packet length, and shown that it can be estimated using our collision probability estimation technique. We have proposed a packet length adaptation algorithm, and shown that it can achieve up to 20% throughput gains.

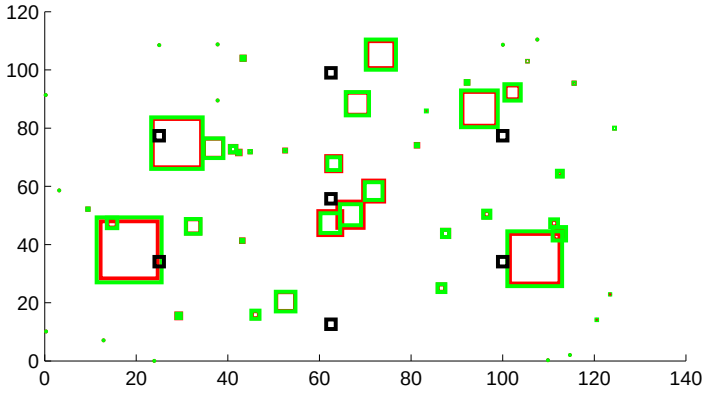
In our method, packet length is not adapted rapidly enough to track fast timescale channel changes; rather it is set over a long period of time, i.e. 5 seconds, to achieve the best overall performance for the distribution of SNRs that are likely to occur over that time period. This will likely have interactions with adapted modulation rate at a fast timescale, but these are not taken into account in this paper. Although packet length adaptation cannot track these rapid channel changes, intelligent packet length selection based on an estimation of the channel distribution can yield gains with respect to using maximum packet length. Future work involves investigating these challenges, particularly in the context of newer WLANs standards such as 802.11n and 802.11ac, which use packet aggregation to get effectively longer packets.



(a)



(b)



(c)

Figure 6.5. *A spatial plot of (a) converged packet length, (b) percent improvement over non-adaptive packet length, and (c) absolute throughput for a scenario with -89dBm noise.*

Chapter 7

Carrier Sense Threshold Adaptation

Two of the most well-known problems with a CSMA MAC protocol are the hidden and exposed node problems. The hidden node problem occurs when the receiver cannot successfully receive the transmitted packet due to interference caused by another transmission initiated by a node located outside the sensing range of the transmitter. The transmitter cannot sense the hidden node if its CST value is larger than the received energy level of the signal transmitted by the hidden node. Conversely, the exposed node problem occurs when a transmitter needlessly avoids transmission in order to avoid interference with another concurrent transmission, even though the interference at its receiver is lower than the decoding threshold. In this case, CST of the transmitter is small enough that it senses faraway transmissions, thus unnecessarily deferring transmission. Both of these problems can lead to reduced throughput, but both are impacted differently by the CST; as such, there is an inherent tradeoff between the number of hidden and exposed nodes depending on the CST value chosen by a given station. While decreasing CST makes the station sense nodes further away and reduces the number of hidden nodes, it also increases the number of exposed nodes unnecessarily sensed by the station. Therefore, for each node there is an optimum CST that balances the impact of the hidden and exposed nodes. In this Chapter, we discuss the relationship between the CST and the BI signal and propose a method for the stations to adaptively select the CST so as to minimize the occurrence of the hidden and exposed node problems. In Section 7.1, we discuss the relationship between the BI signal, CST, and hidden and exposed node problem. We propose a CST adaptation algorithm in Section 7.2, present NS-2 simulation results in Section 7.3, and discuss conclusions in Section 7.4

7.1 Effects of CST

Up to this point, we have considered the BI signal for a fixed CST. Specifically, each station deduces its channel occupancy at a time slot by comparing its received energy level at that time slot with its CST. If the energy level is higher than the threshold, the station assumes the channel to be busy in that time slot; otherwise, the channel is assumed to be idle. Thus, the CST plays a large role in determining the BI signal, and by adjusting its CST, a station can alter its BI signal.

The key idea is that stations must decide whether or not to transmit based on observations of their local channel. In particular, when the received energy of their channel is less than their CST, stations transmit to the AP, otherwise they back off. However, successful reception at the AP depends on the AP's channel status in that time slot. The stations can use the BI signal from the AP to obtain information about the channel status at the AP. Therefore, it is desirable for each station to choose its CST such that its resulting local BI signal mimics the BI signal of the receiving AP. The value of the CST at each station directly affects the existence and number of the hidden and exposed nodes it experiences. Due to the inherent trade-off between the hidden and exposed nodes, increasing the CST at the station decreases the number of the exposed nodes while increasing the number of hidden nodes while decreasing the CST decreases the number of the hidden nodes while it increases the number of the exposed nodes.

Stations can compare the received BI signal from the AP with their own energy profile over the sampling period to gain knowledge about the existence of hidden and exposed nodes. For example, if for a given time slot the station senses the channel to be busy but the AP claims otherwise, the station can deduce that the activity of an exposed node in that time slot has resulted in this difference. On the other hand, the station can detect the existence of a hidden node if it senses the channel idle at a time slot while the AP reports a busy channel. We use this feature of the busy/idle signal to derive an adaptive algorithm for choosing the carrier sense threshold.

Fig. 7.1 shows a sample schematic of the recorded energy profile of a station and the binary BI signal transmitted by the AP. The energy profile at the station has multiple levels while the BI signal of the AP is a binary signal. The highlighted slots in the station energy profile indicate transmissions of exposed nodes and the highlighted slots in the BI signal of AP indicate transmissions of hidden nodes. n_{01} and n_{10} represent the number of slots affected by transmissions of hidden and exposed nodes respectively. Increasing the CST from L_1 to L_4 in Fig. 7.1(a) to Fig. 7.1(d) changes the total number of slots affected by transmissions from the hidden and exposed nodes, $n_{01} + n_{10}$, from 4 to 3, to 5, and to 8 with the minimum occurring at $L_1 < CST < L_2$. Since both hidden and exposed nodes can potentially lower the throughput of the network, it is desirable for each station to choose its CST so as to minimize the number of time slots in which its hidden and exposed nodes affect its transmissions[48]. Since different stations experience different number of hidden

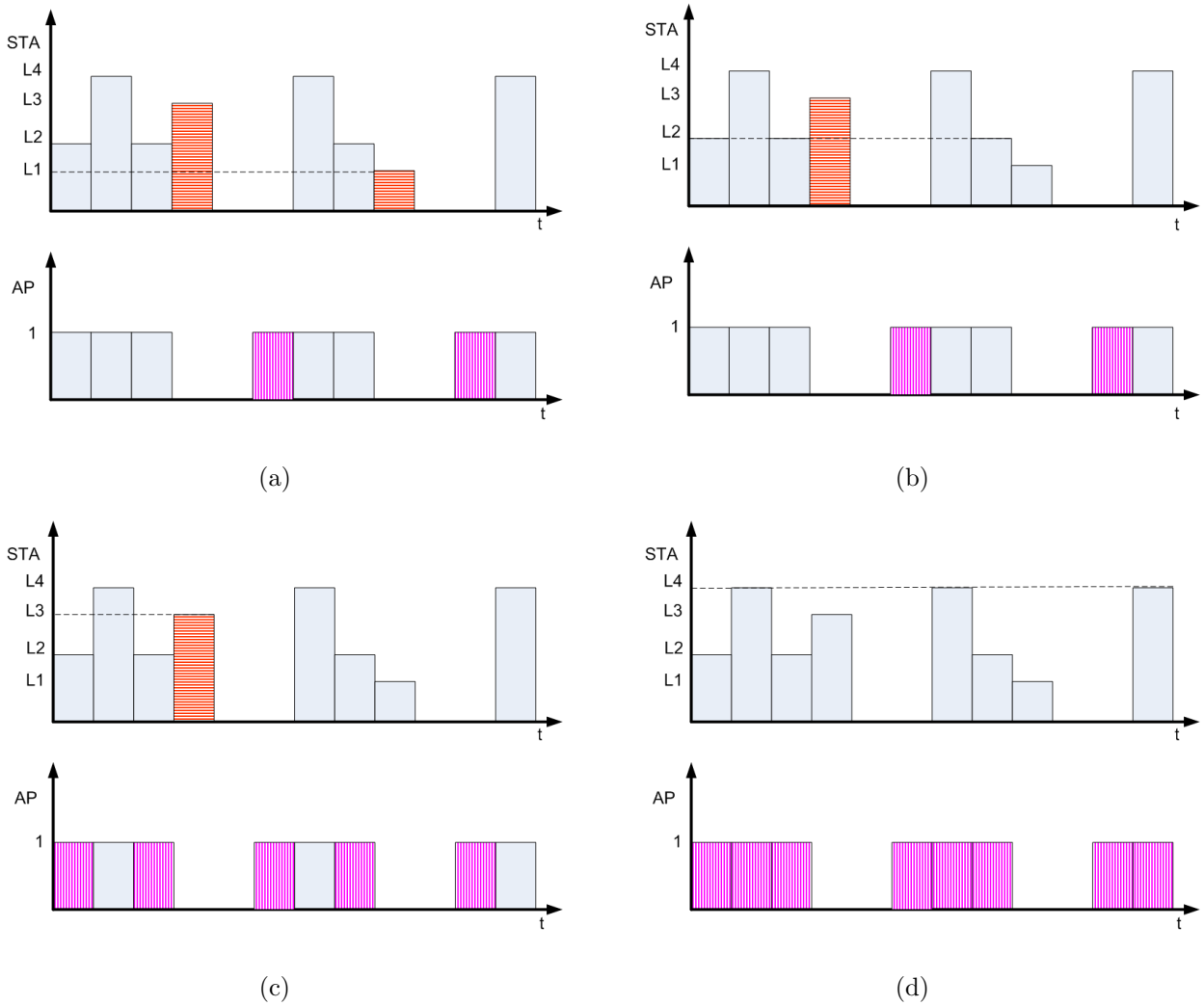


Figure 7.1. BI signal of AP shown at the bottom and energy profile of a station shown on top. (a) $CST < L_1$; $n_{01} = 2$; $n_{10} = 2$. (b) $L_1 < CST < L_2$; $n_{01} = 2$; $n_{10} = 1$. (c) $L_2 < CST < L_3$; $n_{01} = 5$; $n_{10} = 1$. (d) $CST > L_4$; $n_{01} = 8$; $n_{10} = 0$.

or exposed nodes according to their locations, using a constant CST for all nodes is not optimal. We elaborate more on this later in Section 7.3.

7.2 Adaptation Algorithm

In a network with multiple access points, we assume each station connects to the AP with the highest received signal power. Each access point broadcasts its binary BI signal every Δ seconds to indicate its channel occupancy for all time slots of δ second duration. Each station records the received energy level for all time slots in the period of Δ seconds as well. By comparing the energy level with the carrier sense threshold γ, the station generates its own BI signal $BI^{\gamma}_{station}$. Specifically, for $p, q \in \{0, 1\}$, we define

$$P^{\gamma}_{pq}(t) = Pr\{BI^{\gamma}_{station}(t) = p, BI_{AP}(t) = q\} \tag{7.1}$$

where t is the time slot number, and $1 \leq t \leq \lfloor \frac{\Delta}{\delta} \rfloor$. It is assumed that p or q are equal to 0 if the channel is idle, and 1 if the channel is busy. As mentioned earlier, the activities of hidden or exposed nodes result in a difference in the observed channel occupancy at the AP and station. In particular the numbers of transmissions of hidden and exposed nodes are proportional to $P^{\gamma_i}_{01}$ and $P^{\gamma_i}_{10}$ respectively for $CST = \gamma_i$. Specifically,

$$|Transmissions\ of\ Hidden\ \ Nodes| \propto P^{\gamma_i}_{01} \tag{7.2}$$

$$|Transmissions\ of\ Exposed\ Nodes| \propto P^{\gamma_i}_{10} \tag{7.3}$$

To maximize the throughput of the network, it is reasonable to minimize the number of transmissions of hidden and exposed nodes for each station by changing its CST. We therefore define the optimization function F as the sum of the number of hidden nodes and exposed nodes for each station during Δ:

$$\gamma_{opt} = \arg\min_{\gamma_i} F_i(\gamma_i) \tag{7.4}$$

where

$$F_i(\gamma_i) = P^{\gamma_i}_{01} + P^{\gamma_i}_{10}. \tag{7.5}$$

Since the AP broadcasts its BI signal every Δ seconds, each station can incorporate an adaptive algorithm for adjusting its CST every Δ seconds i.e. every iteration. In our proposed adaptive algorithm, each station listens to the channel and records the energy level at all time slots. After receiving the binary busy/idle signal from the AP, the station performs an exhaustive search to choose the best carrier sense threshold, γ_{opt} from a finite set of possible values for the carrier sense, $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_N\}$. For each value of γ_i, the

station has to compute the corresponding $P_{01}^{\gamma_i}$ and $P_{10}^{\gamma_i}$ to evaluate the optimization function $F_i(\gamma_i)$.

As shown in Section 7.3, it is possible that different carrier sense thresholds result in similar values of the optimization function. To avoid excessive variations of CST, we allow the station to adhere to its current CST value if the percentage change between the new optimum value of F and the function value corresponding to the current CST is less than a constant value, ρ. Furthermore, if the percentage change is higher than ρ, the station chooses the closest CST value to its current CST whose difference from the optimum CST is less than ρ percent. Since variation of CST for each node affects the performance of other nodes in the network, this approach makes the network evolve in a more stable fashion, and prevents chaotic situations in which all the nodes significantly change their CST in every iteration. The pseudo-code for the adaptive selection of the carrier sense threshold is shown in Algorithm 5.

7.3 Simulation Results

In this section, we evaluate the performance of our proposed solution for selecting CST. We study an IEEE 802.11b network in which all the stations implement the adaptive carrier sense threshold algorithm, and compare its aggregate throughput to that of the same network with all stations using a constant CST. The APs use a constant CST in both cases. We modify the NS-2 simulation package to (a) allow stations record the energy level of the receiving channel at all time slots, and (b) allow the APs compute their BI signals and to periodically broadcast them to their associated stations. The network and simulation parameters are shown in Table 7.1. The position of the 7 APs is assumed to be unchanged from one simulation scenario to the next, while the random locations of the stations vary from one scenario to the next. The distance between adjacent APs is about 172 meters.

Since the APs locations are fixed, we only study the CS ranges which cover most of the area without APs overhearing each other. Table 7.2 shows the simulated CSTs in Watts and their corresponding coverage radius in meters.

A sample schematic of nodes and access points for a given simulation scenario is depicted in Fig. 7.2. The centers of the depicted circles represent the locations of the stations; the diameter of each circle is proportional to the percentage of throughput difference in each node when adaptive and fixed CST selection schemes are implemented. The empty blue circles show enhancement in throughput while red colored circles show loss of the throughput for that node.

Fig. 7.3 presents the histogram of percentage change in the throughput of each node for adaptive CST algorithm as compared to constant $CST = 7 \times 10^{-14}$. The histogram data is collected from 10 rounds of 60 second simulations, each of which includes 50 stations,

Algorithm 5 The Adaptive Carrier Sense Threshold Algorithm

Initialize $T \leftarrow \frac{\Delta}{\delta}$; $F_{opt} \leftarrow 1$; $tmp \leftarrow CST$; $N \leftarrow \|\Gamma\|$
Receive BI_{AP} from the AP
for $i = 1$ *to* N **do**
    $P_{01} \leftarrow \frac{1}{T}\sum_{t=1}^{T}[(Energy_{station}(t) \leq CST)\&\&(BI_{AP} == 1)]$
    $P_{10} \leftarrow \frac{1}{T}\sum_{t=1}^{T}[(Energy_{station}(t) \geq CST)\&\&(BI_{AP} == 0)]$
    $F[i] \leftarrow P_{01} + P_{10}$
    if $F[i] < F_{opt}$ **then**
        $tmp \leftarrow \gamma_i$
        $F_{opt} \leftarrow F[i]$
        $L \leftarrow i$
    end if
end for
$K \leftarrow k$ *s.t.* $(\gamma_k == CST)$
for j = K to L **do**
    $diff \leftarrow |F[j] - F_{opt}|$
    if $diff \leq \rho \cdot F_{opt}$ **then**
        $tmp \leftarrow \gamma_j$
        break
    end if
end for
$CST \leftarrow tmp$
return CST

i.e. overall 500 samples. In each simulation scenario, we use a different set of random locations for the stations, but the positions of APs are fixed. The figure shows that almost 90% of the nodes have improved their throughput. The median and average of throughput improvement of all stations are about 81% and 131% respectively. Our simulations have shown that those stations losing throughput could have improved their throughput if they were the only station in networking applying the adaptive CST algorithm and all other stations had used a constant CST. Since stations do not have any a priori information about activities of other stations, there is plenty of motivation for the stations to incorporate the adaptive solution in the hope of improving their throughput.

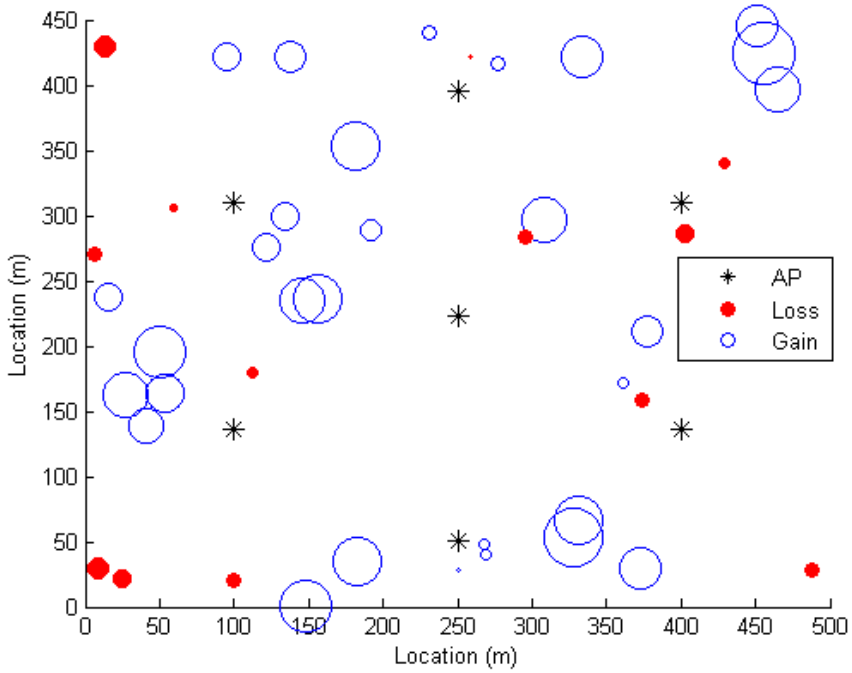


Figure 7.2. Throughput variation vs. location with adaptive CST.

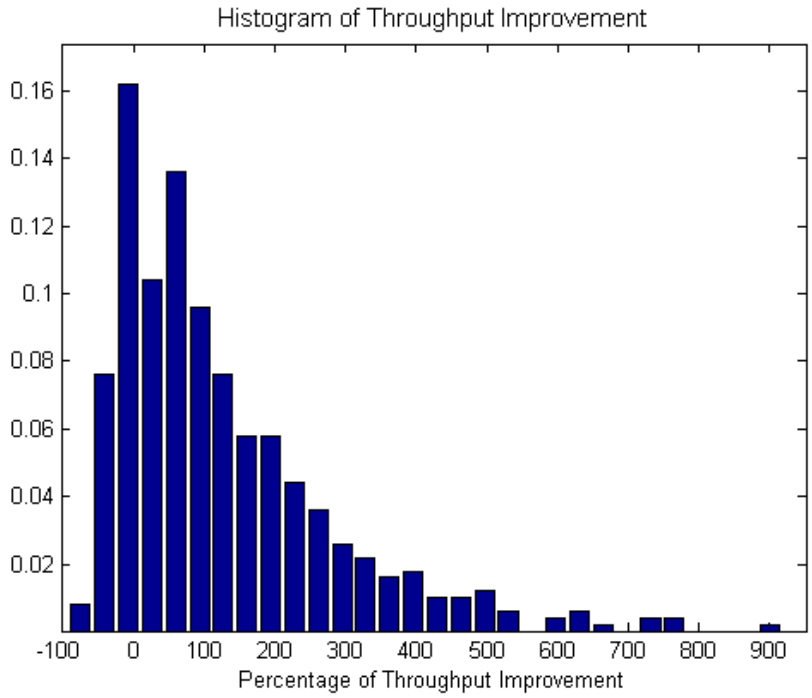


Figure 7.3. Histogram of percentage change in the throughput of stations for 10 scenarios with 50 stations per scenario.

Table 7.1. Simulation Parameters

Parameter	**Value**
No. Stations	50
No. AP	7
Area	500x450 (m)
Carrier Freq.	2.4 GHz
Transmit Power	32mW
Path Loss Factor	4
Bitrate	11 Mbps
Δ	3 sec
δ	$10\mu sec$
ρ	5%

Table 7.2. CST and Coverage

CST (W)	2e-13	7e-14	2e-14	7e-15	2e-16
Radius (M)	63	82	112	272	354

In our next experiment, we examine the aggregate throughput improvement for the various AP CST values listed in Table 7.2. In adaptive approach, all stations use the adaptive CST algorithm but the APs use constant CST and in non-adaptive approach, all stations and APs use the same constant CST. Fig. 7.4 compares the aggregate throughput of the network for both adaptive and non-adaptive schemes as a function of the default CST. Each value is obtained by averaging at least 10 different simulation scenarios of 60 seconds with random station locations varying from one scenario to the next. The percentage of throughput enhancement for each case is also presented in Fig. 7.4. As seen, the adaptive CST selection algorithm can increase the aggregate throughput of the network up to 50% for $CST = 7 \times 10^{-14}$, or decrease it by about 2% when $CST = 2 \times 10^{-16}$; in the latter case, each AP can sense the transmissions of at least 5 other APs and most of their surrounding stations. As such, it is not useful to study CST adaptation for stations in this case, because the APs have very poor spatial reuse.

To investigate the fairness of the adaptive CST algorithm, we study the *log-throughput*

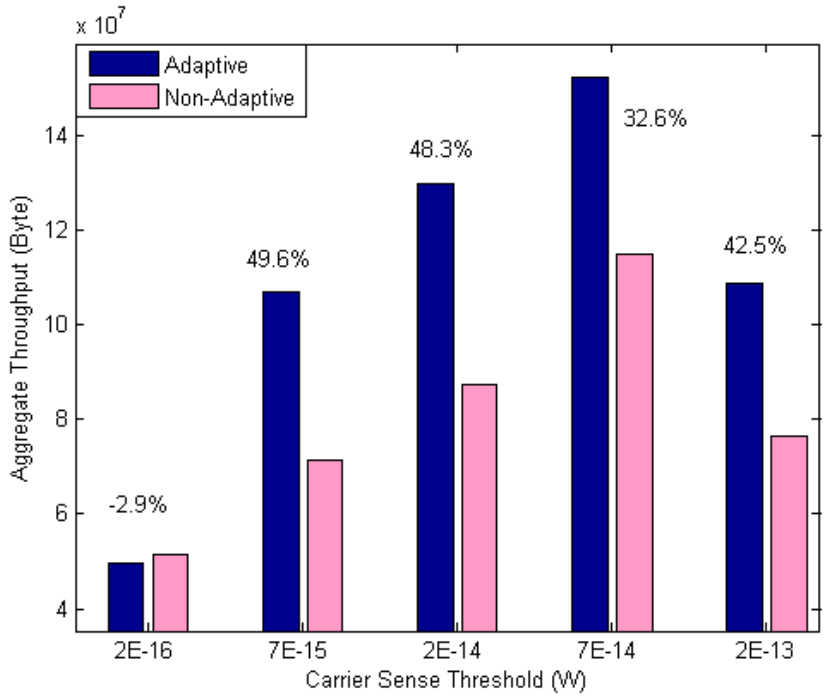


Figure 7.4. Aggregate throughput comparison for different CST values.

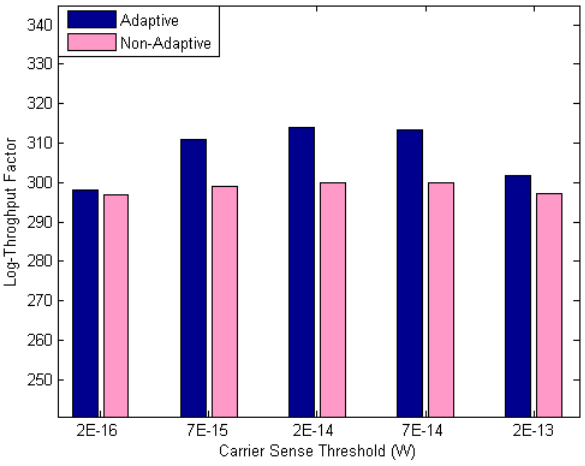


Figure 7.5. Log-Throughput comparison.

factor [20] in the network.

$$\Phi = \frac{1}{n}\sum_{j=1}^{n}\log(TP_j) \tag{7.6}$$

where Φ is the log-throughput of the network, TP_j is the throughput of station j, and n is the number of stations in the network. By studying the log-throughput, it is possible to determine whether the throughput enhancement of the network is due to unfairly increasing the throughput of the stations with better quality physical channels, or conversely resulting from fairly increasing the throughput of all nodes. Fig. 7.5 presents the log-throughput factor for each default CST. As seen, the average of log-throughput is increased in all scenarios, indicating that the adaptive CST algorithm behaves fairly.

Next, we study the probability of packet loss and number of transmission attempts for the stations in both adaptive and non-adaptive scenarios. Fig. 7.6 is a scatter plot of the probabilities of packet losses for all 500 stations in 10 simulation scenarios of 60 seconds each when the default CST is 7×10^{-14}. Since the majority of the points are to the left of the 45 degree line, we conclude that adaptive CST algorithm decreases the probability of packet loss. Since simulations for adaptive and non-adaptive scenarios are conducted under similar PHY conditions, decrease in the probability of packet loss primarily results from decrease in the collision probability. Fig. 7.7 compares the number of transmission attempts of stations throughout the simulation time for 10 scenarios with 50 stations per scenario. As seen, the number of transmission attempts is decreased for most stations using the adaptive CST algorithm. From Figs 7.6 and 7.7 we conclude that using adaptive CST algorithm improves the aggregate throughput by decreasing the probability of collision in the network and without increasing the amount of traffic. Thus in addition to improving throughput, our method also improves energy efficiency.

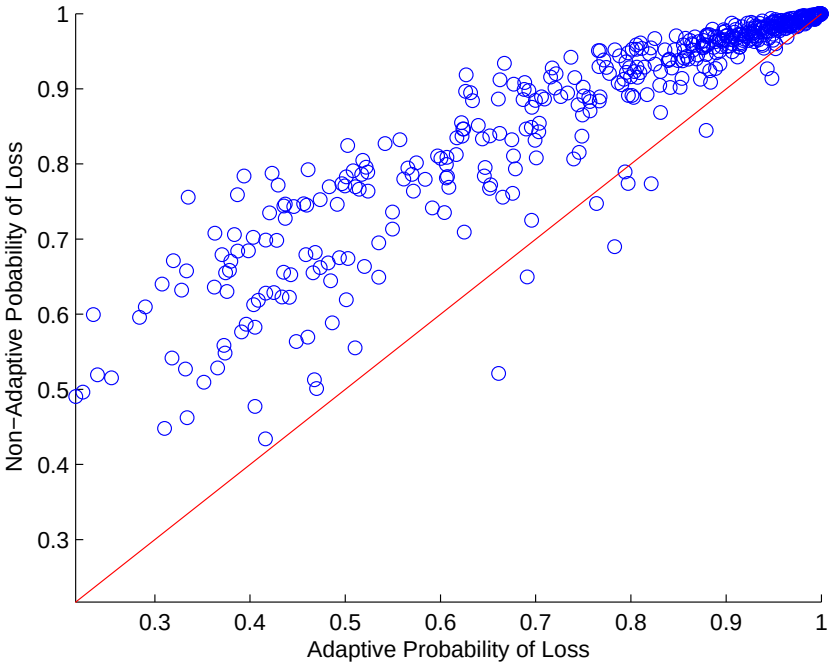


Figure 7.6. Probability of packet loss for 500 stations from 10 simulation scenarios.

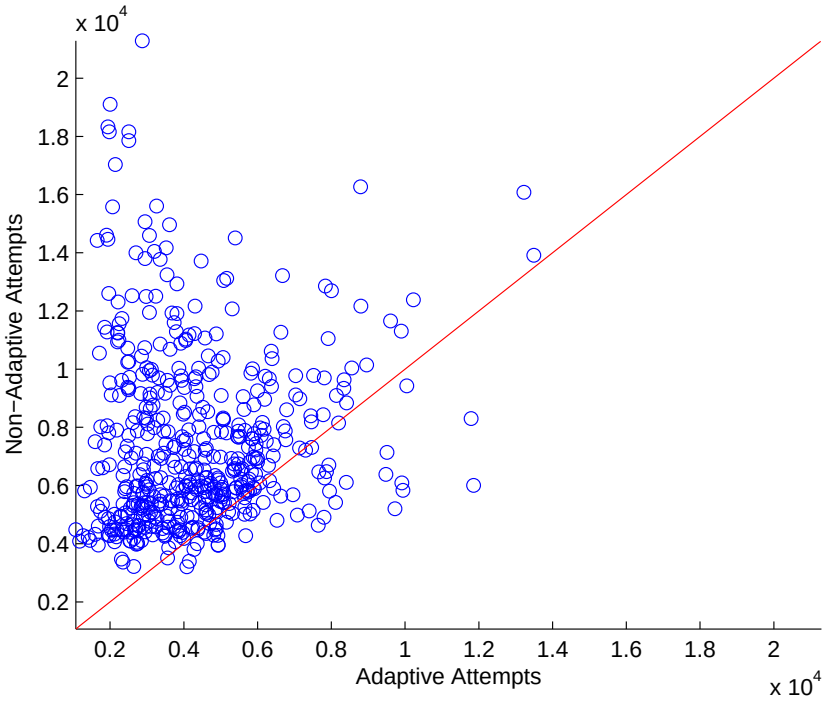


Figure 7.7. Transmission attempts for 500 nodes from 10 simulation scenarios.

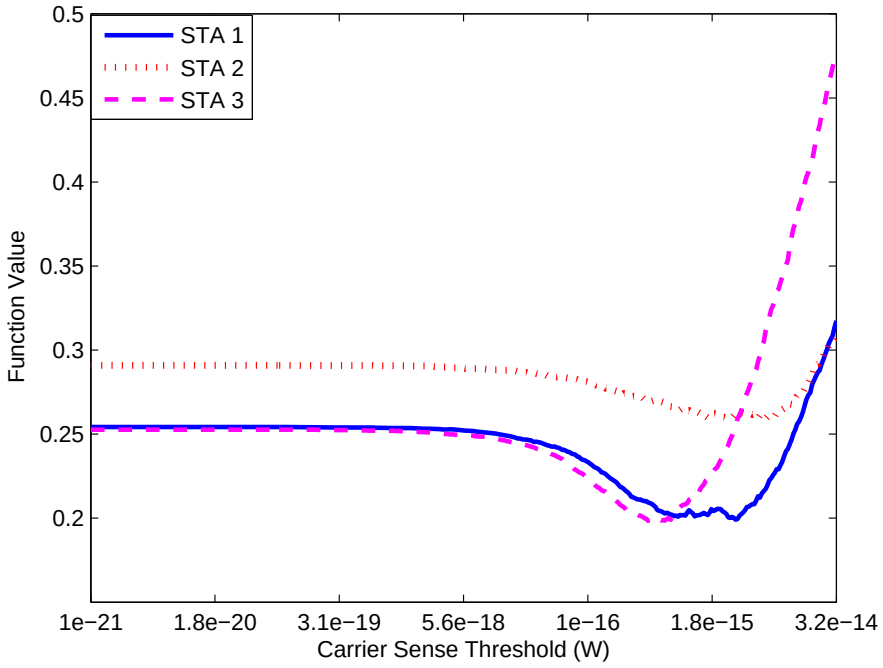


Figure 7.8. F as a function of CST.

As explained in Section 7.2, we do not change the CST in the adaptive CST algorithm unless the percentage change in the optimization function, F, is larger than a constant threshold, ρ. This is because function F exhibits shallow variations around the minimum value in many cases. Furthermore, changing the CST too frequently may prohibit the nodes from reaching their optimum values since changing the CST in one node affects the performance of other nodes which may in turn trigger them to change their CST in response. Fig. 7.8 shows the values of the optimization function, F, for three different stations with respect to the possible CST values in one round of CST updating. As seen, the value of the function shows only small variations around the minimum value in some cases. Fig. 7.9 depicts the variation of the CST vs. simulation time for three different nodes in the network for the adaptive CST algorithm. As seen, the CST values vary smoothly and infrequently over time.

7.4 Conclusions

In this chapter, we have proposed an adaptive and distributed algorithm for adaptively choosing the carrier sense threshold of nodes in IEEE 802.11 networks by utilizing the BI

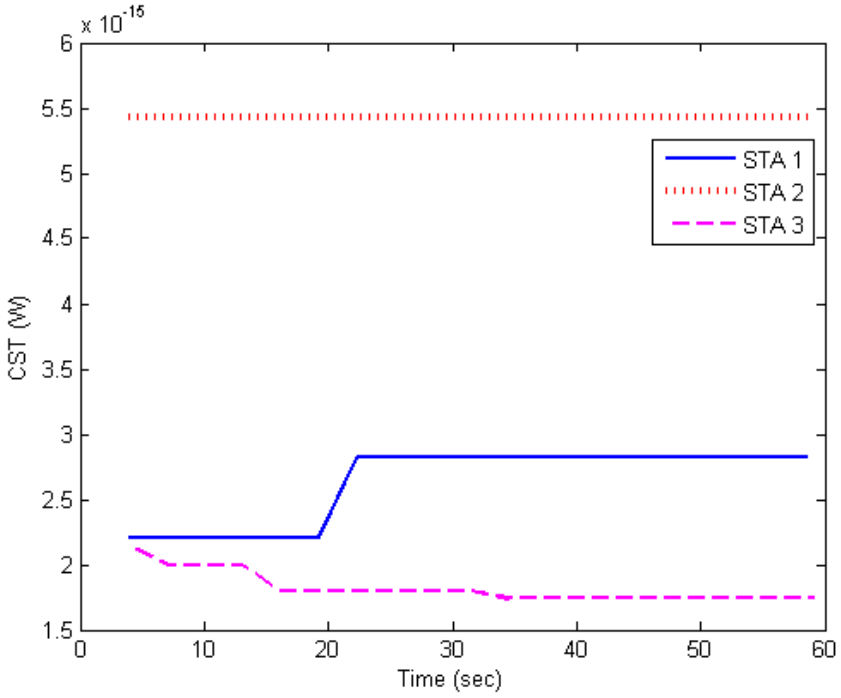


Figure 7.9. Carrier sense variation as a function of time.

signal. We have discussed the trade-off between the number of the hidden and exposed nodes and their relation to the CST. We have used NS-2 simulations to show that our approach can increase the aggregate throughput of the network by as much as 50% and that it does so in an energy efficient way.

The carrier sense threshold is a problematic parameter to adjust because it is difficult to predict the effects on nodes which do not implement the CST adaptation. Decreasing the hidden and exposed node problem for a given station may alleviate collisions for the nodes with which the station is colliding but it may also create hidden and exposed node problems for other nodes. We have shown that the overall throughput gain from our method occurs in a fair way in the sense of total network throughput, but it may also be possible to take a game-theoretic approach to the problem, which is an interesting direction for future work.

Chapter 8

Conclusion and Future Work

In this thesis, we have presented a classification of the types of collisions that occur in WLANs, and proposed a method for accurately estimating them locally at any node via limited sharing of spatial information about network traffic, specifically the BI signal. We have demonstrated an implementation of the BI signal generation and collision probability estimation technique using off-the-shelf hardware and verified that our estimates are accurate to within 5%. Additionally, we have shown that our collision probability estimates as well as the BI signal can be used to to improve throughput and utility in WLANs via adaptation of several key parameters.

We have proposed a new link adaptation algorithm, called SNRg, which provides up to a factor of five throughput gain over standard ARF in scenarios where all losses are due to collision. In scenarios in which the probability channel error is larger, it results in throughput gains of 15% or more. We proposed a novel contention window adaptation scheme in which nodes use the BI signal to optimize contention window sizes to improve network utility in a distributed manner. We have examined three cases in which the standard BEB algorithm performs poorly and shown via NS-2 simulations that our method can improve throughput by as much as 24% in the high traffic scenario, 35% in the high channel error scenario and 350% in the presence of the hidden terminal problem. We have proposed a packet length adaptation algorithm which accounts for a more detailed packet loss model than traditionally used in the literature, and shown that it can achieve up to 20% throughput gains. Finally, we have proposed an adaptive and distributed algorithm for choosing the carrier sense threshold and discussed the trade-off between the number of the hidden and exposed nodes and their relation to the carrier sense threshold. Importantly, all of these adaptations can be done with a single modification the APs to broadcast the BI signal, so a multitude of improvements can be made with the same 2% overhead.

This thesis suggests several potential directions for future work. First, it may be desirable

to extend the implementation to be able to compute estimates in real time to allow for implementation of the adaptation algorithms. This requires minor modifications to the firmware of wireless cards to allow them to store and report the carrier sensing information used to generate the busy-idle signal. Additionally, it may be possible to develop joint adaptation algorithms which combine those suggested in this thesis. For instance, since modulation rate affects packet duration as well as bit-error probability, there is a natural interaction between link adaption and packet length adaptation. Similarly, changes to the carrier sense threshold affect the actual busy-idle signal, as well as the set of hidden nodes, which may greatly impact any of the other adaptations. Due to this significant impact on the BI signal, a joint adaptation with CST must involve multiple timescales so that the other adaptations base their decisions on a relevant BI signal. Finally, the BI signal and collision probability estimates may be used to perform other adaptations not addressed in this thesis, particularly for the new higher-rate WLAN standards such as 802.11n and 802.11ac. Specifically, the BI signal may be useful for access point or channel selection as well as packet aggregation and beamforming decisions.

Bibliography

[1] Ath5k - linux wireless. [Online]. Available: http://wireless.kernel.org/en/users/Drivers/ath5k

[2] Iperf. [Online]. Available: http://iperf.sourceforge.net/

[3] Kismet. [Online]. Available: http://www.kismetwireless.net/

[4] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LANs," in *ACM SIGCOMM 1994*, London, UK, September 1994.

[5] G. Bianchi and I. Tinnirello, "Kalman filter estimation of the number of competing terminals in an IEEE 802.11 network," in *IEEE INFOCOM 2003*, San Francisco, California, April 2003.

[6] G. Bianchi, "Performance analysis of the 802.11 distributed coordination function," *IEEE JSAC*, vol. 18, no. 3, pp. 535–547, March 2000.

[7] J. Choi, J. Na, Y. sup Lim, K. Park, and C. kwon Kim, "Collision-aware design of rate adaptation for multi-rate 802.11 WLANs," *IEEE JSAC*, vol. 26, no. 8, October 2008.

[8] J. Choi, K. Park, and C. kwon Kim, "Cross-layer analysis of rate adaptation, DCF and TCP in multi-rate WLANs," in *IEEE INFOCOM 2007*, Anchorage, Alaska, May 2007.

[9] S. Choi, K. Park, and C. Kim, "On the performance characteristics of WLANs: revisited," in *ACM SIGMETRICS 2005*, Banff, Alberta, Canada, June 2005.

[10] S. Chun, D. Xianhua, L. Pingyuan, and Z. Han, "Adaptive access mechanism with optimal contention window based on node number estimation using multiple thresholds," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, June 2012.

[11] W. Dong, X. Liu, C. Chen, Y. He, G. Chen, Y. Liu, and J. Bu, "DLPC: dynamic packet length control in wireless sensor networks," in *IEEE INFOCOM 2010*, San Diego, California, March 2010.

[12] K. Fall and K. Varadhan. The ns manual. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-documentation.html

[13] S. Gollakota and D. Katabi, "Zigzag decoding: combating hidden terminals in wireless networks," in *SIGCOMM 2008*, Seattle, Washington, August 2008.

[14] X. He, F. Y. Li, and J. Lin, "Link adaptation with combined optimal frame size and rate selection in error-prone 802.11n networks," in *IEEE ISWCS 2008*, Reykjavik, Iceland, October 2008.

[15] C. Hua and R. Zheng, "Starvation modeling and identification in dense 802.11 wireless community networks," in *IEEE INFOCOM 2008*, Phoenix, Arizona, April 2008.

[16] "HFA3861B: direct sequence spread spectrum baseband processor," Data Sheet, Intersil, February 2002.

[17] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," in *DEC Technical Report 301*, 1984.

[18] A. Kamerman and L. Monteban, "WaveLAN-II: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 97–118, Summer 1997.

[19] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, 1998.

[20] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[21] H. Kim, S. Yun, H. Lee, I. Kang, and K.-Y. Choi, "A simple congestion-resilient link adaptation algorithm for IEEE 802.11 WLANs," in *IEEE GLOBECOM 2006*, San Francisco, California, November 2006.

[22] J. Kim, S. Kim, S. Choi, and D. Qiao, "Cara: Collision-aware rate adaptation for IEEE 802.11 WLANS," in *IEEE INFOCOM 2006*, Barcelona, Spain, April 2006.

[23] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, and E. Rozner, "Predictable performance optimization for wireless networks," in *ACM SIGCOMM 2008*, Seattle, Washington, August 2008.

[24] A. Lukyanenko, A. Gurtov, and E. Morozov, "An adaptive backoff protocol with markovian contention window control," *Communications in Statistics - Simulation and Computation*, vol. 41, 2012.

[25] H. Ma, R. Vijayakumar, S. Roy, and J. Zhu, "Optimizing 802.11 wireless mesh networks based on physical carrier sensing," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1550–1563, Oct. 2009.

[26] F. Magulo, M. Lacage, and T. Turletti, "Efficient collision detection for auto rate fallback algorithm," in *IEEE ISCC 2008*, Marrakech, Morocco, July 2008.

[27] MATLAB, *version 8.0.0.783 (R2012b)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[28] T. Nadeem and L. Ji, "Location-aware IEEE 802.11 for spatial reuse enhancement," *IEEE Transactions on Mobile Computing*, vol. 6, no. 10, pp. 1171–1184, Oct. 2007.

[29] ——, "Location enhancement to IEEE 802.11 DCF," *IEEE Transactions on Mobile Computing*, vol. 6, October 2007.

[30] G. A. Naydenov and P. S. Stoyanov, "Bit error period determination and optimal frame length prediction for a noisy communication channel," *AU J.T.*, vol. 11, no. 1, pp. 7–13, July 2007.

[31] Q. Pang, S. C. Liew, and V. C. M. Leung, "Design of an effective loss-distinguishable MAC protocol for 802.11 WLAN," *IEEE Communication Letters*, vol. 9, no. 9, September 2005.

[32] K.-J. Park, L. Kim, and J. Hou, "Adaptive physical carrier sense in topology-controlled wireless networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 1, pp. 87–97, Jan. 2010.

[33] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee, "Diagnosing wireless packet losses in 802.11: Separating collision from weak signal," in *IEEE INFOCOM 2008*, Phoenix, Arizona, April 2008.

[34] D. O. Reudink, "Properties of mobile radio propagation above 400 MHz," *IEEE Trans. of Veh. Technol.*, vol. 23, no. 4, November 1974.

[35] C. Sarr, C. Chaudet, G. Chelius, and I. G. Lassous, "Bandwidth estimation for IEEE 802.11-based ad hoc networks," *IEEE Trans. on Mobile Computing*, vol. 7, no. 10, October 2008.

[36] N. Song, B. Kwak, J. Song, and L. Miller, "Analysis of EIED backoff algorithm for the IEEE 802.11 DCF," in *IEEE VTC 2005 Fall*, Dallas, Texas, September 2005.

[37] W. Song, M. N. Krishnan, and A. Zakhor, "Adaptive packetization for error-prone transmission over 802.11 wlans with hidden terminals," in *IEEE MMSP 2009*, Rio De Janeiro, Brazil, October 2009.

[38] A. L. Toledo, T. Vercauteren, and X. Wang, "Adaptive optimization of IEEE 802.11 DCF based on bayesian estimation of the number of competing terminals," *IEEE Trans. on Mobile Computing*, vol. 5, no. 9, September 2006.

[39] M. C. Vuran and I. F. Akyilidiz, "Cross-layer packet size optimization for wireless terrestrial, underwater, and underground sensor networks," in *IEEE INFOCOM 2008*, Phoenix Arizona, April 2008.

[40] A. K. Vyas, F. A. Tobagi, and R. Narayanan, "Characterization of an ieee 802.11a receiver using measurements in an indoor environmant," in *IEEE GLOBECOM 2006*, San Francisco, California, November 2006.

[41] C. Weng and C. Chen, "Performance study of IEEE 802.11 DCF with optimal contention window," in *IMIS 2012*, Palermo, Italy, July 2012.

[42] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation in 802.11 wireless networks," in *ACM MOBICOM 2006*, Los Angeles, California, September 2006.

[43] Y. Xu, M. Huang, and Y. Z. M. Lin, "A self-adaptive minimum contention window adjusting backoff algorithm in IEEE 802.11 DCF," in *CECNet 2012*, YiChang, China, April 2012.

[44] Y. Yang and S. Lam, "General AIMD congestion control," in *IEEE ICNP 2000*, Osaka, Japan, November 2000.

[45] J. Yin, X. Wang, and D. P. Agrawal, "Optimal packet size in error-prone channel for IEEE 802.11 distributed coordination function," in *IEEE WCNC 2004*, Atlanta, Georgia, USA, March 2004.

[46] D. G. Yoon, S. Y. Shin, W. H. Kwon, and H. S. Park, "Packet error rate analysis of IEEE 802.11b under IEEE 802.15.4 interference," in *IEEE VTC 2006*, Melbourne, Australia, May 2006.

[47] J.-H. Yun and S.-W. Seo, "Collision detection based on transmission time information in IEEE 802.11 wireless lan," in *IEEE PERCOMW*, 2006.

[48] Z. Zeng, Y. Yang, and J. Hou, "How physical carrier sense affects system throughput in IEEE 802.11 wireless networks," in *Proceedings of the IEEE INFOCOM 2008*, April 2008, pp. 1445–1453.

[49] F. Zheng and J. Nelson, "Adaptive design for the packet length of IEEE 802.11n networks," in *IEEE ICC 2008*, Beijing, China, May 2008.

[50] J. Zhu, X. Guo, L. Lily Yang, W. Steven Conner, S. Roy, and M. M. Hazra, "Adapting physical carrier sensing to maximize spatial reuse in 802.11 mesh networks," *Wireless Communications and Mobile Computing Journal*, vol. 4, no. 8, pp. 933–946, 2004.

Appendix A

Computation of the Derivative for Contention Window

The derivative can be broken up into 3 terms: one relating to the node itself, one to its neighbors, and one to its hidden nodes.

$$\begin{aligned}\frac{\partial}{\partial \overline{W}_i} U =& \frac{\partial}{\partial \overline{W}_i} \log TP_i + \sum_{k \in \mathcal{N}_i} \frac{\partial}{\partial \overline{W}_i} \log TP_k \\ &+ \sum_{j \in \mathcal{H}_i} \frac{\partial}{\partial \overline{W}_i} \log TP_j \end{aligned} \tag{A.1}$$

We now compute each of these terms.

The first is straightforward.

$$\frac{\partial}{\partial \overline{W}_i} \log TP_i = \frac{1}{TP_i} R_i (1 - P_{Li}) \frac{\partial}{\partial \overline{W}_i} S_i \tag{A.2}$$

$$= \left(\frac{T_i}{B_i + T_i} \frac{1}{\overline{W}_i} \right)^{-1} \frac{\partial}{\partial \overline{W}_i} \left(\frac{T_i}{B_i + T_i} \frac{1}{\overline{W}_i} \right) \tag{A.3}$$

$$= -\frac{1}{\overline{W}_i} + \left(\frac{1}{T_i} - \frac{1}{B_i + T_i} \right) \frac{\partial}{\partial \overline{W}_i} T_i \tag{A.4}$$

where $\frac{\partial}{\partial \overline{W}_i} T_i$ can be computed as:

$$\frac{\partial}{\partial \overline{W}_i} T_i = \frac{\partial}{\partial \overline{W}_i} \left(1 - \prod_{k \in \mathcal{N}_i} \left(1 - \frac{1}{\overline{W}_k} \right) \right)^{-1} \tag{A.5}$$

$$= \frac{\partial}{\partial \overline{W}_i} \frac{1}{1 - (1 - P_{DCi})(1 - \frac{1}{\overline{W}_i})} \tag{A.6}$$

$$= (1 - P_{DCi}) \frac{T_i^2}{\overline{W}_i^2} \tag{A.7}$$

The second term in (A.1) is:

$$\frac{\partial}{\partial \overline{W}_i} \sum_{k \in \mathcal{N}_i} \log TP_k$$

$$= \sum_{k \in \mathcal{N}_i} \frac{1}{TP_k} \frac{R_k(1 - P_{ei})}{\overline{W}_k} \frac{\partial}{\partial \overline{W}_i} \left(\frac{T_k}{B_i + T_k} (1 - P_{DCk}) \right) \tag{A.8}$$

$$= \sum_{k \in \mathcal{N}_i} \left[\frac{B_i + T_k}{T_k} \frac{\partial}{\partial \overline{W}_i} \left(\frac{T_k}{B_i + T_k} \right) - \frac{\frac{\partial}{\partial \overline{W}_i} P_{DCk}}{1 - P_{DCk}} \right] \tag{A.9}$$

$$= \sum_{k \in \mathcal{N}_i} \left[\frac{B_i}{T_k(B_i + T_k)} \frac{\partial}{\partial \overline{W}_i} T_k - \frac{\frac{\partial}{\partial \overline{W}_i} P_{DCk}}{1 - P_{DCk}} \right] \tag{A.10}$$

$\frac{\partial}{\partial \overline{W}_i} P_{DCk}$ can be computed as:

$$\frac{\partial}{\partial \overline{W}_i} P_{DCk} = \frac{\partial}{\partial \overline{W}_i} \left[1 - \prod_{j \in \mathcal{N}_k} \left(1 - \frac{1}{\overline{W}_j} \right) \right] \tag{A.11}$$

$$= - \left[\prod_{j \in \mathcal{N}_k \setminus i} \left(1 - \frac{1}{\overline{W}_j} \right) \right] \frac{\partial}{\partial \overline{W}_i} \left(1 - \frac{1}{\overline{W}_i} \right) \tag{A.12}$$

$$= - \frac{1}{\overline{W}_i^2} \prod_{j \in \mathcal{N}_k \setminus i} \left(1 - \frac{1}{\overline{W}_j} \right) \tag{A.13}$$

$$= - \frac{1 - P_{DCk}}{1 - \frac{1}{\overline{W}_i}} \frac{1}{\overline{W}_i^2} \tag{A.14}$$

Substituting this in to Equation (A.10) yields

$$\frac{\partial}{\partial \overline{W}_i} \sum_{k \in \mathcal{N}_i} \log TP_k$$
$$= \sum_{k \in \mathcal{N}_i} \left[\frac{B_i}{T_k(B_i + T_k)} \frac{\partial}{\partial \overline{W}_i} T_k - \frac{1}{\overline{W}_i^2 - \overline{W}_i} \right] \tag{A.15}$$

If we assume $T_i \approx T_k$, which is reasonable since the nodes are neighbors, we get

$$\frac{\partial}{\partial \overline{W}_i} \sum_{k \in \mathcal{N}_i} \log TP_k$$
$$= \sum_{k \in \mathcal{N}_i} \left[\frac{B_i}{T_i(B_i + T_i)} \frac{\partial}{\partial \overline{W}_i} T_i - \frac{1}{\overline{W}_i^2 - \overline{W}_i} \right]$$
$$= |\mathcal{N}_i| \left[\frac{B_i}{T_i(B_i + T_i)} (1 - P_{DCi}) \frac{T_i^2}{\overline{W}_i^2} - \frac{1}{\overline{W}_i^2 - \overline{W}_i} \right] \tag{A.16}$$

The third term in (A.1) is:

$$\frac{\partial}{\partial \overline{W}_i} \sum_{j \in \mathcal{H}_i} \log TP_k$$
$$= \sum_{j \in \mathcal{H}_i} \frac{1}{TP_j} \frac{TP_j}{1 - P_{SCj}} \frac{\partial}{\partial \overline{W}_i} (1 - P_{SCj}) \tag{A.17}$$
$$= \sum_{j \in \mathcal{H}_i} \frac{1}{1 - P_{SCj}} \frac{(1 - P_{SCj})}{e^{-(2D-1)S_i}} \frac{\partial}{\partial \overline{W}_i} e^{-(2D-1)S_i} \tag{A.18}$$
$$= \sum_{j \in \mathcal{H}_i} (-(2D-1)) \frac{\partial}{\partial \overline{W}_i} S_i \tag{A.19}$$
$$= \sum_{j \in \mathcal{H}_i} \frac{(2D-1)T_i}{(B_i + T_i)\overline{W}_i^2} \tag{A.20}$$
$$= |\mathcal{H}_i| \frac{(2D-1)T_i}{(B_i + T_i)\overline{W}_i^2} \tag{A.21}$$

Substituting Equations (A.4), (A.16) and (A.21) into (A.1) yields the complete expression

for $\frac{\partial}{\partial \overline{W}_i} U$.

$$\begin{aligned}\frac{\partial}{\partial \overline{W}_i} U = &- \frac{1}{\overline{W}_i} + \left(\frac{1}{T_i} - \frac{1}{B_i + T_i}\right)(1 - P_{DCi})\frac{T_i^2}{\overline{W}_i^2} \\ &+ |\mathcal{N}_i| \left[\frac{B_i}{T_i(B_i + T_i)}(1 - P_{DCi})\frac{T_i^2}{\overline{W}_i^2} - \frac{1}{\overline{W}_i^2 - \overline{W}_i}\right] \\ &+ |\mathcal{H}_i| \frac{(2D-1)T_i}{(B_i + T_i)\overline{W}_i^2}\end{aligned} \tag{A.22}$$