Achieving Energy Efficiency with Transmission Pushbacks in Sensor Networks

*Sha Liu, †Rahul Srivastava, †Can Emre Koksal, and *Prasun Sinha
*Department of Computer Science and Engineering
The Ohio State University
Columbus, Ohio 43210
{liusha,prasun}@cse.ohio-state.edu

†Department of Electrical and Computer Engineering
The Ohio State University
Columbus, Ohio 43210
{srivastr,koksal}@ece.osu.edu

***Abstract*— In sensor networks, application layer QoS requirements are critical to meet while conserving energy. One of the leading factors for energy wastage is failed transmission attempts due to channel dynamics and interference. Existing techniques are unaware of the channel dynamics and lead to suboptimal channel access patterns. We propose a MAC layer solution called *pushback*, that appropriately delays packet transmissions to overcome periods of poor channel quality and high interference, while ensuring that the throughput requirement of the node is met. It uses a Hidden Markov Model (HMM) based channel model that is maintained without any additional signaling overhead. The pushback algorithm is shown to improve the packet success rate by up to 71% and reduce the number of transmissions needed by up to 38% while ensuring the same throughput.**

I. Introduction

Sensor networking applications often have stringent requirements for QoS parameters such as throughput and delay. In battery operated sensor networks, such QoS requirements must be met while consuming the least amount of energy. Since a high percentage of the energy is spent on data communication, support for efficient and reliable communication is critical. However, high variability in channel quality caused by factors such as fading, mobility, and time-varying multiuser interference make it difficult to achieve those objectives. Indeed, Woo *et al.* [1], and Zhao and Govindan [2] have both observed a significant variability in link quality in wireless sensor networks with radios in the 433 MHz band. The former paper points out that the instantaneous packet error probability varies by approximately 30% around its mean. The latter paper, as well as Willig *et al.* [3], both show that the packet-error stochastic process exhibits significant long-term dependence.

Without any effort for adapting to the variability, the system resources are consumed highly inefficiently. Due to high packet loss rates, a high fraction of the energy of a node is consumed by multiple retransmissions per packet. However, in the currently available sensor hardware platforms (such as MicaZ or iMotes from Crossbow Inc.), the computational power rules out sophisticated control actions for adaptation. Only very simple strategies (e.g., transmit or do not transmit a packet at a given time) are implementable. Our objective in this paper is to design an optimum packet transmission strategy that meets the required throughput constraint while considering the dynamics of the channel.

The existing CSMA protocol is an example of an adaptive transmission technique. The CSMA protocol uses carrier sensing to avoid collisions and backoffs to address the problem of contention among nearby nodes. However, packet transmissions may fail due to cumulative interference from other nodes in the network. Indeed in our testbed experiments with Mica2 nodes, we have observed that with interfering sources that are sufficiently far away, 69% of the packets for which the CSMA granted a transmission permit are lost. From this example, we can conclude that the combined effect of a large number of interfering sources can be very detrimental and the CSMA based protocols - designed to suppress collisions - are not effective in avoiding such losses. Immediate solution to this problem is reducing the carrier sense threshold that triggers a backoff and consequently increasing the carrier sense range. This, in effect, would enable a node to sense this combined interference and hidden terminals to some extent, and avoid part of the losses. However, the increase of carrier sense range makes a node overly conservative with respect to interference and it leads to a lowered effective throughput. Therefore, simple adjustment of the carrier sense range is not sufficient to avoid transmissions during poor channel conditions.

In the context of transmission rate adjustment for 802.11 networks, several solutions have been proposed based on estimated channel conditions [4]–[7]. In contrast, our objective in this paper is to optimize the energy consumption with constraint on the throughput. Our solution methodology is also different as it explicitly models the channel based on rigorous estimation techniques. Existing MAC layer solutions for sensor networks such as [8]–[11], have not considered direct modeling of the time-varying channel quality for optimization of transmission attempts. Several back-pressure based mechanisms have been proposed at the link layer [12]–[15] to address high interference and network congestion. These approaches are orthogonal to our proposed solution, and can be used in conjunction with our solution to improve performance, as shown in Section V.

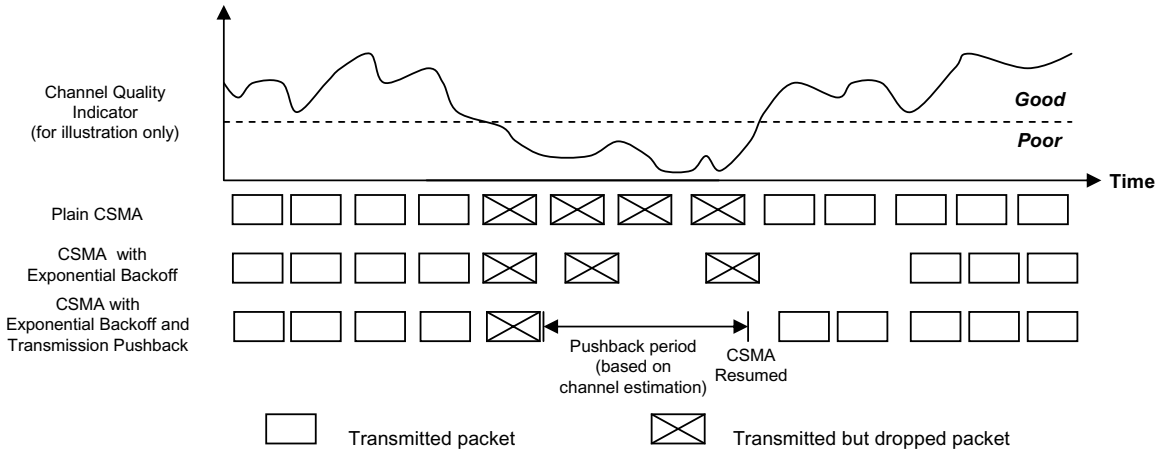


Fig. 1. Transmission Pushback: CSMA unnecessarily transmits during poor channel conditions. With exponential backoff, there will be fewer transmissions during poor channel conditions, however opportunities to transmit during good states will also be lost. With transmission pushback CSMA can avoid periods with poor channel quality, while making use of good channel states.

In this paper, we systematically study the problem of addressing packet losses due to cumulative interference, and propose a binary control technique over CSMA. Our approach is based on exploiting the temporal correlations of the interference process. We introduce a new concept called *transmission pushbacks*, which refers to an appropriately computed delay introduced at the MAC layer in order to avoid periods with bad-channel quality while considering a node's throughput requirement. Therefore, we reduce the number of transmissions per packet as well as the number of transmission attempts per unit time. In case of bursty losses, avoiding the bad channel state may also lead to a higher throughput (visible at higher layers) despite lower number of transmission attempts.

The main idea of transmission pushbacks is to defer packet transmission attempts for an appropriately selected period upon failed packet transmissions. Figure 1 illustrates the benefits of using transmission pushbacks in comparison with CSMA based approaches in the presence of time-varying channels. Plain CSMA leads to failed transmissions, and thus wasted energy, during periods with poor channel quality. CSMA with exponential backoff may reduce such failed transmissions, but it also cuts down the transmission attempts, even at times of improved channel quality. Our proposed transmission pushback mechanism predicts the duration for which the channel quality will remain poor. Thus, unnecessary transmissions can be avoided to conserve energy and the good channel states are taken advantage of. The contrast is also highlighted in Table I which shows that Pushback along with CSMA with exponential backoff can handle various types of packet losses. Observe that as pushback operates over the CSMA algorithm, improvements of the CSMA mechanism such as [16] are orthogonal to pushback.

To determine the pushback time, we need to estimate the channel quality and how it varies over time. We use an adaptive channel prediction technique, based on estimating the parameters of a simple hidden Markov model (HMM), which represents our channel. The parameters of the HMM are dynamically updated based solely on the binary ACK sequence (transmission success or failure) for the previous packet transmissions. We choose the appropriate pushback period by considering the throughput requirement measured by the incoming data rate, and the predicted quality of the channel. Such an adjustment in rate, based on the throughput requirement is also seen in lazy packet schedulers (e.g., [17]). The proposed approach is simple to implement over existing CSMA based MAC solutions, as well as queue and congestion control algorithms. Therefore it is highly suited for existing sensor network platforms.

TABLE I

TYPES OF LOSSES ADDRESSED BY VARIOUS MECHANISMS

Mechanisms	Designed for collision losses	Designed for interference and channel losses
CSMA Plain	Partially	No
CSMA/EB	Yes	Partially
CSMA/EB+Pushback	Yes	Yes

This paper makes the following contributions:

- Using data collected from a sensor network testbed, temporal characteristics of the channel variations and the interference are studied.
- A novel concept called *pushbacks* is introduced, that is used to increase the packet success rate while considering the throughput constraint at each node.
- Through simulations it is shown that significant gains in energy and/or throughput can be observed in all scenarios using the proposed technique.

The rest of the paper is organized as follows. Section II presents related work. Section III presents our approach to model the channel losses. Section IV presents a description of our pushback algorithm. Section V presents evaluation of the proposed scheme. Finally, Section VI concludes the paper and presents pointers to future research directions.

II. Related Work

Transmission Rate Adaptation: Transmission strategies based on channel estimation has been considered in the context of 802.11 networks [4]–[7]. More specifically, the past packet success and failure reports have been used to design strategies to dynamically adapt the physical layer transmission rate to optimize the throughput. ARF [4] uses a heuristic to predict the channel quality based on past transmission success and failure records, but it is ignorant of the underlying time-varying properties of the channel. RBAR [5] uses RTS/CTS to get immediate feedback from the destination to learn about the quality of the channel and determine the transmission rate. However, these packets have high overhead especially in sensor networks where the data packet sizes are comparable to the size of RTS/CTS packets. In OAR [6] during good channel periods the transmitter opportunistically transmits multiple packets back-to-back at a high data rate. In contrast to the opportunistic nature of [6], our solution uses a rigorous channel model to predict the duration for which the channel will continue in a poor state. In [7], authors present a history based mechanism to predict the quality of the channel and adjust the transmission rate. Our objective of optimizing energy consumption for a given throughput constraint is different from the past work. Our solution methodology is also different as it uses *rigorous estimation of dynamic channel properties*.

MAC and Link Layer for Sensor Networks: Various MAC layer protocols have been proposed to conserve energy in sensor networks by sleep scheduling. These protocols can be categorized as either synchronized or unsynchronized. SMAC [10], TMAC [18] and DMAC [11] are examples of synchronized MAC layer protocols. These MAC protocols require periodic message exchange for synchronization. In these protocols nodes wake up for a short duration at synchronized times and sleep otherwise to conserve energy. In order to address the overhead of synchronization, unsynchronized approaches have also been proposed. BMAC [19] and SP [20] are two such examples that make use of long preambles before data transmission to ensure that the receiving node will receive the packet. More recently, CMAC [9] and XMAC [8] have been proposed to avoid the overhead of long preambles. Anycasting has been considered in the joint design of MAC and routing layers to operate in unsynchronized networks [21]–[24]. However, none of these solutions have considered the time-varying properties of the underlying channels in the design of the channel access mechanism.

In order to address high interference and network congestion, several back-pressure based mechanisms have been proposed for sensor networks [12]–[15]. CODA [12] uses a moving average of channel samples to detect the onset of congestion and sends back-pressure messages to control the data rate of upstream nodes. Fusion [13] uses the concept of hop-by-hop flow control and prioritized MAC along with token buckets to control the packet rate at each node. IFRC [14] and RAIN [15] use variations of the AIMD (Additive Increase Multiplicative Decrease) and the back-pressure mechanisms. Various schemes have been proposed that select the backoff counter [25] and the backoff window [26] based on the channel state.

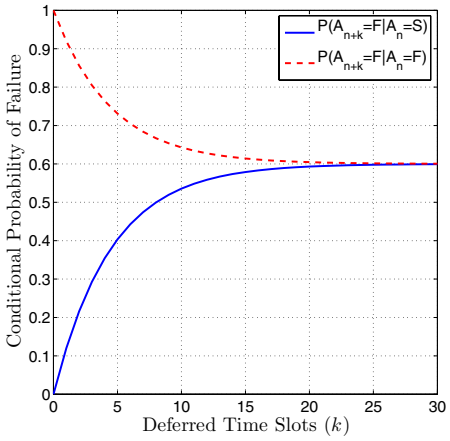


Fig. 2. Conditional probability of failure as a function of deferred time slots setting $p = 0.6$ and $\alpha = 0.8$.

We conclude this section by observing that these solutions are unaware of the time-varying characteristics of the channel, and hence still do blind retransmissions ignorant of current channel condition. In addition, MAC protocols with backoffs tuned to channel conditions do not address the time-scales associated with channel variations as shown in Figure 1. On contrary, the pushback algorithm proposed in this paper can learn the channel characteristics and schedule retransmissions accordingly. In fact, our solutions can be used in conjunction with the above link layer mechanisms and bring extra benefit as shown in our simulation results.

III. Channel Model

In this section we describe our channel loss model and its parameters and give an experimental justification for our model. We use this channel loss model to derive the theoretical expressions for Packet Success Ratio (PSR) and throughput as functions of channel and system parameters. We describe how to estimate these parameters based on the available measurements at the sensor nodes in the next section.

A. Channel Model

Let $A_n,\ n \geqslant 1$ be the process, which takes on an 'S' or an 'F' depending on whether a packet transmitted at time n is successfully decoded at the receiver or not. We assume that only one packet can be transmitted in each time slot and that A_n is a *wide-sense Markov* of order 1 for the packet error process. We will discuss the validity of this assumption later, but first we describe A_n further.

A sequence $A_n, n \geq 1$ is said to be wide-sense Markov if the probability of a future value, A_{n+m}, is completely

determined by its most recent value, i.e. A_n and the time difference (m) between the two events. The autocovariance function for such a process is exponential. More specifically, suppose the process is in steady state[1] at time n and let $P(A_n = \mathrm{F}) = p$. Then the autocovariance function of A_n is $K_A(m) = p(1-p)\alpha^{|m|}$ for some α, $|\alpha| < 1$. Here p is the long-term loss probability and α quantifies the "coherence" or correlation between a successful (failed) transmission in the future based on the present observation. The loss probability p increases with the number of interfering users and the noise in the channel and α is a measure for the burst length for the failed transmissions.

For this wide-sense Markov process, the probability of a successful (failed) transmission in a future time slot, conditioned on a successful (failed) transmission in the present slot, is unique (Appendix I):

$$P(A_{n+m} = \mathrm{F}|A_n = \mathrm{S}) = p(1-\alpha^m) \quad (1)$$
$$P(A_{n+m} = \mathrm{S}|A_n = \mathrm{S}) = 1-p(1-\alpha^m) \quad (2)$$
$$P(A_{n+m} = \mathrm{F}|A_n = \mathrm{F}) = p+(1-p)\alpha^m \quad (3)$$
$$P(A_{n+m} = \mathrm{S}|A_n = \mathrm{F}) = 1-p-(1-p)\alpha^m. \quad (4)$$

Hence only a pair of parameters is sufficient to represent the channel with the first order Markov assumption. For the purpose of illustration, the conditional probabilities of failure for $p = 0.6$ and $\alpha = 0.8$ are plotted in Fig. 2. The "deferred time slots" represents the number, k, of time slots waited before the next transmission attempt after an event S or F.

We can form an underlying Markov chain corresponding to this wide-sense Markov process as follows. The Markov chain has two states S (Good) and F (Bad) and the chain makes a transition between these states depending on A_n taking on an S or an F as shown in Figure 3. One can evaluate the parameters p and α for the associated Markov process given the transition probabilities, x and y for this Markov chain and vice versa as shown in Section III-B. The channel model is an HMM, since x and y are unknown initially and we estimate them based on the observed values of A_n using a maximum likelihood estimator. Based on these estimated values, we calculate the two channel parameters α and p, which is detailed in Section IV. Note that the reason for estimating x and y initially is due to the simplicity of the task (based on observation of A_n). The reason why we pursue our analyses with α and p afterwards, instead of x and y, is that the performance metrics of the pushback algorithm are tied to α and p more naturally.

In fact, HMMs have been used to model the channel behavior previously by representing the (bit or packet level) error process in wireless communications [27]–[29]. While not as simple as a Bernoulli or an independent loss channel model, Markov models are more capable of characterizing the statistical dependencies which might occur in a wireless channel.

[1]We assume that the process started at time $n = -\infty$. In practice the data transmission is only finite, however we make this technical assumption, since we do not have any information about the initial channel state.

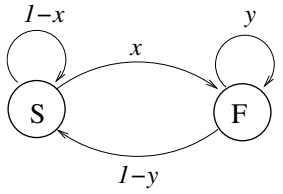


Fig. 3. Markov chain representation of the channel.

From the two curves in Fig. 2, one can see the reasoning behind choosing a pushback duration conditional on an event F only. If we schedule the next packet for immediate transmission (i.e., $k = 1$) after an S, we have the best chance of observing another S. Intuitively, we are taking advantage of the good channel state. On the other hand, if we defer scheduling the transmission (i.e., $k > 1$) of the next packet after an F, we lower the probability of failing in that transmission. The longer the deferral time, the higher the probability of an S in the next transmission. However, waiting indefinitely can cause the throughput to drop significantly. So we need to strike a balance between these two requirements, throughput and probability of success. Hence, with the Markov channel model, the problem reduces to finding the appropriate pushback period after a failed transmission. In the next subsection, we will derive the expressions for the throughput and packet success rate for this scheme using our channel model.

B. Channel Parameters

To find the channel parameters, packet success ratio and throughput, we sketch the Markov chain associated with our first order Markov process. Our main objective here is to find the throughput as a function of the number of deferred time slots, k, on a transmission failure[2]. Once we have this function, we can choose k according to the desired throughput based on the incoming data rate. To validate this model using real data, we also find the expression for the PSR.

Our Markov chain has two states, S and F as illustrated in Fig. 3. The current state is S if the final packet transmission is successful and F, otherwise. Note that a transition does not necessarily occur every time slot, rather it occurs for every packet transmission attempt. Since we schedule a transmission immediately after a successful event, the expression for x is obtained by substituting $k = 1$ in (1). Direct application of (3) gives the expression for y.

$$x = P(A_{n+1} = \mathrm{F}|A_n = \mathrm{S}) = p(1-\alpha) \quad (5)$$
$$y = P(A_{n+k} = \mathrm{F}|A_n = \mathrm{F}) = p+(1-p)\alpha^k \quad (6)$$

Notice that the transition probabilities at state F are functions of k as well as p and α. This is due to the effect of the pushback period of k time slots after the failed transmission

[2]Upon a successful transmission, the deferral time is 1 (no deferral).

attempts. The associated steady state probabilities are therefore functions of k as well, and these probabilities for state S and state F are respectively,

$$\pi_{\mathrm{S}}(k) = \frac{(1-p)(1-\alpha^k)}{p(1-\alpha)+(1-p)(1-\alpha^k)} \qquad (7)$$
$$\pi_{\mathrm{F}}(k) = 1 - \pi_{\mathrm{S}}(k).$$

We define the packet success ratio (PSR) as the total fraction of the packets that are successfully transmitted, i.e., it is equal to the steady state probability, $\pi_{\mathrm{S}}(k)$ of state S.

To formulate an expression for throughput, consider the following: on a transmission attempt, we wait for k time slots in state F and 1 time slot in state S for the next transmission attempt. Thus, the average number of slots per attempt is $\pi_S(k) + k\pi_F(k)$. Consequently the number of packet transmissions per slot,

$$X(k) = \frac{1}{\pi_S(k)+k\pi_F(k)} = \frac{p(1-\alpha)+(1-p)(1-\alpha^k)}{kp(1-\alpha)+(1-p)(1-\alpha^k)}. \qquad (8)$$

The resulting throughput, $\rho(k)$, is thus

$$\rho(k) = \pi_{\mathrm{S}}(k)X(k) = \frac{(1-p)(1-\alpha^k)}{kp(1-\alpha)+(1-p)(1-\alpha^k)}. \qquad (9)$$

We tested the validity of our Markov model using an experimental setup in the Kansei testbed [30] by setting up a wireless link between two sensor motes. This link was encircled with seven sensor motes spread around the perimeter of a 20 m^2 area. We measured the packet success rate between two motes, while the rest of the motes acted as sources of interference. For this experiment, the wireless link transmitted 36 byte packets at 100 ms interval. We programmed the nodes causing interference to transmit bursts of packets once every second. The burst size was selected according to a uniform distribution between [0,15]. We estimated the two parameters, α and p using the ACK-level data of the entire trace of 30 mins. Note that the purpose of this experiment was to validate the model. In our actual algorithm, the estimation of the two parameters is much simpler and does not depend on a long trace of data. The theoretical packet success rate based on the estimated α and its actual experimental value are plotted in Fig. 4 as a function of k. This plot gives credence to the Markov model.

IV. The Pushback Algorithm

The objective of the pushback algorithm is to estimate the period for which the channel will remain in a poor state and defer retransmissions accordingly in order to conserve energy. In addition, the algorithm must provide similar throughput as CSMA for a reduced number of transmission attempts.

The proposed pushback algorithm is based on CSMA. If a transmission is successful, the next transmission is scheduled by CSMA. However, in case of a failed transmission, the next transmission is pushed back by k slots after which CSMA takes effect. Note that a pushback "slot" is a packet slot, i.e., the time it takes to transmit a packet. Therefore it is different from the contention slot of CSMA. Also, even though the value of k is generated deterministically, the randomness of the backoff period (in contention slots) of CSMA avoids persistent pushbacks caused by local synchronization of nodes due to similar channel conditions.

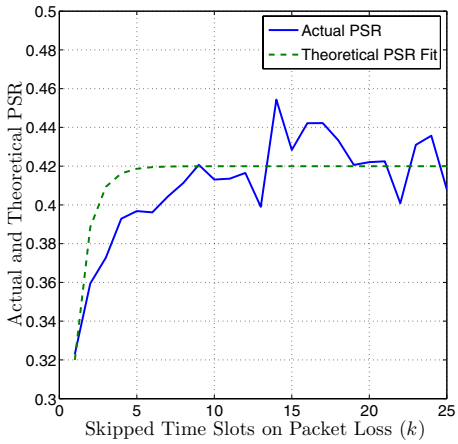


Fig. 4. Comparison of the actual PSR gain achieved by our pushback algorithm and the theoretical fit using (7).

When nodes boot up, an initial value k_{init} is assigned to k, and then k is recalculated periodically each time the m-th (predefined) transmission failure happens using a simple mechanism based on the estimates of the channel parameters and throughput constraint. Our computation is both practical to implement and is also shown to perform well using simulations in Section V.

The proposed mechanism for computing k is based on the formulation presented in Section III. First, k is set to an initial value k_{init}, and based on the ACK-level observations of success and failure in the recent past, a maximum likelihood (ML) estimation of parameters α and p is made. In fact, estimating the transition probabilities of the Markov chain (parameters x and y in Fig. 3) is sufficient for the desired ML estimation. Indeed, given the ML estimates $\hat{x}$ and $\hat{y}$ (of x and y respectively), we show in the technical report [31] that the solution $(\hat{\alpha}, \hat{p})$ to the system of Equations (5) and (6) gives the ML estimates of α and p which characterize the channel. In addition, to ensure that a node can sustain the incoming rate of packets, the computation of k must take the incoming packet rate into account. We use the running average of the incoming packet rate, $\rho_{\text{new}} = \gamma/t_e + (1-\gamma)\rho_{\text{old}}$, to represent the throughput constraint. Here ρ_{new} (ρ_{old}) is the new (previous) estimate of the throughput requirement, t_e is the time elapsed since the last packet arrival and γ is smoothing factor. Using ρ_{new} (throughput constraint), $\hat{\alpha}$ and $\hat{p}$, Equation (9) can be used to compute the new value of k.

To avoid the complexity of direct computation of k, we

TABLE II

LOOKUP TABLES USED IN THE PUSHBACK ALGORITHM.

Table	Equations	Purpose
$\mathrm{T}_\alpha(x, y, k)$	Eqns. (5), (6)	To compute α for given x, y and k.
$\mathrm{T}_k(p, \alpha, \rho)$	Eqn. (9)	To compute k for given p, α and ρ.

propose the use of look-up tables. The first table $T_{\hat{\alpha}}(x, y, k)$ contains the values of $\hat{\alpha}$ corresponding to k and discretized x and y. The second table $T_k(\hat{p}, \hat{\alpha}, \rho)$ contains the values of $\rho(k)$ corresponding to k and discretized α and p. A brief description of the various tables used is given in Table II. These tables will not change during the operation of the node, so they can be computed offline and uploaded. The available storage spaces on the nodes will determine the size of the tables. From our experimental experience, it should suffice to have a $10 \times 20 \times 20$ table. For instance, this table can have 10 values of k (2 to 11), 20 values of p (0 to 0.95 in increments of 0.05) and 20 values of α (0 to 0.95 in increments of 0.05). These numbers could be stored as integers between 0 and 100. Hence, the two tables would take 8K bytes.

In summary, upon the m-th transmission failure, function **Pushback()** (Algorithm 1) is called. In this algorithm, The ML estimates $\hat{x}$ and $\hat{y}$ are calculated in lines 3 and 4. A table lookup is employed to find the value of $\hat{\alpha}$ corresponding to $\hat{x}$, $\hat{y}$ and k, and then $\hat{p}$ can be calculated according to Equation (5). Finally the pushback period k is estimated using another table lookup with the appropriate values of $\hat{p}$, $\hat{\alpha}$ and ρ.

Algorithm 1: Pushback()

1 **if** $(failureCount = m)$ **then**
2 $\quad failureCount \leftarrow 0$;
3 $\quad \hat{x} \leftarrow \frac{\text{Number of S}\rightarrow\text{F transitions}}{\text{Total number of stays in S states}}$;
4 $\quad \hat{y} \leftarrow \frac{\text{Number of F}\rightarrow\text{F transitions}}{\text{Total number of stays in F states}}$;
5 $\quad \hat{\alpha} \leftarrow \mathrm{T}_{\hat{\alpha}}(\hat{x}, \hat{y}, k)$;
6 $\quad \hat{p} \leftarrow \hat{x}/(1 - \hat{\alpha})$;
7 $\quad k \leftarrow \mathrm{T}_k(\hat{p}, \hat{\alpha}, \rho)$;
8 **end**
9 Delay the retransmission for k slots;

A. Remedial Mechanisms

The pushback algorithm above can work well if the real packet loss pattern is captured well by our channel model introduced earlier and the transition probabilities are accurately measured. However, either of them may deviate from reality, in which case the throughput may not be maintained if k is chosen too aggressively. Hence, we introduce two remedial mechanisms to solve such problems.

1) Measuring Actual Pushback Amount: In our pushback algorithm, the delay amount, k slots, is calculated according to the state transition probabilities and the throughput constraint. However, after delaying k slots, nodes may need to delay their retransmissions further due to contentions from other senders. This could lead to the loss in throughput since the delaying amount is longer than expected by the model. Hence, the running average of the difference between the calculated delay amount and the actual delay amount is maintained, and subtracted from the newly calculated k.

2) Controlling the Pushback Amount at the Interface Queue: Once our channel model deviates from the actual channel, simply adjusting the k as in Section IV-A.1 may still not work very well. To cope with such situations, we let the interface queue impose a pushback control policy to speed up the packet forwarding once the queue is excessively built up. This policy simply commands the pushback algorithm to fall back to CSMA (using $k = 1$) if the queue length is above a certain threshold. In our evaluations, this value is set to half of the queue capacity.

V. SIMULATION EVALUATION

We conduct simulations in *ns2* [32] to compare the performance of our pushback algorithm with plain CSMA with and without binary exponential backoff in wireless sensor networks. Here the CSMA without exponential backoff (denoted as CSMA) simulates BMAC, the default MAC layer protocol, while CSMA with exponential backoff (denoted as CSMA/EB) represents other general CSMA protocols. We also study the performance improvement when the pushback algorithm works jointly with other congestion control mechanisms such as based on rate control and back-pressure. In this section, the radio propagation model used in our simulations is introduced, followed by the simulation results.

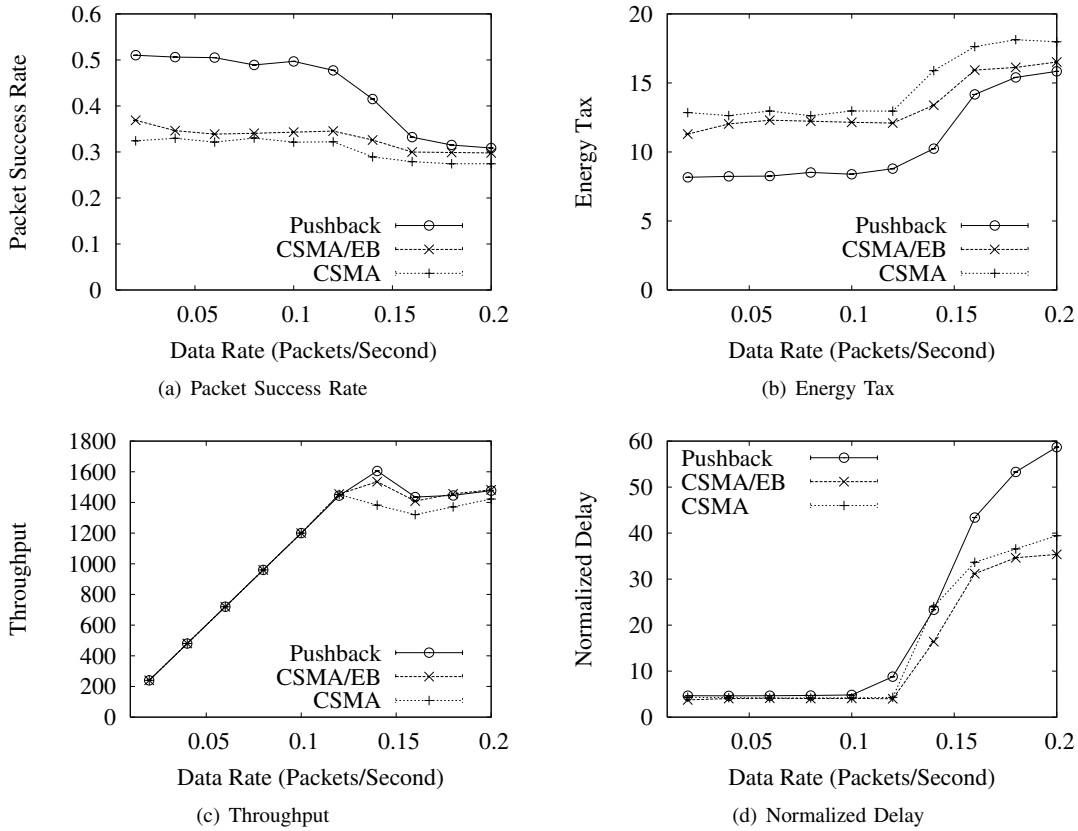
A. Radio Model

The CSMA (MAC and PHY) protocol simulated in *ns2* has two shortcomings. First, it fails to consider interference from nodes outside the carrier sensing range. However, the cumulative interference from more than one node sufficiently far away may still affect packet receptions. Second, it does not calculate the packet loss probability according to the Signal-to-Noise Ratio (SNR). In our simulations, we extended the physical layer of *ns2* to combine all sources of noise and interference to calculate the SNR and then use Equation (9) in [33] to calculate the packet success rate.

In our simulations, we use a radio propagation model based on the shadowing model implemented in *ns2*. Consequently, the received power level at a receiver is determined by

$$\left[\frac{P_{\text{rec}}(d)}{\overline{P}_{\text{rec}}(d_0)}\right]_{\text{dB}} = -10\beta \log\left(\frac{d}{d_0}\right) + X_{\text{dB}},$$

where $P_{\text{rec}}(d)$ is the received power at this receiver which is at a distance d away from the sender, β is the path loss exponent, $\overline{P}_{\text{rec}}(d_0)$ is the average received power level at a reference distance d_0, and X_{dB} is a Gaussian random variable with mean 0 and standard deviation σ_{dB} (called shadowing deviation).



(a) Packet Success Rate (b) Energy Tax

(c) Throughput (d) Normalized Delay

Fig. 5. Simulation results for various data rates.

In standard *ns2*, X_{dB} is independent for different packets. However, X_{dB} usually varies according to some random process (see e.g., [33] [34]) and we use an order 1 autoregressive model (AR(1)) as follows.

$$X_{\text{dB}}(t) = \phi X_{\text{dB}}(t-1) + Z(t),$$

where ϕ is called the channel coherence coefficient which quantifies the memory in channel variations and $Z(t)$, the error term, is independently and identically distributed with normal distribution $\mathcal{N}(0, \sigma_Z^2)$. To make the variance of $X_{\text{dB}}(t)$ independent of ϕ, we choose $\sigma_Z = \sigma_{\text{dB}}\sqrt{1-\phi^2}$. In our simulations, the time is discretized such that 1 slot is roughly equal to the average time to transmit a packet, and the value of $X_{\text{dB}}(t)$ is constant within a time slot. Note that in the modified shadowing model, if the autoregression coefficient $\phi = 0$, then the model just falls back to the default shadowing model provided by *ns2*. In our evaluation we also study the impact of different values of the channel coherence coefficient, ϕ.

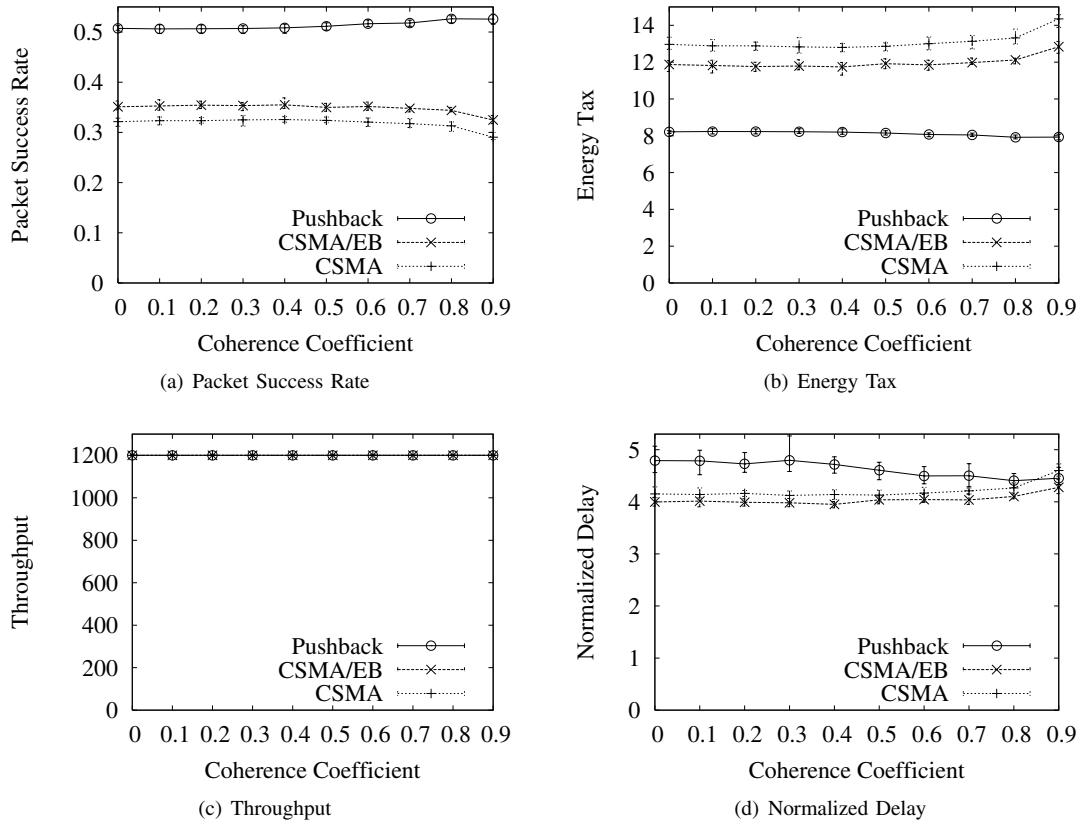
B. Simulation Evaluations

We conduct simulation evaluations on our pushback algorithm in data gathering networks. The node located at one corner of the area serves as the sink, while all other nodes generate data periodically to be sent to the sink. We evaluate the performance of all three protocols (CSMA, CSMA/EB and CSMA/EB with Pushback) under different data rates, channel coherence coefficients ϕ, and network sizes. We also show that Pushback can be used to enhance the performance of rate control and backpressure based algorithms. Additional results for different shadowing deviations, node densities in grid and random topology, and packet sizes can be found in the technical report [31]. The metrics focused on in this study include the following four.

- **Throughput**: number of packets received at the sink in 500 seconds.
- **Packet success rate (PSR)**: average success rate for each transmission attempt in the network.
- **Energy tax**: average number of transmissions needed to deliver one packet to the sink.
- **Normalized delay**: average delay per hop.

Each set of simulations is carried out for 10 times with different random seeds, and the error bar denoting the minimum and maximum values of each simulation set is also plotted. In all simulations, the default parameter values simulating the XSM nodes [35] are summarized in Table III.

1) Data Rates: The pushback algorithm takes advantage of the flexibility provided by nodes that can afford to delay the retransmissions (e.g., the ones away from the sink) without reducing the throughput. On the other hand, some nodes (e.g., those close to the sink) may have very small room



(a) Packet Success Rate (b) Energy Tax

(c) Throughput (d) Normalized Delay

Fig. 6. Simulation results for various channel coherence coefficients ϕ in a 5×5 network.

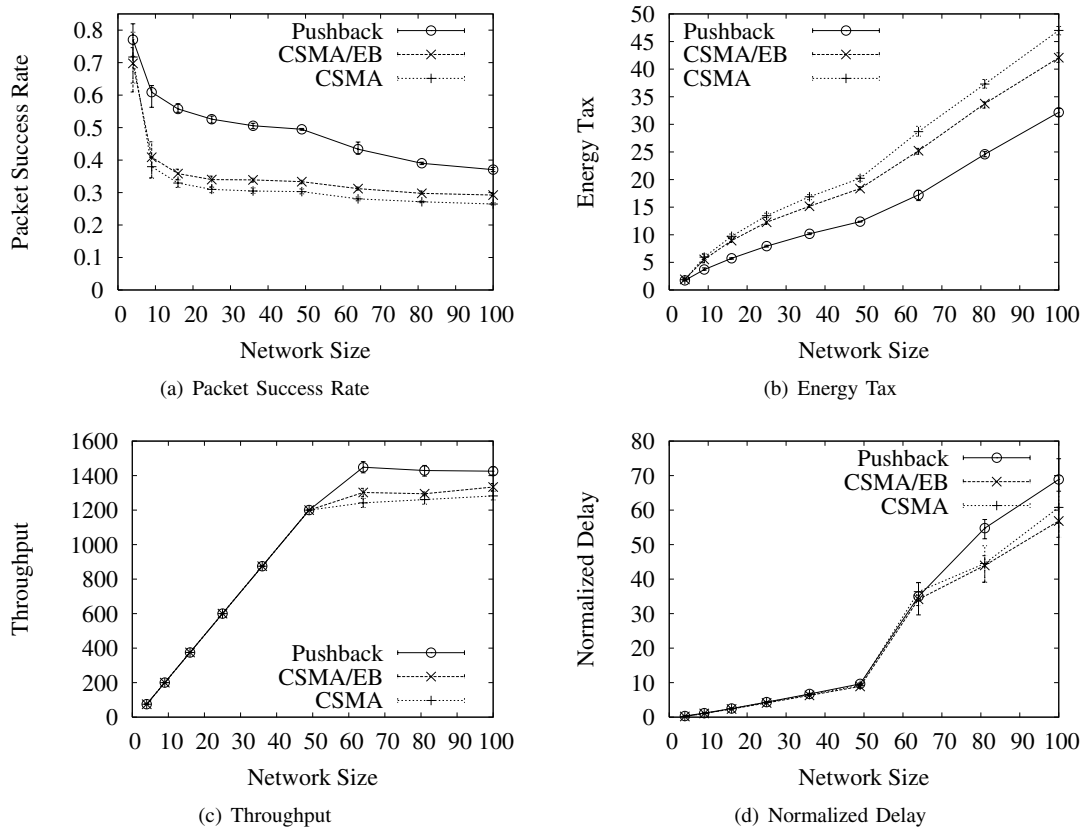
TABLE III
SIMULATION PARAMETERS

Packet Size	100 bytes	Ack Size	5 bytes
Bandwidth	19.2 Kbps	Transmit Power	0 dBm
Backoff Slot	0.4167 μs	Pushback Slot	18.33 ms
β	4 [33]	σ_{dB}	4
ϕ	0.8	Data Rate	0.1 packet/sec.
Number of nodes	25 (5×5)	Node separation	45 m

for pushback (i.e., $k \approx 1$) since they need to accommodate higher data rates. In this section, we evaluate the pushback algorithm for data generation rate at each non-sink node varying from 0.01 packets/second (pps) to 0.2 pps. Fig. 5 shows the simulation results, from which it can be observed that when data rate is low (< 0.15 pps in this case), the pushback algorithm can improve the PSR by 51% and 71% when compared to CSMA/EB and CSMA respectively. Observe that the network throughput is saturated beyond 0.15 pps data rate. For even higher data rates, the improvement in PSR is smaller, but the throughput is higher than the CSMA protocols. This implies that with pushbacks the queue drop rates are reduced. Similar improvement in energy tax can also be observed. Note that for data rates higher than 0.18 pps the queuing delay caused by the pushback algorithm results in higher normalized delay. The queuing delay is also caused due to the higher throughput provided by pushback. In this region, the network's capacity is reached, as depicted by the throughput curve. If applications can tolerate delay then even at data rates beyond the network's capacity, pushback is preferable over CSMA protocols.

2) Channel Coherence Coefficient: The pushback algorithm is very effective in the presence of temporal correlation in channel losses. Such correlations may be caused by channel coherence or correlated interference. In this section, we evaluate the performance of the pushback algorithm under channel coherence coefficients ϕ ranging from 0 to 0.8. Fig. 6 shows the simulation results. It can be seen that by utilizing the pushback algorithm, the improvement in PSR over CSMA/EB is between 46% to 63%, and it increases with ϕ. Similarly, there is a reduction in energy tax between 33% to 38%, which increases with ϕ. Meanwhile, the throughput is maintained, and even though the latency is higher, the difference is small for large ϕ. As expected, the pushback algorithm is more effective when the channel is more coherent. *The results also show that even in case of low channel coherence, our pushback algorithm can bring significant benefit in PSR and energy tax.*

3) Network Size: The benefit of the pushback algorithm depends on the amount of delay nodes can afford before retransmissions as discussed in Section V-B.1. In a large network, even though the rate of data generated at each node



(a) Packet Success Rate (b) Energy Tax

(c) Throughput (d) Normalized Delay

Fig. 7. Simulation results for various network sizes.

may be low, the cumulative data rates at nodes close to the sink may still be high. Hence, we evaluate the pushback algorithm for various network sizes in this section. We vary the number of rows and columns in a grid network from 2 to 10 (i.e., the network size varies from 4 to 100 nodes) while keeping other parameters fixed as in Table III. The results are shown Fig. 7. One can observe that the pushback algorithm can bring the most benefit in PSR when the network size is smaller than 64. For larger networks, the pushback algorithm can still provide overall improvement due to the relatively lower data rates at nodes not in the hot-spots. Throughput for large network sizes can also be improved even though the delay is higher due to the queuing delay for delivering more packets.

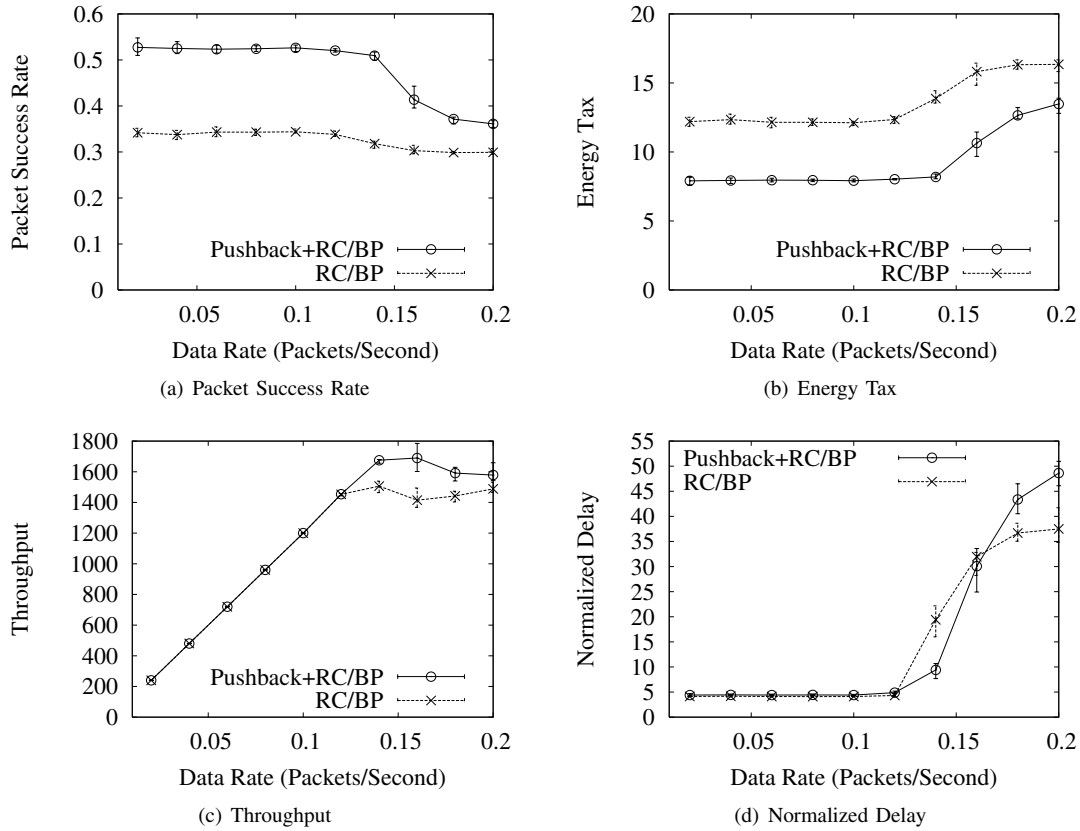
4) Cooperation with Rate Control and Back Pressure: Many link layer protocols [12]–[15] use packet rate control and back-pressure techniques to mitigate the congestion in the network. These techniques can work in conjunction with and benefit from the pushback algorithm. To show this, we implement and test the rate limiting and back-pressure mechanisms (denoted as RC/BP) proposed in [13] along with the pushback algorithm. Fig. 8(a) and 8(b) show the PSR and energy tax when CSMA/EB and RC/BP are used with and without the pushback algorithm under different data rates. It can be seen that the two metrics have similar variation trends as in Fig. 5, which shows that the pushback algorithm can still bring significant benefit when other congestion control mechanisms are used in conjunction. The results for throughput and delay (Fig. 8(c)-(d)) also show similar trends as in Fig. 5.

VI. Conclusions and Future Research

This paper introduces a channel aware transmission pushback mechanism to optimize energy efficiency. Using a simple but effective packet loss model, this approach does not incur high computational overhead on the sensor nodes. Using simulations we show that the pushback algorithm can significantly improve the packet success rate and the energy tax without degrading the throughput. In addition, this algorithm is easy to implement over existing MAC and link layer protocols. Hence, we conclude that the pushback algorithm is highly suitable for energy constrained wireless sensor networks.

Future research directions based on the concepts introduced in this paper are described below.

Push-Backs in Other Networks: This paper has focused on the application of the novel concept of *pushbacks* in the context of sensor networks. However, the concept of *pushbacks* when applied to other networking scenarios such as ad-hoc networks and mesh networks, can be used to optimize other parameters such as the number of transmissions.



(a) Packet Success Rate (b) Energy Tax

(c) Throughput (d) Normalized Delay

Fig. 8. Simulation results for the cooperation of the pushback algorithm and RC/BP.

In these networks, reduced interference due to reduction in number of transmission is expected to result in increased throughput.

Joint Optimization of Transmission Parameters: In this work we have used channel quality prediction to appropriately delay transmissions. However, channel quality prediction can be used to adjust other parameters such as physical layer data rate, transmission power, and carrier-sense threshold, some of which are inter-related.

VII. Acknowledgments

This material is based upon work partially supported by the National Science Foundation under Grants CNS-0546630 (CAREER Award), CNS-0721434, CNS-0721817 and CNS-0403342. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Appendix I
Derivation of the Conditional Probabilities

In this section we derive the conditional probabilities of a wide-sense Markov process given in Eqns. (1)–(4). A wide-sense Markov process has an exponential autocovariance function. In our analysis, we assume that the autocovariance function is of the form $K_A(m) = p(1-p)\alpha^{|m|}$ and the unconditioned probability of failure ($\mathrm{P}(A_n = \mathrm{F})$) and success ($\mathrm{P}(A_n = \mathrm{S})$) are p and $1-p$ respectively.

First, we derive the probabilities conditioned on failure, i.e., Eqns. (3) and (4). To facilitate the evaluation of the expectation, we code the event of Failure (F) as '1' and the event of Success (S) as '0'. Covariance is defined as,

$$\begin{aligned}
K_A(m) &= \mathrm{E}[A_{n+m}A_n] - \mathrm{E}[A_{n+m}]\mathrm{E}[A_n] \\
&= \sum_{x\in A_{n+m}} \sum_{y\in A_n} xy\mathrm{P}(A_{n+m}=x|A_n=y)\,\mathrm{P}(A_n=y) \\
&\quad - \sum_{x\in A_{n+m}} x\mathrm{P}(A_{n+m}=x) \sum_{y\in A_n} y\mathrm{P}(A_n=y) \\
&= \sum_{x\in\{0,1\}} \sum_{y\in\{0,1\}} xy\mathrm{P}(A_{n+m}=x|A_n=y)\,\mathrm{P}(A_n=y) \\
&\quad - \sum_{x\in\{0,1\}} x\mathrm{P}(A_{n+m}=x) \sum_{y\in\{0,1\}} y\mathrm{P}(A_n=y) \\
&= \mathrm{P}(A_{n+m}=1|A_n=1)\,p - p^2. \qquad (10)
\end{aligned}$$

For $m \geq 0$, we can compare Eqn. (10) to the autocovariance function for a wide-sense Markov process,

$$K_A(m) = p(1-p)\alpha^m = \mathrm{P}(A_{n+m}=1|A_n=1)\,p - p^2,$$

to get Eqn. (3) or $\mathrm{P}(A_{n+m}=1|A_n=1) = p + (1-p)\alpha^m$. Using the fact, $\mathrm{P}(A_{n+m}=0|A_n=1) =$

$1 - \mathrm{P}(A_{n+m} = 1|A_n = 1)$, we get Eqn. (4) or $\mathrm{P}(A_{n+m} = 0|A_n = 1) = 1 - p - (1-p)\alpha^m$.

To derive the probabilities conditioned on success, i.e., Eqns. (1) and (2), we follow the same steps with minor modifications. In this case, we code the event of Success (S) as '1' and the event of Failure (F) as '0'. From the definition of Covariance, we get,

$$K_A(m) = p(1-p)\alpha^m = \mathrm{P}(A_{n+m} = 1|A_n = 1)(1-p) - (1-p)^2. \quad (11)$$

On simplification of Eqn. (11), we get Eqn. (2) or $\mathrm{P}(A_{n+m} = 1|A_n = 1) = 1 - p(1 - \alpha^m)$. Finally, $\mathrm{P}(A_{n+m} = 0|A_n = 1) = 1 - \mathrm{P}(A_{n+m} = 1|A_n = 1)$ gives Eqn. (1) or $\mathrm{P}(A_{n+m} = 0|A_n = 1) = p(1 - \alpha^m)$.

REFERENCES

[1] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proc. of ACM SenSys*, Oct. 2003, pp. 14–27.

[2] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance in Dense Wireless Sensor Networks," in *Proc. of ACM SenSys*, Oct. 2003, pp. 1–13.

[3] A. Willig, M. Kubisch, H. Christian, and A. Wolisz, "Measurements of a Wireless Link in an Industrial Environment Using an 802.11-Compilant Physical Layer," *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1265–1282, Dec. 2002.

[4] A. Kamerman and L. Monteban, "WaveLAN II: A High-performance Wireless LAN for the Unlicensed Band," in *Bell Labs Technical Journal*, vol. 2, no. 3, 1997, pp. 118–133.

[5] G. Holland, N. Vaidya, and P. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in *Proc. of ACM MOBICOM*, Jul. 2001, pp. 236–251.

[6] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knighlty, "Opportunistic Media Access for Multirate Ad Hoc Networks," in *Proc. of ACM MOBICOM*, Jul. 2002, pp. 24–35.

[7] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust Rate Adaptation for 802.11 Wireless Networks," in *Proc. of ACM MOBICOM*, Sept. 2006, pp. 146–157.

[8] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks," in *Proc. of ACM SenSys*, Nov. 2006, pp. 307–320.

[9] S. Liu, K.-W. Fan, and P. Sinha, "CMAC: An Energy Efficient MAC Layer Protocol Using Convergent Packet Forwarding for Wireless Sensor Networks," in *Proc. of IEEE SECON*, Jun. 2007.

[10] W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 3, pp. 493–506, Jun. 2004.

[11] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks," in *Proc. of IPDPS*, Apr. 2004, pp. 224–231.

[12] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," in *Proc. of ACM SenSys*, Oct. 2003, pp. 266–279.

[13] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks," in *Proc. of ACM SenSys*, Nov. 2004, pp. 134–147.

[14] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-aware Fair Rate Control in Wireless Sensor Networks," in *Proc. of ACM SIGCOMM*, Sept. 2006, pp. 63–74.

[15] C. Lim, H. Luo, and C.-H. Choi, "RAIN: A Reliable Wireless Network Architecture," in *Proc. of IEEE ICNP*, Nov. 2006, pp. 228–237.

[16] M. Heusse, F. Rousseau, R. Guillier, and A. Duda, "Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs," in *ACM SIGCOMM*, Nov. 2005, pp. 134–147.

[17] B. Prabhakar, E. U. Biyikoglu, and A. E. Gamal, "Energy-Efficient Transmission over a Wireless Link via Lazy Packet Scheduling," in *Proc. of IEEE INFOCOM*, Apr. 2001, pp. 386–394.

[18] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Procotol for Wireless Sensor Networks," in *Proc. of ACM SenSys*, Nov. 2003, pp. 171–180.

[19] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proc. of ACM SenSys*, Nov. 2004, pp. 95–107.

[20] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A Unifying Link Abstraction for Wireless Sensor Networks," in *Proc. of ACM SenSys*, Nov. 2005, pp. 76–89.

[21] M. Zorzi and R. R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Multihop Performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 337–348, Oct. 2003.

[22] ——, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance," *IEEE Trans. Mobile Comput.*, vol. 2, no. 4, pp. 349–365, Oct. 2003.

[23] S. Jain and S. R. Das, "Exploiting Path Diversity in the Link Layer in Wireless Ad Hoc Networks," in *Proc. of WoWMoM*, Jun. 2005, pp. 22–30.

[24] R. R. Choudhury and N. H. Vaidya, "MAC-Layer Anycasting in Ad Hoc Networks," *SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 75–80, Jan. 2004.

[25] J.-G. Choi and S. Bahk, "Channel aware MAC scheme based on CSMA/CA," in *Proc. of IEEE VTC'04-Spring*, May 2004, pp. 1559–1563.

[26] P. Papadimitratos, A. Mishra, and D. Rosenburgh, "A cross-layer design approach to enhance 802.15.4," in *Proc. of IEEE MILCOM*, Oct. 2005, pp. 1719–1726.

[27] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1266, Sept. 1960.

[28] B. D. Fritchman, "A Binary Characterization Using Partitioned Markov Chains," *IEEE Trans. Inform. Theory*, vol. 13, no. 2, pp. 221–227, Apr. 1967.

[29] W. Turin and M. M. Sondhi, "Modeling Error Sources in Digital Channels," *IEEE J. Select. Areas Commun.*, vol. 11, no. 3, pp. 340–347, Apr. 1993.

[30] A. Arora, E. Ertin, R. Ramnath, W. Leal, and M. Nesterenko, "Kansei: A High-Fidelity Sensing Testbed," *IEEE Internet Computing*, vol. 10, no. 2, pp. 35–47, Mar. 2006.

[31] S. Liu, R. Srivastava, C. E. Koksal, and P. Sinha, "Optimizing Energy Consumption with Transmission Pushbacks in Sensor Networks," Technical Report, Department of Computer Science and Engineering, Ohio State University, Feb. 2008. [Online]. Available: http://www.cse.ohio-state.edu/~prasun/publications/tr/pushback.pdf

[32] "The Network Simulator – ns-2," http://www.isi.edu/nsnam/ns/.

[33] M. Zuniga and B. Krishnamachari, "Analyzing the Transitional Region in Low Power Wireless Links," in *Proc. of IEEE SECON*, Oct. 2004, pp. 517–526.

[34] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Prentice Hall, 2001.

[35] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events," in *Proc. of IPSN*, Apr. 2005, pp. 497–502.