Throughput Improvement in 802.11 WLANs using Collision Probability Estimates in Link Adaptation

Michael N. Krishnan and Avideh Zakhor
Department of EECS, U.C. Berkeley

***Abstract*—The 802.11 standard includes several modulation rates, each of which is optimal for a different channel condition. 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. In this paper, we show that an estimate of the probability of collision can be used to improve link adaptation in 802.11 networks with hidden terminals, and significantly increase throughput by up to a factor of five. We demonstrate this through NS-2 simulations of a few link adaptation techniques including a new algorithm, called SNRg.**

I. Introduction and Related Work

While the 802.11 standard includes several modulation rates, it does not provide a standardized method for selecting the appropriate rate to use at any given time. The higher rates achieve better throughput when the channel has sufficiently high signal-to-noise ratio (SNR), while the lower rates are more robust to low SNR. The process of choosing the appropriate modulation rate, called link adaptation (LA), is left to the wireless card manufacturers.

Most current LA algorithms are based on Auto-Rate Fallback (ARF) [1], 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[2]. While lowering the modulation rate reduces the probability of loss due to channel errors, it actually *increases* the probability of loss due to collisions with hidden terminals, because it results in packets with longer durations.

A number of methods have been proposed to modify ARF to overcome this shortcoming by differentiating collisions from channel errors [3–7]. In [3–5] 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, making it unattractive in practice.

The techniques in [6] and [7] focus on distinguishing collisions from channel errors on a per-packet basis. In [6], the Pang, et. al. suggest the use of negative acknowledgements (NACKs) for when packets are received with errors. The shortcoming of this approach 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; 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 [8]. In [7], 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 collision probability increases, this can result in an excessive amount of overhead.

In this paper, we argue that it is not necessary for nodes to differentiate between collisions and channel errors on a per-packet basis. All that is needed is an approximation of the *probability* that future packets will collide. In [9], 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 [10], we propose a method for nodes in an 802.11 network with hidden terminals to estimate their collision probabilities based on shared information about channel occupancy by the access points.

The basic idea behind the collision probability estimation technique in [10] is that all nodes continually measure the occupancy of the channel around them, and the access point (AP) periodically broadcasts its local information to all the nodes that it serves. The nodes then compare their own local measurements with those of the AP to obtain a spatial picture of the network traffic. Based on this information, they estimate the probability of occurrence of several different scenarios which lead to collisions, thus obtaining an accurate estimate of the probability of collision. In 802.11b, the overhead incurred by the AP broadcasts is less than 2%

In this paper we leverage this estimate to improve link adaptation and thus increase throughput in an 802.11 network with hidden terminals via two different approaches. First, we use the estimate in a modified version of ARF, based on [9]. 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 remainder of the paper is organized as follows: Section II describes the packet loss model; the link adaptation algorithms are explained in Section III, simulation results are presented in Section IV, and the paper is concluded in Section V.

II. Packet Loss Model

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. A collision occurs when a node's packet overlaps in time with that of another node which is close enough to the destination to interfere. A channel error occurs when the SNR of a received packet is low due to a large path loss or a deep multipath fade. The total packet loss probability P_L can be computed as

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

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. The following subsections discuss in detail how each type of loss occurs in carrier-sense multiple access protocols such as 802.11.

A. Collisions

There are three primary causes of collisions. The first is a direct artifact of the distributed coordination function. Since nodes access the channel randomly, there is a chance that two nodes will begin their transmission at the same time, making neither of their packets decodable at the destination. These collisions are referred to as *direct collisions(DCs)* because the packets start at the same time and directly overlap.

The other types of collisions occur due to the hidden node problem, occuring when multiple far-away nodes that cannot sense each other transmit at the same time. In these collisions, transmissions do not necessarily start at exactly the same time. As such, these collisions are referred to as *staggered collisions (SCs)*. SCs can be further subdivided in to 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 transmits 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 as a result they must be estimated and adapted to in different ways.

In our P_C estimation technique described in [10], nodes obtain spatial information about network traffic via periodically broadcast information from the AP, and use it to estimate the probability of each of these three types of collisions. In particular, each node or AP in the network generates a binary-valued *busy-idle signal*, which is a function of time, taking value 1 when there is enough energy on the local channel that the node would not be able to transmit or successfully receive a packet, and taking value 0 otherwise. The AP then periodically compresses and broadcasts its busy-idle signal to all its associated nodes. By comparing its local busy-idle signal to that of the AP, each node can estimate the probability that its packets will experience each type of collision.

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 combatted by LA, from all other losses. As such, in this paper, 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.

B. Channel Errors

For a sequence of L bits sent at a constant modulation rate R over a channel whose bit-error rate, as a function of R and SNR, is denoted by $BER_R(SNR)$, the probability of success is given by $(1 - BER_R(SNR))^L$. 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 can be computed as

$$P_e = 1 - (1 - BER_{R_h}(SNR))^{L_h}(1 - BER_{R_p}(SNR))^{L_p} \tag{2}$$

where L_h and L_p 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 $BER_R(SNR)$ is assumed to be a known function, which depends on the the signal constellation for each rate. In this paper, we fix L_h, R_h, and L_p, and adapt R_p using the LA algorithms.

Since the SNR is unknown and varies between packets, it is modeled as a random variable. Thus the probability of packet error is the expectation of the expression in Equation (2) taken over the distribution of SNR, $P_{SNR}(s)$.

$$P_e = 1 - \int_s (1 - BER_{R_h}(s))^{L_h}(1 - BER_{R_p}(s))^{L_p} P_{SNR}(s) ds \tag{3}$$

For the simulations in this paper, 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 variability in the environment.

Figure 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 [11]. Inserting these values into Equation (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 1(b).

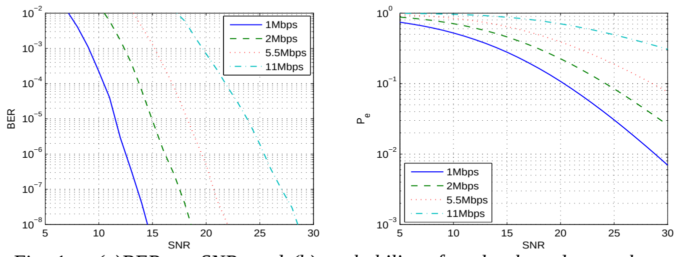

Fig. 1. *(a)BER vs. SNR, and (b) probability of packet loss due to channel error vs. mean SNR for each modulation rate. SNR standard deviation is fixed at 7 dB.*

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

$$Throughput = \frac{(1 - P_L)L_p}{\frac{L_p}{R_p} + \frac{L_h}{R_h} + T_{ov}} \tag{4}$$

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 (3) and (4), we can plot throughput as a function of mean SNR for each modulation rate as shown in Figure 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.

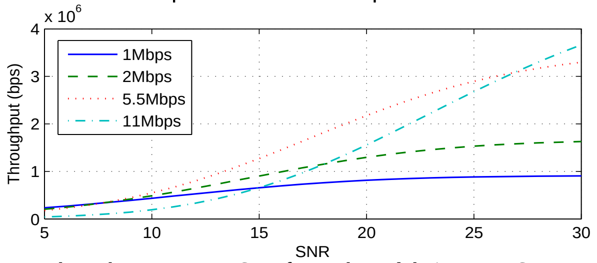

Fig. 2. *Throughput vs. mean SNR for each modulation rate. SNR standard deviation is fixed at 7 dB.*

III. Link Adaptation Algorithms

In this paper, we examine the potential throughput improvement achievable using the P_C estimate of [10] 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 of the algorithm proposed in [9]; the third is a new algorithm proposed in this paper, called SNRg.

A. 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 [1].

B. Modified COLA

The Congestion-aware Link Adaptation (COLA) algorithm proposed in [9] 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 is 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 $m+1$ failed, 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 [9], 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 the P_C estimate from [10] 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 [10]. 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 [9] 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.

C. 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

Algorithm 1 The mCOLA algorithm

```
1:  Initialize $n_t, n_s, n_f \leftarrow 0$; $\forall m, N_m \leftarrow 1$
2:  Transmit at current rate for 5 seconds
3:  Compute estimate of $P_C$ using [10]
4:  for each transmission do
5:      $n_t \leftarrow n_t + 1$
6:      if no ACK then
7:          $n_f \leftarrow n_f + 1$
8:          $n_s \leftarrow n_s + P_C$
9:          if $n_f \geq n_t \cdot P_C + k$ then
10:             $n_s \leftarrow 0$
11:             if $m \neq m_{min}$ then
12:                 $m \leftarrow m - 1$
13:                 if $n_t == n_f$ then
14:                     $N_m \leftarrow 2N_m$
15:                 end if
16:                 $n_t, n_f \leftarrow 0$
17:                 Goto 2
18:             end if
19:         end if
20:     else
21:         $n_s \leftarrow n_s + 1$
22:         if $m \neq m_{min}$ then
23:             $N_{m-1} \leftarrow 1$
24:         end if
25:         if $n_s > N_m$ and $m \neq m_{max}$ then
26:             $m \leftarrow m + 1$
27:             $N_m \leftarrow 1$
28:             $n_t, n_s, n_f \leftarrow 0$
29:             Goto 2
30:         end if
31:     end if
32: end for
```

[10], estimate P_L based on empirical measurements, and insert these into Equation (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 paper, 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, as described in [10]. 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 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 2. Pseudo-code is shown in Algorithm 2.

Algorithm 2 The SNRg algorithm

```
1:  Transmit at current rate for 5 seconds
2:  Compute estimate of $P_C$ using [10]
3:  $P_L \leftarrow n_f/n_t$
4:  if $P_L == 1$ then
5:      $m \leftarrow m - 1$
6:      Goto 1
7:  end if
8:  $P_e \leftarrow 1 - (1 - P_L)/(1 - P_C)$
9:  Lookup $SNR$ based on $P_e$ and current rate
10: Set current rate to best rate for $SNR$
11: Goto 1
```

A timing diagram comparing COLA and SNRg is shown in Figure 3. The red arrows correspond to times when P_C is estimated, and the green arrows correspond to times when the new modulation rate is chosen. Because COLA requires a variable number of transmission attempts before changing the modulation rate, the times when P_C is estimated, represented by the red 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.

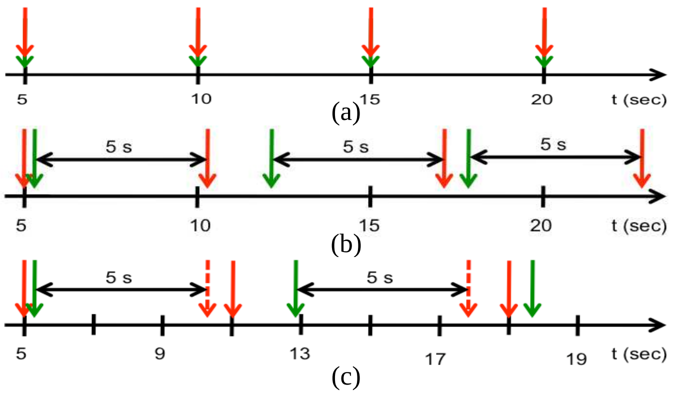

Fig. 3. *Timing diagrams for (a) SNRg; (b) idealized COLA used in simulations; (c) COLA in a practical implentation. The red arrows correspond to times when P_C is estimated, and the green arrows correspond to times when the modulation rate is chosen. In (c), the dotted red arrows correspond to ideal P_C estimation times.*

For the simulations in this paper, 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 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 dotted red arrows, it must delay its estimation to the times marked by solid red 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.

IV. 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. We have made modifications to allow all the nodes to compute their collision probabilities as in [10] as well as count the actual results of each packet transmission: collision, channel error, or success. 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 the estimation technique in [10], henceforth called *est*. The results in this paper 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. We fix $L_h = 1152$ bits, $R_h = 1$ Mbps, and $L_p = 18432$ bits, which are typical values for 802.11b. An example topology with 50 nodes is shown in Figure 6, where the squares represent the AP locations, and the circle centers correspond to the node locations. 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. As seen, when P_e gets sufficiently high, the rate drops, and when P_e decrease, the rate increases.

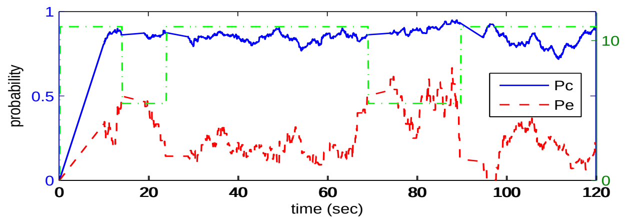

Fig. 4. *P_e, P_C, and modulation rate over time for one node using SNRg.*

Figure 5 shows the throughput improvement, with respect to ARF, of each of the four techniques using P_C estimates, as a function of noise power. Figures 5(a), 5(b), and 5(c) correspond to scenarios with 50, 30, and 10 nodes respectively. The blue lines denote mCOLA, the red lines 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, lowering the noise power in the simulation has a similar effect on SNR to clustering nodes closer to their APs.

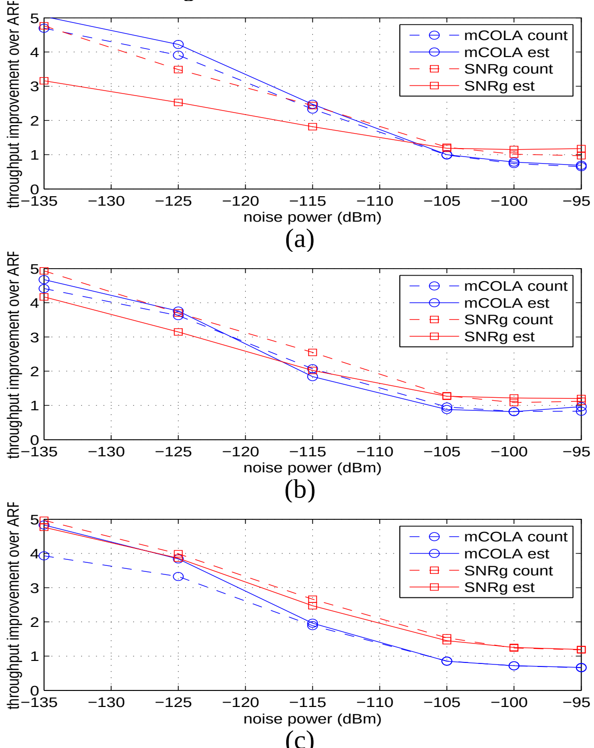

Fig. 5. *Average throughput improvement over ARF for scenarios with (a) 50 nodes; (b) 30 nodes; (c) 10 nodes.*

In most situations, there is little difference between using *count* and *est* for either algorithm. This indicates that, in general, the P_C estimate from [10] 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 6(a) and 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 which have 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.

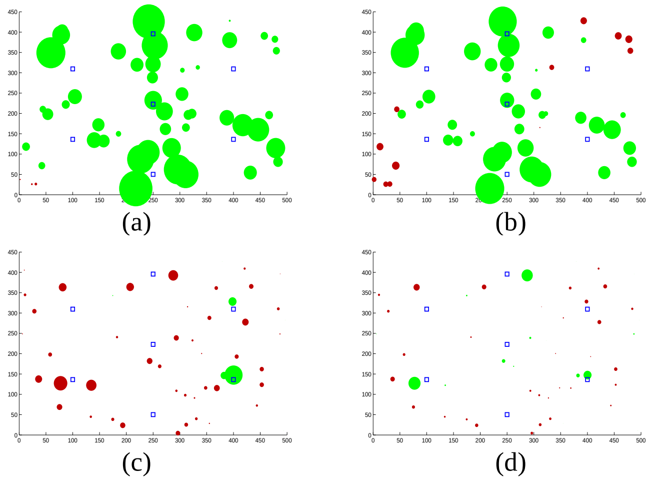
Fig. 6. *Throughput improvement over ARF of each node over space in representative topologies with 50 nodes for (a) SNRg count with noise power -125 dBm; (b) SNRg est with noise power -125 dBm; (c) mCOLA est with noise power -95 dBm; and (d) SNRg est with noise power -95 dBm. The small squares represent the position of the APs, and the circles represent the nodes. Green circles indicate nodes with increase throughput relative to ARF, and red circles indicate nodes with decreased throughput. The size of the circle is proportional to the node's change in throughput.*

At high noise levels, such as -95dBm, SNRg achieves a 15% average throughput improvement over ARF; however, mCOLA achieves less 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 6(c) and 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 causes most nodes in Figure 6(c) to have reduced throughput as compared to ARF. However, in Figure 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 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.

V. Conclusions and Future Work

In this paper we have shown that the P_C estimates obtained in [10] 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 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.

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; (b) investigation of the performance of COLA 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. Other future work includes development of a multi-parameter optimization to improve throughput using the P_C estimates, by adapting other parameters such as packet length along with modulation rate; hardware implementation and experimentation using an open-source wireless card driver or a software radio platform is also important to verify the validity of our approach.

References

[1] Ad Kamerman and Leo Monteban, "WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band", *Bell Labs Technical Journal*, Vol.2, No.3, Summer 1997, pp.118-133.

[2] Sunwoong Choi, Kihong Park, and Chong-kwon Kim, "On the Performance Characteristics of WLANs: Revisited", in *Proc. of ACM SIGMETRICS 2005*, Banff, Alberta, Canada, June 2005.

[3] Jongseok Kim, Seongkwan Kim, Sunghyun Choi, and Daji Qiao, "CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANS", in *Proc. of IEEE INFOCOM 2006*, Barcelona, Spain, April 2006.

[4] Starsky H.Y. Wong, Hao Yang, Songwu Lu, and Vanduvur Bharghavan, "Robust Rate Adaptation in 802.11 Wireless Networks", in *Proc. of ACM MOBICOM 2006*, Los Angeles, California, September 2006.

[5] Federico Magulo, Mathieu Lacage, Thierry Turletti, "Efficient Collision Detection for Auto Rate Fallback Algorithm", in *Proc. of IEEE ISCC 2008*, Marrakech, Morocco, July 2008.

[6] Qixiang Pang, Soung C. Liew, and Victor C. M. Leung, "Design of an Effective Loss-Distinguishable MAC Protocol for 802.11 WLAN", *IEEE Communication Letters*, Vol. 9, No. 9, pp. 781-783, September 2005.

[7] Ji-Hoon Yun and Seung-Woo Seo, "Collision Detection based on Transmission Time Information in IEEE 802.11 Wireless LAN", IEEE PERCOMW, 2006.

[8] Amit K. Vyas, Fouad A. Tobagi, and Rajesh Narayanan, "Characterization of an IEEE 802.11a Receiver using Measurements in an Indoor Environmant", in *Proc. of IEEE GLOBECOM 2006*, San Francisco, California, November 2006.

[9] Hyogon Kim, Sangki Yun, Heejo Lee, Inhye Kang, and Kyu-Young Choi, "A simple congestion-resilient link adaptation algorithm for IEEE 802.11 WLANs", in *Proc. of IEEE GLOBECOM 2006*, San Francisco, California, November 2006.

[10] Michael N. Krishnan, Sofie Pollin, and Avideh Zakhor, "Local Estimation of Probabilities of Direct and Staggered Collisions in 802.11 WLANs", to appear in *IEEE GLOBECOM 2009*, Honolulu, Hawaii, December 2009.

[11] Intersil, "HFA3861B: Direct Sequence Spread Spectrum Baseband Processor", Data Sheet, February 2002.